Entwicklerhandbuch

AWS Key Management Service

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Key Management Service: Entwicklerhandbuch

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Handelsmarken und Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

AWS Key Management Service ... 1
Warum verwenden AWS KMS? ... 1
AWS KMS in AWS-Regionen ... 1
AWS KMS Preisgestaltung ... 1
AWS KMS Service Level Agreement ... 2
Zugreifen AWS Key Management Service ... 3
AWS Management Console ... 3
Für die Verwendung der AWS KMS Konsole sind Berechtigungen erforderlich ... 3
AWS Command Line Interface ... 4
AWS KMS REST API ... 4
AWS SDKs ... 4
Arbeitet mit AWS SDKs ... 5
AWS Encryption SDK ... 6
AWS KMS letztendliche Konsistenz ... 6
Hybrid-Post-Quantum-TLS ... 7
Über Post-Quantum-TLS ... 8
Verwendung ... 9
Konfigurieren Sie hybrides Post-Quantum-TLS ... 10
Weitere Informationen ... 12
Stellen Sie eine Connect AWS KMS über einen VPC-Endpunkt her ... 12
Erstellen Sie einen VPC-Endpunkt für AWS KMS ... 14
Stellen Sie eine Connect zu einem VPC-Endpunkt her ... 15
Verwenden Sie VPC-Endpunkte, um den Zugriff auf Ressourcen zu kontrollieren AWS KMS ... 16
Protokollierung von AWS KMS Anfragen, die einen VPC-Endpunkt verwenden ... 20
Dual-Stack-Endpunkte ... 21
Funktionen, die nicht verfügbar sind über IPv6 ... 21
Konzepte ... 22
Einführung ... 22
Designziele ... 23
AWS KMS keys ... 24
Kundenverwaltete Schlüssel ... 27
Von AWS verwaltete Schlüssel ... 28
AWS-eigene Schlüssel ... 29

AWS KMS key Hierarchie ... 30
Schlüsselkennungen () KeyId ... 32
Asymmetrische Schlüssel ... 35
HMAC-Schlüssel ... 37
ML-DSA-Schlüssel ... 39
Multiregionale Schlüssel ... 40
Importiertes Schlüsselmaterial ... 51
KMS-Schlüssel in einem CloudHSM-Schlüsselspeicher ... 59
KMS-Schlüssel in externen Schlüsselspeichern ... 61
AWS KMS Grundlagen der Kryptografie ... 65
Entropie und Zufallszahlengenerierung ... 65
Symmetrische Schlüsseloperationen (nur Verschlüsselung) ... 66
Asymmetrische Schlüsseloperationen (Verschlüsselung, digitale Signatur und Signaturüberprüfung) ... 66
Schlüsselableitungsfunktionen ... 67
AWS KMS interne Verwendung digitaler Signaturen ... 67
Umschlagverschlüsselung ... 68
Kryptografische Operationen ... 69
Unterstützte Algorithmen ... 71
Kryptografische Algorithmen ... 72
KMS-Schlüsselzugriff und Berechtigungen ... 75
KMS-Schlüsselrichtlinien ... 75
KMS-Schlüsselzuschüsse ... 76
Schlüsselrichtlinien ... 77
Erstellen einer Schlüsselrichtlinie ... 78
Standardschlüsselrichtlinie ... 85
Wichtige Richtlinien anzeigen ... 100
Ändern Sie eine wichtige Richtlinie ... 103
Berechtigungen für AWS Dienste ... 106
IAM-Richtlinien ... 106
Mehreren IAM-Prinzipalen Zugriff auf einen KMS-Schlüssel gewähren ... 108
Bewährte Methoden für IAM-Richtlinien ... 109
Angeben von KMS-Schlüsseln in IAM-Richtlinienanweisungen ... 112
Beispiele ... 115
Richtlinien zur Ressourcenkontrolle ... 122
Gewährungen ... 125

Konzepte für Erteilungen ... 127
Bewährte Methoden ... 132
Steuerung des Zugriffs auf Erteilungen ... 133
Erstellen einer Erteilung ... 135
Anzeigen einer Erteilung ... 144
Verwenden eines Erteilungs-Token ... 145
Außerbetriebnahme und Widerruf von Erteilungen ... 146
Bedingungsschlüssel ... 148
AWS globale Bedingungsschlüssel ... 148
AWS KMS Bedingungsschlüssel ... 152
AWS KMS Bedingungsschlüssel für AWS Nitro Enclaves ... 226
Berechtigungen mit den geringsten Rechten ... 230
Implementieren der geringsten Berechtigungen ... 232
Attributbasierte Zugriffskontrolle (ABAC) ... 235
ABAC-Bedingungsschlüssel für AWS KMS ... 236
Tags oder Aliasse? ... 240
Problembehandlung bei ABAC für AWS KMS ... 242
Rollenbasierte Zugriffskontrolle (RBAC) ... 247
Kontoübergreifender Zugriff ... 249
Schritt 1: Hinzufügen einer Schlüsselrichtlinienanweisung im lokalen Konto ... 251
Schritt 2: Hinzufügen von IAM-Richtlinien im externen Konto ... 255
Zulassen der Verwendung externer KMS-Schlüssel mit AWS-Services ... 257
Verwenden von KMS-Schlüsseln in anderen Konten ... 257
Steuern Sie den Zugriff auf Schlüssel für mehrere Regionen ... 258
Grundlagen der Autorisierung für multiregionale Schlüssel ... 259
Autorisieren von Administratoren und Benutzern für multiregionale Schlüssel ... 261
Bestimmen des Zugriffs ... 266
Untersuchen der Schlüsselrichtlinie ... 266
Untersuchen von IAM-Richtlinien ... 270
Prüfen von Erteilungen ... 272
Verschlüsselungskontext ... 273
Regeln für den Verschlüsselungskontext ... 274
Verschlüsselungskontext in Richtlinien ... 275
Verschlüsselungskontext in Erteilungen ... 276
Protokollieren des Verschlüsselungskontexts ... 277
Speichern des Verschlüsselungskontexts ... 277

Testen der Berechtigungen ... 277
Was ist DryRun? ... 278
DryRun Mit der API spezifizieren ... 279
Problembehandlung bei Berechtigungen AWS KMS ... 280
Beispiel 1: Dem Benutzer wird der Zugriff auf einen KMS-Schlüssel in seinem AWS-Konto . 281
Beispiel 2: Der Benutzer übernimmt die Rolle mit der Erlaubnis, einen KMS-Schlüssel in einem anderen System zu verwenden AWS-Konto ... 282
Glossar ... 286
Authentifizierung ... 286
Autorisierung ... 286
Authentifizierung mit Identitäten ... 286
Verwalten des Zugriffs mit Richtlinien ... 290
AWS KMS Ressourcen ... 293
Erstellen Sie einen KMS-Schlüssel ... 295
Berechtigungen zum Erstellen von KMS-Schlüsseln ... 297
Wählen Sie aus, welche Art von KMS-Schlüssel erstellt werden soll ... 298
Erstellen Sie einen KMS-Schlüssel für die symmetrische Verschlüsselung ... 301
Erstellen eines asymmetrischen KMS-Schlüssels ... 307
Erstellen Sie einen HMAC-KMS-Schlüssel ... 314
Primärschlüssel für mehrere Regionen erstellen ... 320
Erstellen Sie Replikatschlüssel für mehrere Regionen ... 325
Schritt 1: Wählen Sie Replikatregionen ... 326
Schritt 2: Replikatschlüssel erstellen ... 327
Erstellen Sie einen KMS-Schlüssel mit importiertem Schlüsselmaterial ... 333
Berechtigungen zum Importieren von Schlüsselmaterial ... 335
Anforderungen an importiertes Schlüsselmaterial ... 336
Schritt AWS KMS key 1: Erstellen Sie eine ohne Schlüsselmaterial ... 339
Schritt 2: Herunterladen des öffentlichen Verpackungsschlüssels und des Import-Tokens ... 342
Schritt 3: Verschlüsselung des Schlüsselmaterials ... 352
Schritt 4: Importieren des Schlüsselmaterials ... 362
Erstellen Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher ... 369
Erstellen Sie einen neuen KMS-Schlüssel in Ihrem CloudHSM-Schlüsselspeicher ... 370
Erstellen Sie einen KMS-Schlüssel in externen Schlüsselspeichern ... 377
Anforderungen an einen KMS-Schlüssel in einem externen Schlüsselspeicher ... 379
Erstellen Sie einen neuen KMS-Schlüssel in Ihrem externen Schlüsselspeicher ... 380
Schlüssel identifizieren und anzeigen ... 389

Finden Sie die Schlüssel-ID und den Schlüssel-ARN ... 389
KMS-Schlüsseldetails aufrufen und auflisten ... 391
Identifizieren Sie verschiedene Schlüsseltypen ... 401
Identifizieren Sie asymmetrische KMS-Schlüssel ... 401
Identifizieren Sie HMAC-KMS-Schlüssel ... 403
Identifizieren Sie KMS-Schlüssel für mehrere Regionen ... 403
Identifizieren Sie KMS-Schlüssel mit importiertem Schlüsselmaterial ... 404
Identifizieren Sie KMS-Schlüssel in AWS CloudHSM Schlüsselspeichern ... 405
Identifizieren Sie KMS-Schlüssel in externen Schlüsselspeichern ... 406
Passen Sie Ihre Konsolenansicht an ... 407
Sortieren und filtern Sie Ihre KMS-Schlüssel ... 408
Passen Sie Ihre KMS-Schlüsseltabellen an ... 410
Suchen Sie KMS-Schlüssel und Schlüsselmaterial in einem AWS CloudHSM Schlüsselspeicher ... 412
Suchen Sie die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher ... 413
Suchen Sie nach allen Schlüsseln für einen AWS CloudHSM Schlüsselspeicher ... 415
Suchen Sie den KMS-Schlüssel für einen AWS CloudHSM Schlüssel ... 417
Suchen Sie den AWS CloudHSM Schlüssel für einen KMS-Schlüssel ... 422
Tasten aktivieren und deaktivieren ... 426
Schlüssel rotieren ... 429
Warum sollten KMS-Schlüssel rotiert werden? ... 431
So funktioniert die Schlüsselrotation ... 432
Automatische Schlüsselrotation aktivieren ... 437
Deaktivieren Sie die automatische Schlüsselrotation ... 440
Führen Sie die Schlüsselrotation bei Bedarf durch ... 442
Initiierung der Schlüsselrotation bei Bedarf (Konsole) ... 442
Initiierung der On-Demand-Schlüsselrotation (API)AWS KMS ... 444
Rotationen und wichtige Materialien auflisten ... 445
Rotationen und wichtige Materialien auflisten (Konsole) ... 445
Rotationen und Schlüsselmaterialien auflisten (API)AWS KMS ... 446
Schlüssel manuell drehen ... 448
Ändern Sie den Primärschlüssel in einer Reihe von Schlüsseln für mehrere Regionen ... 450
Aktualisieren Sie die primäre Region ... 451
Löschen von Schlüsseln ... 455
Über die Wartezeit ... 456
Besondere Überlegungen ... 457

Steuert den Zugriff auf das Löschen von Schlüsseln ... 460
Erlauben Sie Schlüsseladministratoren, das Löschen von Schlüsseln zu planen und abzubrechen ... 461
Löschen von Schlüsseln planen ... 463
... 464
Löschen des Schlüssels abbrechen ... 465
Alarm erstellen ... 466
Ermitteln Sie die frühere Verwendung eines KMS-Schlüssels ... 469
Untersuchen Sie die Schlüsselberechtigungen für KMS, um den Umfang der potenziellen Nutzung zu ermitteln ... 469
Untersuchen Sie die AWS CloudTrail Protokolle, um die tatsächliche Nutzung zu ermitteln . 470
Importiertes Schlüsselmaterial löschen ... 473
Datenschlüssel generieren ... 476
Erstellen eines Datenschlüssels ... 476
Wie funktionieren kryptografische Operationen mit Datenschlüsseln ... 477
Verschlüsseln von Daten mit einem Datenschlüssel ... 477
Entschlüsseln von Daten mit einem Datenschlüssel ... 477
Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel ... 477
Generieren Sie Datenschlüsselpaare ... 480
Erstellen eines Datenschlüsselpaars ... 480
So funktionieren kryptografische Operationen mit Datenschlüsselpaaren ... 481
Verschlüsseln von Daten mit einem Datenschlüsselpaar ... 481
Entschlüsseln von Daten mit einem Datenschlüsselpaar ... 481
Signieren von Nachrichten mit einem Datenschlüsselpaar ... 482
Überprüfen einer Signatur mit einem Datenschlüsselpaar ... 482
Leiten Sie ein gemeinsames Geheimnis mit Datenschlüsselpaaren ab ... 483
Führen Sie Offline-Operationen mit öffentlichen Schlüsseln durch ... 484
Besondere Überlegungen zum Herunterladen öffentlicher Schlüssel ... 485
Laden Sie den öffentlichen Schlüssel herunter ... 486
Beispiel für Offline-Operationen ... 488
Offline-Ableitung gemeinsam genutzter Geheimnisse ... 488
Offline-Überprüfung mit ML-DSA-Schlüsselpaaren ... 490
Offline-Überprüfung mit SM2 Schlüsselpaaren (nur Regionen China) ... 493
Tasten überwachen ... 499
Überwachungstools ... 500
Automatisierte Tools ... 500

Manuelle Tools ... 501
Protokollierung mit AWS CloudTrail ... 502
Suchen nach AWS KMS Protokolleinträgen in CloudTrail ... 503
AWS KMS Ereignisse aus einer Spur ausschließen ... 504
Beispiele für AWS KMS Protokolleinträge ... 505
Überwachen Sie die Schlüssel mit CloudWatch ... 590
AWS KMS Metriken und Dimensionen ... 591
Einen CloudWatch Alarm für den Ablauf von importiertem Schlüsselmaterial erstellen ... 601
CloudWatch Alarme für externe Schlüsselspeicher erstellen ... 602
Tasten mit Amazon überwachen EventBridge ... 607
KMS-CMK-Drehung ... 607
Ablauf der über KMS importierten Schlüsselinformationen ... 608
KMS-CMK-Löschung ... 609
Aliasnamen ... 610
Wie funktionieren Aliase ... 611
Steuern des Zugriffs auf Aliasse ... 614
km: CreateAlias ... 615
km: ListAliases ... 616
km: UpdateAlias ... 616
km: DeleteAlias ... 618
Begrenzen von Alias-Berechtigungen ... 619
Aliase erstellen ... 620
Suchen Sie den Aliasnamen und den Alias-ARN ... 623
Aliase aktualisieren ... 627
Löscht einen Alias ... 629
Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern ... 630
km: RequestAlias ... 632
km: ResourceAliases ... 633
Erfahren Sie, wie Sie Aliase in Ihren Anwendungen verwenden ... 634
Suchen Sie nach Aliasen in Protokollen AWS CloudTrail ... 636
Tags ... 638
Zugriffssteuerung mit Tags ... 639
Tag-Berechtigungen in Richtlinien ... 640
Beschränken von Tag-Berechtigungen ... 642
Tags hinzufügen ... 644
Hinzufügen von Tags beim Erstellen eines KMS-Schlüssels ... 645

Fügen Sie Tags zu vorhandenen KMS-Schlüsseln hinzu ... 646
Bearbeiten von Tags ... 648
Tags entfernen ... 649
Anzeigen von Tags ... 650
Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern ... 652
Wichtige Geschäfte ... 657
AWS KMS Standard-Schlüsselspeicher ... 657
AWS KMS Standard-Schlüsselspeicher mit importiertem Schlüsselmaterial ... 658
AWS KMS benutzerdefinierte Schlüsselspeicher ... 659
AWS CloudHSM Schlüsselspeicher ... 660
Externer Schlüsselspeicher ... 660
AWS CloudHSM wichtige Geschäfte ... 661
AWS CloudHSM Konzepte für Schlüsselspeicher ... 664
Kontrollieren Sie den Zugang zu Ihrem AWS CloudHSM Schlüsselspeicher ... 668
Erstellen Sie einen AWS CloudHSM Schlüsselspeicher ... 669
Sehen Sie sich einen AWS CloudHSM Schlüsselspeicher an ... 677
Einstellungen AWS CloudHSM für den Schlüsselspeicher bearbeiten ... 680
Einen AWS CloudHSM Schlüsselspeicher Connect ... 684
Verbindung zu einem AWS CloudHSM Schlüsselspeicher trennen ... 689
Einen AWS CloudHSM Schlüsselspeicher löschen ... 692
Fehlerbehebung für einen Custom Key Store ... 695
Externe Schlüsselspeicher ... 712
Konzepte für externe Schlüsselspeicher ... 718
Funktionsweise externer Schlüsselspeicher ... 727
Steuern Sie den Zugriff auf Ihren externen Schlüsselspeicher ... 728
Wählen Sie eine Proxy-Konnektivitätsoption ... 733
Erstellen Sie einen externen Schlüsselspeicher ... 746
Eigenschaften des externen Schlüsselspeichers bearbeiten ... 760
Externe Schlüsselspeicher anzeigen ... 768
Überwachen Sie externe Schlüsselspeicher ... 774
Externe Schlüsselspeicher Connect und trennen ... 787
Löschen Sie einen externen Schlüsselspeicher ... 798
Fehlerbehebung bei externen Schlüsselspeichern ... 800
Sicherheit ... 829
Datenschutz ... 830
Schutz von Schlüsselmaterial ... 830

Datenverschlüsselung ... 832
Datenschutz zwischen Netzwerken ... 833
Identity and Access Management ... 834
AWS verwaltete Richtlinien ... 835
Service-verknüpfte Rollen ... 838
Protokollierung und Überwachung ... 845
Compliance-Validierung ... 847
Compliance-und Sicherheitsdokumente ... 847
Weitere Informationen ... 848
Ausfallsicherheit ... 848
Regionale Isolierung ... 849
Design mit mehreren Mandanten ... 849
Bewährte Methoden zur Resilienz in AWS KMS ... 850
Sicherheit der Infrastruktur ... 850
Isolierung auf physischen Hosts ... 852
Kontingente ... 853
Ressourcenkontingente ... 853
AWS KMS keys: 100.000 ... 854
Aliasse pro KMS-Schlüssel: 50 ... 854
Erteilungen pro KMS-Schlüssel: 50 000 ... 855
Ressourcenkontingent für benutzerdefinierte Schlüsselspeicher: 10 ... 855
Rotation auf Anfrage: 10 ... 856
Anforderungskontingente ... 856
Fordern Sie Kontingente für jeden AWS KMS API-Vorgang an ... 857
Anwenden von Anforderungskontingenten ... 864
Gemeinsame Kontingente für kryptografische Operationen ... 865
API-Anforderungen in Ihrem Namen ... 866
Kontoübergreifende Anforderungen ... 867
Anforderungskontingente für benutzerdefinierte Schlüsselspeicher ... 867
Drosselung von -Anforderungen ... 869
Codebeispiele ... 872
Grundlagen ... 876
Hallo AWS KMS ... 877
Erlernen der Grundlagen ... 880
Aktionen ... 954
Szenarien ... 1106

Arbeiten Sie mit Tabellenverschlüsselung ... 1107
Kryptografische Bescheinigung für Nitro-Enklaven AWS ... 1110
Wie ruft man nach einer AWS KMS APIs Nitro-Enklave ... 1112
Überwachung von Anfragen für Nitro-Enklaven ... 1113
Entschlüsseln (für eine Enklave) ... 1113
GenerateDataKey (für eine Enklave) ... 1114
GenerateDataKeyPair (für eine Enklave) ... 1116
GenerateRandom (für eine Enklave) ... 1117
Dienste verschlüsseln AWS ... 1119
Amazon Elastic Block Store (Amazon EBS) ... 1120
Amazon-EBS-Verschlüsselung ... 1120
Verwenden von KMS-Schlüsseln und Datenschlüsseln ... 1121
Amazon-EBS-Verschlüsselungskontext ... 1121
Erkennen von Amazon-EBS-Fehlern ... 1122
Verwenden AWS CloudFormation , um verschlüsselte Amazon EBS-Volumes zu erstellen 1123
Amazon EMR ... 1123
Verschlüsseln von Daten auf dem EMR-Dateisystem (EMRFS) ... 1124
Verschlüsseln von Daten auf den Speicher-Volumes von Cluster-Knoten ... 1127
Verschlüsselungskontext ... 1128
Amazon Redshift ... 1129
Amazon-Redshift-Verschlüsselung ... 1130
Verschlüsselungskontext ... 1131
Referenz ... 1132
Referenz zum Schlüsselstatus ... 1133
Schlüsselstatus und KMS-Schlüsseltypen ... 1133
Schlüsselstatus-Tabelle ... 1134
Schlüsseltypreferenz ... 1142
Schlüsseltyp-Tabelle ... 1143
Tabelle „Spezielle Funktionen“ ... 1151
Referenz zu wichtigen Spezifikationen ... 1159
Schlüsselspezifikation SYMMMETRIC_DEFAULT ... 1161
RSA-Schlüsselspezifikationen ... 1163
Elliptic Curve(EC)-Schlüsselspezifikationen ... 1167
Schlüsselspezifikationen für HMAC-KMS-Schlüssel ... 1169
ML-DSA — wichtige Spezifikationen ... 1170
SM2 wichtige Spezifikation (nur Regionen China) ... 1171

Berechtigungsreferenz 1172
Beschreibungen der Spalten 1222
AWS KMS interne Operationen 1224
Domänen und Domänenstatus 1225
Interne Kommunikationssicherheit 1229
Replikationsprozess für Schlüssel mit mehreren Regionen 1232
Beständigkeitsschutz 1233
Dokumentverlauf 1235
Neueste Aktualisierungen 1235
Frühere Aktualisierungen 1243
..... mccxlviii

AWS Key Management Service

AWS Key Management Service (AWS KMS) ist ein AWS verwalteter Dienst, mit dem Sie die Verschlüsselungsschlüssel, die zur Verschlüsselung Ihrer Daten verwendet werden, auf einfache Weise erstellen und kontrollieren können. Die AWS KMS keys , in denen Sie erstellen, AWS KMS sind durch FIPS 140-3 Security Level 3-validierte Hardware-Sicherheitsmodule (HSM) geschützt. Sie werden niemals unverschlüsselt verlassen. AWS KMS Um Ihre KMS-Schlüssel zu verwenden oder zu verwalten, interagieren Sie mit AWS KMS.

Warum verwenden AWS KMS?

Wenn Sie Daten verschlüsseln, müssen Sie Ihren Verschlüsselungsschlüssel schützen. Wenn Sie Ihren Schlüssel verschlüsseln, müssen Sie seinen Verschlüsselungsschlüssel schützen. Schließlich müssen Sie den Verschlüsselungsschlüssel der höchsten Ebene (auch Stammschlüssel genannt) in der Hierarchie schützen, der Ihre Daten schützt. Hier AWS KMS kommt es ins Spiel.

AWS KMS schützt Ihre Root-Schlüssel. KMS-Schlüssel werden vollständig innerhalb erstellt, verwaltet, verwendet und gelöscht AWS KMS. Sie verlassen den Dienst niemals unverschlüsselt. Um Ihre KMS-Schlüssel zu verwenden oder zu verwalten, rufen Sie an AWS KMS.

Darüber hinaus können Sie wichtige Richtlinien erstellen und verwalten AWS KMS, um sicherzustellen, dass nur vertrauenswürdige Benutzer Zugriff auf KMS-Schlüssel haben.

AWS KMS in AWS-Regionen

Welche davon AWS KMS unterstützt werden, sind unter AWS Key Management Service Endpunkte und Kontingente aufgeführt. AWS-Regionen Wenn eine AWS KMS Funktion in einer, AWS-Region die dies unterstützt, nicht AWS KMS unterstützt wird, werden die regionalen Unterschiede im Thema zu der Funktion beschrieben.

AWS KMS Preisgestaltung

Wie bei anderen AWS Produkten sind für die Nutzung AWS KMS keine Verträge oder Mindestkäufe erforderlich. Weitere Informationen zur AWS KMS Preisgestaltung finden Sie unter AWS Key Management Service Preisgestaltung.

AWS KMS Service Level Agreement

AWS Key Management Service wird durch ein Service Level Agreement unterstützt, das unsere Richtlinien zur Serviceverfügbarkeit festlegt.

Zugreifen AWS Key Management Service

Sie können auf folgende AWS KMS Weise damit arbeiten:

Themen

- AWS Management Console
- AWS Command Line Interface
- AWS KMS REST API
- AWS SDKs
- Verwenden dieses Dienstes mit einem AWS SDK
- AWS Encryption SDK
- AWS KMS letztendliche Konsistenz
- Verwendung von hybridem Post-Quantum-TLS mit AWS KMS
- Stellen Sie eine Connect AWS KMS über einen VPC-Endpunkt her
- Unterstützung für Dual-Stack-Endgeräte

AWS Management Console

Die Konsole ist eine webbasierte Benutzeroberfläche für Verwaltung AWS KMS und AWS Ressourcen. Wenn Sie sich für eine angemeldet haben AWS-Konto, können Sie auf die AWS KMS Konsole zugreifen, indem Sie sich bei der anmelden AWS Management Console und auf AWS KMS der AWS Management Console Startseite auswählen.

Für die Verwendung der AWS KMS Konsole sind Berechtigungen erforderlich

Um mit der AWS KMS Konsole arbeiten zu können, müssen Benutzer über Mindestberechtigungen verfügen, die es ihnen ermöglichen, mit den AWS KMS Ressourcen in ihrer Konsole zu arbeiten AWS-Konto. Zusätzlich zu diesen AWS KMS -Berechtigungen müssen Benutzer auch über die Berechtigungen zum Auflisten von IAM-Benutzern und IAM-Rollen verfügen. Wenn Sie eine IAM-Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die AWS KMS -Konsole nicht wie vorgesehen für Benutzer mit dieser IAM-Richtlinie.

Die Mindestberechtigungen, um einem Benutzer reinen Lesezugriff auf die AWS KMS -Konsole zu gewähren, finden Sie unter Erlauben Sie einem Benutzer, KMS-Schlüssel in der AWS KMS Konsole anzuzeigen.

Damit Benutzer bei der Erstellung und Verwaltung von KMS-Schlüsseln mit der AWS KMS Konsole arbeiten können, fügen Sie dem Benutzer die AWSKeyManagementServicePowerUserverwaltete Richtlinie an, wie unter beschriebenAWS verwaltete Richtlinien für AWS Key Management Service.

Sie müssen Benutzern, die mit der AWS KMS API arbeiten, keine Mindestberechtigungen für die Konsole über AWS SDKsAWS Command Line Interface, oder gewähren AWS -Tools für PowerShell. Sie müssen diesen Benutzern jedoch die Berechtigung zur Verwendung der API erteilen. Weitere Informationen finden Sie unter Berechtigungsreferenz.

AWS Command Line Interface

Sie können die AWS CLI Tools verwenden, um Befehle auszugeben oder Skripts an der Befehlszeile Ihres Systems zu erstellen, um AWS (einschließlich AWS KMS) Aufgaben auszuführen.

Weitere Informationen zur Verwendung von AWS KMS finden Sie in der AWS CLI Befehlsreferenz AWS CLI

AWS KMS REST API

Die Architektur von AWS KMS ist so konzipiert, dass sie programmiersprachenneutral ist. Das REST API ist eine HTTP-Schnittstelle zu. AWS KMS Mit dem verwenden Sie Standard-HTTP-AnfragenREST API, um Schlüssel zu erstellen, abzurufen und zu löschen.

Weitere Informationen zur Verwendung von finden Sie in der AWS Key Management Service API-Referenz AWS KMSREST API

AWS SDKs

AWS stellt SDKs (Software Development Kits) bereit, die aus Bibliotheken und Beispielcode für gängige Programmiersprachen und Plattformen (Java JavaScript, C, Python usw.) bestehen. AWS SDKs Sie bieten eine bequeme Möglichkeit, programmatischen Zugriff auf AWS KMS und AWS zu erstellen. AWS KMS ist ein REST Service. Sie können Anfragen an AWS KMS die AWS SDK-Bibliotheken senden, die die zugrunde liegenden Daten AWS KMS REST API zusammenfassen und

Ihre Programmieraufgaben vereinfachen. Informationen zu den AWS SDKs, einschließlich deren Download und Installation, finden Sie unter Tools, auf denen Sie aufbauen können AWS.

Das Codebeispiele für die AWS KMS Verwendung AWS SDKs bietet einen guten Ausgangspunkt für die AWS KMS Verwendung von AWS SDKs.

Verwenden dieses Dienstes mit einem AWS SDK

AWS Software Development Kits (SDKs) sind für viele gängige Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele
AWS CLI	AWS CLI Code-Beispiele
AWS SDK für Go	AWS SDK für Go Code-Beispiele
AWS SDK für Java	AWS SDK für Java Code-Beispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Code-Beispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Code-Beispiele
AWS SDK für .NET	AWS SDK für .NET Code-Beispiele
AWS SDK für PHP	AWS SDK für PHP Code-Beispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Code-Beispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Code-Beispiele
AWS SDK für Ruby	AWS SDK für Ruby Code-Beispiele
AWS SDK für Rust	AWS SDK für Rust Code-Beispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Code-Beispiele
AWS SDK für Swift	AWS SDK für Swift Code-Beispiele

> **Beispiel für die Verfügbarkeit**
>
> Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Feedback geben auswählen.

AWS Encryption SDK

Das AWS Encryption SDK ist ein Tool zur Implementierung der clientseitigen Verschlüsselung in Ihrer Anwendung. Es bietet keinen vollständigen Zugriff auf KMS, sondern integriert sich in dieses oder kann als eigenständiges SDK verwendet werden AWS KMS, ohne auf KMS-Schlüssel zu verweisen. Bibliotheken sind für Java, C JavaScript, Python und andere Programmiersprachen verfügbar.

Weitere Informationen finden Sie im AWS Encryption SDK -Entwicklerhandbuch.

AWS KMS key Richtlinien und IAM-Richtlinien

AWS KMS letztendliche Konsistenz

Die AWS KMS API folgt aufgrund der verteilten Struktur des Systems einem Modell der endgültigen Konsistenz. Daher sind Änderungen an AWS KMS Ressourcen möglicherweise nicht sofort für die nachfolgenden Befehle sichtbar, die Sie ausführen.

Wenn Sie AWS KMS API-Aufrufe ausführen, kann es zu einer kurzen Verzögerung kommen, bis die Änderung vollständig verfügbar ist AWS KMS. In der Regel dauert es weniger als ein paar Sekunden, bis sich die Änderung im gesamten System verbreitet, in einigen Fällen kann es jedoch mehrere Minuten dauern. Während dieser Zeit können unerwartete Fehler auftreten, wie z. B. eine `NotFoundException` oder eine `InvalidStateException`. AWS KMS Könnte beispielsweise a zurückgeben, `NotFoundException` wenn Sie `GetParametersForImport` unmittelbar nach dem Anruf anrufen`CreateKey`.

Wir empfehlen Ihnen, auf Ihren AWS KMS Clients eine Wiederholungsstrategie zu konfigurieren, sodass Vorgänge nach einer kurzen Wartezeit automatisch wiederholt werden. Weitere Informationen finden Sie unter Verhalten bei Wiederholungsversuchen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Für API-Aufrufe im Zusammenhang mit Zuschüssen können Sie ein Grant-Token verwenden, um mögliche Verzögerungen zu vermeiden, und die in einem Grant enthaltenen Berechtigungen sofort nutzen. Weitere Informationen finden Sie unter Letztendliche Konsistenz (für Erteilungen).

Verwendung von hybridem Post-Quantum-TLS mit AWS KMS

AWS Key Management Service (AWS KMS) unterstützt eine hybride Option für den Schlüsselaustausch nach dem Quantenzugriff für das Netzwerkverschlüsselungsprotokoll Transport Layer Security (TLS). Sie können diese TLS-Option verwenden, wenn Sie eine Verbindung zu AWS KMS -API-Endpunkten herstellen. Diese optionalen Hybrid-Post-Quantum-Schlüsselaustauschfunktionen sind mindestens so sicher wie die heute verwendete TLS-Verschlüsselung und bieten wahrscheinlich zusätzliche langfristige Sicherheitsvorteile. Sie wirken sich jedoch auf Latenz und Durchsatz im Vergleich zu den klassischen Schlüsselaustauschprotokollen aus, die heute verwendet werden.

Die Daten, die Sie an AWS Key Management Service (AWS KMS) senden, sind bei der Übertragung durch die Verschlüsselung geschützt, die über eine Transport Layer Security (TLS) -Verbindung bereitgestellt wird. Die klassischen Cipher-Suites, die TLS-Sitzungen AWS KMS unterstützen, machen Brute-Force-Angriffe auf die Schlüsselaustauschmechanismen mit der aktuellen Technologie undurchführbar. Wenn jedoch in Zukunft das große Quantencomputing praktikabel wird, werden die klassischen Cipher-Suites, die in TLS-Schlüsselaustauschmechanismen verwendet werden, für diese Angriffe anfällig sein. Wenn Sie Anwendungen entwickeln, die auf der langfristigen Vertraulichkeit von Daten basieren, die über eine TLS-Verbindung übertragen werden, sollten Sie einen Plan zur Umstellung auf Post-Quanten-Kryptografie in Betracht ziehen, bevor große Quantencomputer für den Einsatz verfügbar sind. AWS arbeitet daran, sich auf diese future vorzubereiten, und wir möchten, dass auch Sie gut vorbereitet sind.

Um heute verschlüsselte Daten vor möglichen future Angriffen zu schützen, beteiligt AWS sich die Kryptografie-Community an der Entwicklung quantenresistenter oder Post-Quanten-Algorithmen. Wir haben hybride Chiffrier-Suites für den Postquanten-Schlüsselaustausch implementiert AWS KMS , die klassische und Post-Quantum-Elemente kombinieren, um sicherzustellen, dass Ihre TLS-Verbindung mindestens so stark ist wie bei klassischen Cipher Suites.

Diese hybriden Verschlüsselungssammlungen sind in den meisten Fällen für den Einsatz in Ihren Produktions-Workloads verfügbar. AWS-Regionen Da sich die Leistungsmerkmale und Bandbreitenanforderungen hybrider Verschlüsselungssammlungen jedoch von denen klassischer Schlüsselaustauschmechanismen unterscheiden, empfehlen wir, sie bei Ihren AWS KMS API-Aufrufen unter anderen Bedingungen zu testen.

Feedback

Wie immer freuen wir uns über Ihr Feedback und Ihre Teilnahme an unseren Open Source-Repositories. Wir möchten besonders erfahren, wie Ihre Infrastruktur mit dieser neuen Variante des TLS-Datenverkehrs interagiert.

- Um Feedback zu diesem Thema zu geben, verwenden Sie den Link Feedback in der oberen rechten Ecke dieser Seite.
- Wir entwickeln diese hybriden Verschlüsselungssammlungen als Open Source im Repository am. s2n-tls GitHub Um Feedback zur Benutzerfreundlichkeit der Cipher-Suites zu geben oder neue Testbedingungen oder Ergebnisse zu teilen, erstellen Sie eine Anforderung im s2n-tls-Repository.
- Wir schreiben Codebeispiele für die Verwendung von hybridem Post-Quantum-TLS AWS KMS im Repository. aws-kms-pq-tls-example GitHub Um Fragen zu stellen oder Ideen zur Konfiguration Ihres HTTP-Clients oder AWS KMS -Clients für die Verwendung der Hybrid-Cipher Suites auszutauschen, erstellen Sie ein Problem im Repository. aws-kms-pq-tls-example

Unterstützt AWS-Regionen

Post-Quantum-TLS für AWS KMS ist in allen AWS KMS unterstützten AWS-Regionen Versionen verfügbar.

Eine Liste der jeweiligen AWS KMS AWS-Region Endpunkte finden Sie unter AWS Key Management Service Endpunkte und Kontingente in der. Allgemeine Amazon Web Services-Referenz Weitere Informationen über FIPS-Endpunkte finden Sie unter FIPS-Endpunkte in der Allgemeine Amazon Web Services-Referenz.

Über den Hybrid-Post-Quantum-Schlüsselaustausch in TLS

AWS KMS unterstützt hybride Chiffrier-Suites nach dem Quantum-Schlüsselaustausch. Sie können Common Runtime AWS SDK for Java 2.x und AWS Common Runtime auf Linux-Systemen verwenden, um einen HTTP-Client zu konfigurieren, der diese Verschlüsselungssammlungen verwendet. Wenn Sie dann mit Ihrem HTTP-Client eine Verbindung zu einem AWS KMS Endpunkt herstellen, werden die Hybrid-Verschlüsselungssammlungen verwendet.

Dieser HTTP-Client verwendet s2n-tls, eine Open-Source-Implementierung des TLS-Protokolls. Die Hybrid-Cipher-Suites , die s2n-tls verwendet, sind nur für den Schlüsselaustausch implementiert, nicht für die direkte Datenverschlüsselung. Während des Schlüsselaustauschs berechnen Client und Server den Schlüssel, den sie verwenden, um die Daten auf der Leitung zu verschlüsseln und zu entschlüsseln.

s2n-tlsBei den verwendeten Algorithmen handelt es sich um einen Hybrid, der Elliptic Curve Diffie-Hellman (ECDH), einen klassischen Schlüsselaustauschalgorithmus, der heute in TLS verwendet wird, mit Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) kombiniert, einem Algorithmus zur Verschlüsselung und Schlüsselerstellung mit öffentlichen Schlüsseln, den das National Institute for Standards and Technology (NIST) als seinen ersten Standard-Post-Quantum-Schlüsselvereinbarungsalgorithmus bezeichnet hat. Dieser Hybrid verwendet jeden der Algorithmen unabhängig, um einen Schlüssel zu generieren. Dann kombiniert es die beiden Schlüssel kryptografisch. Mit s2n-tls können Sie einen HTTP-Client so konfigurieren, dass er Post-Quantum-TLS bevorzugt, wodurch ECDH mit ML-KEM an erster Stelle in der Präferenzliste steht. Klassische Schlüsselaustauschalgorithmen sind in der Einstellungsliste enthalten, um die Kompatibilität sicherzustellen, aber sie sind in der Präferenzreihenfolge niedriger.

Verwendung von hybridem Post-Quantum-TLS mit AWS KMS

Sie können hybrides Post-Quantum-TLS für Ihre Anrufe bei verwenden. AWS KMS Beachten Sie beim Einrichten der HTTP-Client-Testumgebung die folgenden Informationen:

Verschlüsselung während der Übertragung

Die Hybrid-Cipher-Suites in s2n-tls werden nur für die Verschlüsselung während der Übertragung verwendet. Sie schützen Ihre Daten auf dem Weg von Ihrem Client zum AWS KMS Endpunkt. AWS KMS verwendet diese Cipher Suites nicht zum Verschlüsseln von Daten unter. AWS KMS keys

Stattdessen verwendet es bei der AWS KMS Verschlüsselung Ihrer Daten unter KMS-Schlüsseln symmetrische Kryptografie mit 256-Bit-Schlüsseln und den AES-GCM-Algorithmus (Advanced Encryption Standard in Galois Counter Mode), der bereits quantenresistent ist. Theoretische zukünftige, groß angelegte Quantenrechnungsangriffe auf Verschlüsselungstexte, die unter 256-Bit-AES-GCM-Schlüsseln erstellt wurden, reduzieren die effektive Sicherheit des Schlüssels auf 128-Bits. Diese Sicherheitsstufe reicht aus, um Brute-Force-Angriffe auf Chiffretexte unmöglich zu machen. AWS KMS

Unterstützte Systeme

Die Verwendung der Hybrid-Cipher-Suites in s2n-tls wird derzeit nur auf Linux-Systemen unterstützt. Darüber hinaus werden diese Verschlüsselungssammlungen nur unterstützt, wenn sie die AWS Common Runtime unterstützen SDKs , wie z. B. die. AWS SDK for Java 2.x Ein Beispiel finden Sie unter Konfigurieren Sie hybrides Post-Quantum-TLS.

AWS KMS Endpunkte

Wenn Sie die hybriden Cipher Suites verwenden, verwenden Sie den Standardendpunkt. AWS KMS AWS KMS unterstützt kein hybrides Post-Quantum-TLS für FIPS 140-3-validierte Endpunkte.

Wenn Sie einen HTTP-Client so konfigurieren, dass er Post-Quantum-TLS-Verbindungen mit s2n-tls bevorzugt, stehen die Post-Quantum-Ciphers in der Liste der bevorzugten Ciphers an erster Stelle. Die Voreinstellungsliste enthält jedoch die klassischen, nicht-hybriden Verschlüsselungen, die in der Präferenzreihenfolge aus Gründen der Kompatibilität niedriger sind. Wenn Sie einen HTTP-Client so konfigurieren, dass er Post-Quantum-TLS mit einem AWS KMS FIPS 140-3-validierten Endpunkt bevorzugt, s2n-tls handelt es sich um eine klassische Chiffre für den Schlüsselaustausch ohne Hybrid.

Eine Liste der jeweiligen Endpunkte finden Sie unter AWS KMS Endpunkte und Kontingente in der. AWS-RegionAWS Key Management ServiceAllgemeine Amazon Web Services-Referenz Weitere Informationen über FIPS-Endpunkte finden Sie unter FIPS-Endpunkte in der Allgemeine Amazon Web Services-Referenz.

Konfigurieren Sie hybrides Post-Quantum-TLS

Fügen Sie in diesem Verfahren eine Maven-Abhängigkeit für den AWS Common Runtime HTTP Client hinzu. Anschließend konfigurieren Sie einen HTTP-Client, der Post-Quantum-TLS bevorzugt. Erstellen Sie dann einen AWS KMS Client, der den HTTP-Client verwendet.

Eine vollständige Liste von Arbeitsbeispielen zum Konfigurieren und Verwenden von hybriden Post-Quantum-TLS mit AWS KMS, finden Sie im aws-kms-pq-tls-example-Repository.

> **Note**
>
> Der AWS Common Runtime HTTP Client, der als Vorversion verfügbar war, wurde im Februar 2023 allgemein verfügbar. In dieser Version werden die `tlsCipherPreference`-Klasse und der `tlsCipherPreference()`-Methodenparameter durch den `postQuantumTlsEnabled()`-Methodenparameter ersetzt. Wenn Sie dieses Beispiel in der Vorversion verwendet haben, müssen Sie Ihren Code aktualisieren.

1. Fügen Sie den AWS Common Runtime-Client zu Ihren Maven-Abhängigkeiten hinzu. Wir empfehlen, die neueste verfügbare Version zu verwenden.

   Diese Anweisung fügt beispielsweise eine Version `2.30.22` des AWS Common Runtime-Clients zu Ihren Maven-Abhängigkeiten hinzu.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>aws-crt-client</artifactId>
    <version>2.30.22</version>
</dependency>
```

2. Um die hybriden Post-Quantum-Cipher-Suites zu aktivieren, fügen Sie sie AWS SDK for Java 2.x zu Ihrem Projekt hinzu und initialisieren Sie sie. Aktivieren Sie dann die hybriden Post-Quantum-Cipher-Suites auf Ihrem HTTP-Client, wie im folgenden Beispiel gezeigt.

   Dieser Code verwendet den `postQuantumTlsEnabled()` Methodenparameter, um einen AWS gemeinsamen Runtime-HTTP-Client zu konfigurieren, der die empfohlene hybride Post-Quantum-Verschlüsselungssuite ECDH with bevorzugt. ML-KEM Anschließend verwendet er den konfigurierten HTTP-Client, um eine Instanz des asynchronen Clients zu erstellen,. AWS KMS `KmsAsyncClient` Nachdem dieser Code abgeschlossen ist, verwenden alle AWS KMS -API-Anfragen auf der `KmsAsyncClient`-Instance hybrides Post-Quantum-TLS.

```
// Configure HTTP client
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
          .postQuantumTlsEnabled(true)
          .build();

// Create the AWS KMS async client
KmsAsyncClient kmsAsync = KmsAsyncClient.builder()
         .httpClient(awsCrtHttpClient)
         .build();
```

3. Testen Sie Ihre AWS KMS Anrufe mit hybridem Post-Quantum-TLS.

   Wenn Sie AWS KMS API-Operationen auf dem konfigurierten AWS KMS Client aufrufen, werden Ihre Aufrufe mithilfe von Hybrid-Post-Quantum-TLS an den AWS KMS Endpunkt übertragen. Um Ihre Konfiguration zu testen, rufen Sie eine AWS KMS API auf, z. B. `ListKeys`

```
ListKeysReponse keys = kmsAsync.listKeys().get();
```

Testen Sie Ihre hybride Post-Quantum-TLS-Konfiguration

Erwägen Sie, die folgenden Tests mit Hybrid-Verschlüsselungssammlungen für Ihre aufrufenden Anwendungen durchzuführen. AWS KMS

- Führen Sie Belastungstests und Benchmarks aus. Die Hybrid-Cipher-Suites funktionieren anders als herkömmliche Schlüsselaustauschalgorithmen. Möglicherweise müssen Sie Ihre Verbindungszeitüberschreitungen anpassen, um längere Handshake-Zeiten zu ermöglichen. Wenn Sie innerhalb einer AWS Lambda Funktion arbeiten, verlängern Sie die Einstellung für das Ausführungstimeout.
- Versuchen Sie, eine Verbindung von verschiedenen Standorten herzustellen. Abhängig vom Netzwerkpfad Ihrer Anforderung können Sie feststellen, dass Zwischenhosts, Proxys oder Firewalls mit Deep Packet Inspection (DPI) die Anforderung blockieren. Dies kann auf die Verwendung der neuen Cipher Suites im ClientHelloTeil des TLS-Handshakes oder auf die größeren Schlüsselaustauschnachrichten zurückzuführen sein. Wenn Sie Probleme bei der Behebung dieser Probleme haben, arbeiten Sie mit Ihrem Sicherheitsteam oder IT-Administratoren zusammen, um die entsprechende Konfiguration zu aktualisieren und die Blockierung der neuen TLS-Cipher-Suites aufzuheben.

Weitere Informationen über Post-Quantum-TLS finden Sie unter AWS KMS

Weitere Informationen zur Verwendung von Hybrid-Post-Quantum-TLS finden Sie in AWS KMS den folgenden Ressourcen.

- Weitere Informationen zur Post-Quanten-Kryptografie AWS, einschließlich Links zu Blogbeiträgen und Forschungsarbeiten, finden Sie unter Post-Quantum-Kryptografie.
- Weitere Informationen zu s2n-tls finden Sie unter Introducing s2n-tls, a New Open Source TLS Implementation und Using s2n-tls.
- Informationen zum AWS Common Runtime HTTP Client finden Sie unter Konfiguration des AWS CRT-basierten HTTP-Clients im Entwicklerhandbuch.AWS SDK for Java 2.x
- Weitere Informationen zum Post-Quantum-Kryptographie-Projekt am National Institute for Standards and Technology (NIST) finden Sie unter Post-Quantum-Kryptographie.
- Informationen zur Standardisierung der Post-Quantum-Kryptografie durch NIST finden Sie unter Standardisierung der Post-Quantum-Kryptografie.

Stellen Sie eine Connect AWS KMS über einen VPC-Endpunkt her

Sie können AWS KMS über einen privaten Schnittstellen-Endpunkt in Ihrer Virtual Private Cloud (VPC) eine direkte Verbindung herstellen. Wenn Sie einen VPC-Schnittstellen-Endpunkt verwenden,

AWS KMS erfolgt die Kommunikation zwischen Ihrer VPC und dem vollständig innerhalb des AWS Netzwerks.

AWS KMS unterstützt Amazon Virtual Private Cloud (Amazon VPC) -Endpunkte, die von betrieben werden. AWS PrivateLink Jeder VPC-Endpunkt wird durch eine oder mehrere Elastic Network Interfaces (ENIs) mit privaten IP-Adressen in Ihren VPC-Subnetzen repräsentiert.

Der VPC-Endpunkt der Schnittstelle verbindet Ihre VPC direkt, AWS KMS ohne dass ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder AWS Direct Connect eine Verbindung erforderlich ist. Die Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen für die Kommunikation. AWS KMS

Regionen

AWS KMS unterstützt VPC-Endpunkte und VPC-Endpunktrichtlinien insgesamt AWS-Regionen , was AWS KMSunterstützt wird.

Überlegungen zu AWS KMS VPC-Endpunkten

Bevor Sie einen Schnittstellen-VPC-Endpunkt für einrichten AWS KMS, lesen Sie das Thema Eigenschaften und Einschränkungen von Schnittstellenendpunkten im AWS PrivateLink Handbuch.

AWS KMS Die Unterstützung für einen VPC-Endpunkt umfasst Folgendes.

- Sie können den VPC-Endpunkt verwenden, um alle AWS KMS -API-Vorgänge in Ihrer VPC aufzurufen.
- Sie können einen VPC-Schnittstellen-Endpunkt erstellen, der eine Verbindung zu einem AWS KMS Regionsendpunkt oder einem AWS KMS FIPS-Endpunkt herstellt.
- Sie können AWS CloudTrail Protokolle verwenden, um Ihre Verwendung von KMS-Schlüsseln über den VPC-Endpunkt zu überprüfen. Details hierzu finden Sie unter Protokollierung von AWS KMS Anfragen, die einen VPC-Endpunkt verwenden.

Themen

- Erstellen Sie einen VPC-Endpunkt für AWS KMS
- Stellen Sie eine Connect zu einem AWS KMS VPC-Endpunkt her
- Verwenden Sie VPC-Endpunkte, um den Zugriff auf Ressourcen zu kontrollieren AWS KMS
- Protokollierung von AWS KMS Anfragen, die einen VPC-Endpunkt verwenden

Erstellen Sie einen VPC-Endpunkt für AWS KMS

Sie können AWS KMS mithilfe der Amazon VPC-Konsole oder der Amazon VPC-API einen VPC-Endpunkt für erstellen. Folgen Sie den Verfahren zum Erstellen eines Schnittstellenendpunkts mit einem der folgenden Werte.

- Verwenden Sie den folgenden Dienstnamen AWS KMS, um einen VPC-Endpunkt für zu erstellen:

  ```
  com.amazonaws.region.kms
  ```

  In der Region USA West (Oregon) (`us-west-2`) würde der Servicename wie folgt lauten:

  ```
  com.amazonaws.us-west-2.kms
  ```

- Verwenden Sie den folgenden Servicenamen, um einen VPC-Endpunkt zu erstellen, der eine Verbindung zu einem AWS KMS -FIPS-Endpunkt herstellt:

  ```
  com.amazonaws.region.kms-fips
  ```

  In der Region USA West (Oregon) (`us-west-2`) würde der Servicename wie folgt lauten:

  ```
  com.amazonaws.us-west-2.kms-fips
  ```

Um die Verwendung des VPC-Endpunkts zu vereinfachen, können Sie einen privaten DNS-Namen für Ihren VPC-Endpunkt aktivieren. Wenn Sie die Option Enable Private DNS Name (Privaten DNS-Namen aktivieren) auswählen, wird der standardmäßige AWS KMS -DNS-Hostname in Ihren VPC-Endpunkt aufgelöst. `https://kms.us-west-2.amazonaws.com` würde beispielsweise in einen VPC-Endpunkt aufgelöst, der mit dem Servicenamen `com.amazonaws.us-west-2.kms` verbunden ist.

Diese Option vereinfacht die Verwendung des VPC-Endpunkts. Das AWS SDKs und AWS CLI verwendet standardmäßig den AWS KMS Standard-DNS-Hostnamen, sodass Sie die VPC-Endpunkt-URL in Anwendungen und Befehlen nicht angeben müssen.

Weitere Informationen finden Sie unter Zugriff auf einen Service über einen Schnittstellenendpunkt im AWS PrivateLink -Leitfaden.

Stellen Sie eine Connect zu einem AWS KMS VPC-Endpunkt her

Sie können AWS KMS über den VPC-Endpunkt eine Verbindung herstellen, indem Sie ein AWS SDK AWS CLI, das oder AWS -Tools für PowerShell verwenden. Um den VPC-Endpunkt anzugeben, verwenden Sie seinen DNS-Namen.

Dieser list-keys-Befehl verwendet zur Angabe des VPC-Endpunkts z. B. den Parameter `endpoint-url`. Wenn Sie einen solchen Befehl verwenden möchten, ersetzen Sie die Beispiels-ID des VPC-Endpunkts durch eine ID in Ihrem Konto.

```
$ aws kms list-keys --endpoint-url https://vpce-1234abcdf5678c90a-09p7654s-us-east-1a.ec2.us-east-1.vpce.amazonaws.com
```

Erforderliche Berechtigungen

Damit eine AWS KMS Anfrage, die einen VPC-Endpunkt verwendet, erfolgreich ist, benötigt der Principal Berechtigungen aus zwei Quellen:

- Eine -Schlüsselrichtlinie, IAM-Richtlinie oder Erteilung muss dem Prinzipal die Berechtigung erteilen, die Operation für die Ressource (KMS-Schlüssel oder Alias) aufzurufen.
- Eine VPC-Endpunktrichtlinie muss dem Prinzipal die Berechtigung erteilen, den Endpunkt für die Anforderung zu verwenden.

So kann eine Schlüsselrichtlinie einem Prinzipal die Berechtigung zum Aufrufen von Decrypt auf einen bestimmten KMS-Schlüssel erteilen. Allerdings könnte die VPC-Endpunktrichtlinie diesem Prinzipal nicht erlauben, `Decrypt` auf diesem KMS-Schlüssel mithilfe des Endpunkts aufzurufen.

Oder eine VPC-Endpunktrichtlinie könnte es einem Prinzipal ermöglichen, den Endpunkt zu verwenden, um bestimmte KMS-Schlüssel DisableKeyaufzurufen. Wenn der Prinzipal jedoch nicht über diese Berechtigungen aus einer Schlüsselrichtlinie, IAM-Richtlinie oder Erteilung verfügt, schlägt die Anforderung fehl.

Sie können beim Erstellen des Endpunkts eine VPC-Endpunktrichtlinie erstellen und die VPC-Endpunktrichtlinie jederzeit ändern. Verwenden Sie die VPC-Managementkonsole oder die ModifyVpcEndpointOperationen CreateVpcEndpointoder. Sie können eine VPC-Endpunktrichtlinie auch mithilfe einer AWS CloudFormation Vorlage erstellen und ändern. Hilfe zur Verwendung der VPC-Managementkonsole finden Sie unter Erstellen eines Schnittstellenendpunkts und Ändern eines Schnittstellenendpunkts im AWS PrivateLink -Leitfaden.

Private Hostnamen

Wenn Sie beim Erstellen Ihres VPC-Endpunkts private Hostnamen aktiviert waren, müssen Sie die URL des Endpunkts in Ihren CLI-Befehlen oder in Ihrer Anwendungskonfiguration angeben. In diesem Fall wird der standardmäßige AWS KMS -DNS-Hostname in Ihren VPC-Endpunkt aufgelöst. Der AWS CLI und SDKs verwenden standardmäßig diesen Hostnamen, sodass Sie den VPC-Endpunkt verwenden können, um eine Verbindung zu einem AWS KMS regionalen Endpunkt herzustellen, ohne etwas an Ihren Skripten und Anwendungen zu ändern.

Zur Verwendung privater Hostnamen müssen die Attribute `enableDnsHostnames` und `enableDnsSupport` Ihrer VPC auf `true` eingestellt sein. Verwenden Sie den Vorgang, um diese Attribute festzulegen. ModifyVpcAttribute Details dazu finden Sie unter Anzeigen und Aktualisieren von DNS-Attributen für Ihre VPC im Amazon-VPC-Benutzerhandbuch.

Verwenden Sie VPC-Endpunkte, um den Zugriff auf Ressourcen zu kontrollieren AWS KMS

Sie können den Zugriff auf AWS KMS Ressourcen und Vorgänge steuern, wenn die Anfrage von einer VPC kommt oder einen VPC-Endpunkt verwendet. Verwenden Sie dazu einen der folgenden globalen Bedingungsschlüssel in einer Schlüsselrichtlinie oder IAM-Richtlinie.

- Verwenden Sie den `aws:sourceVpce`-Bedingungsschlüssel zum Erteilen oder Beschränken des Zugriffs anhand des VPC-Endpunkts.
- Verwenden Sie den `aws:sourceVpc`-Bedingungsschlüssel zum Erteilen oder Beschränken des Zugriffs anhand des VPC, auf der der private Endpunkt gehostet wird.

> **Note**
>
> Beim Erstellen von Schlüsselrichtlinien und IAM-Richtlinien anhand von Ihrem VPC-Endpunkt ist Vorsicht geboten. Wenn eine Richtlinienerklärung vorschreibt, dass Anfragen von einer bestimmten VPC oder einem VPC-Endpunkt kommen, schlagen Anfragen von integrierten AWS Diensten, die eine AWS KMS Ressource in Ihrem Namen verwenden, möglicherweise fehl. Weitere Informationen dazu finden Sie unter Verwenden von VPC-Endpunkt-Bedingungen in Richtlinien mit AWS KMS -Berechtigungen.
> Weiterhin ist der Bedingungsschlüssel `aws:sourceIP` nicht wirksam, wenn die Anforderung von einem Amazon-VPC-Endpunkt kommt. Um die Anforderungen an einen VPC-Endpunkt zu beschränken, verwenden Sie die Bedingungsschlüssel `aws:sourceVpce` oder

`aws:sourceVpc`. Weitere Informationen finden Sie unter Identity and Access Management für VPC-Endpunkte und VPC-Endpunkt-Services im AWS PrivateLink -Leitfaden.

Sie können diese globalen Bedingungsschlüssel verwenden, um den Zugriff auf AWS KMS keys (KMS-Schlüssel), Aliase und solche Operationen zu kontrollieren, CreateKeydie nicht von einer bestimmten Ressource abhängen.

Die folgende Beispiel-Schlüsselrichtlinie erlaubt es einem Benutzer z. B., nur dann kryptografische Operationen mit einem KMS-Schlüssel durchzuführen, wenn die Anforderung den angegebenen VPC-Endpunkt nutzt. Wenn ein Benutzer eine Anfrage an stellt AWS KMS, wird die VPC-Endpunkt-ID in der Anfrage mit dem `aws:sourceVpce` Bedingungsschlüsselwert in der Richtlinie verglichen. Wenn sie nicht übereinstimmen, wird die Anforderung abgelehnt.

Um eine Richtlinie wie diese zu verwenden, ersetzen Sie die AWS-Konto Platzhalter-ID und den VPC-Endpunkt IDs durch gültige Werte für Ihr Konto.

JSON

```
{
    "Id": "example-key-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableIAMpolicies",
            "Effect": "Allow",
            "Principal": {"AWS":["111122223333"]},
            "Action": ["kms:*"],
            "Resource": "*"
        },
        {
            "Sid": "Restrict usage to my VPC endpoint",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringNotEquals": {
                "aws:sourceVpce": "vpce-1234abcdf5678c90a"
                }
            }
        }

    ]
}
```

Sie können auch den `aws:sourceVpc`-Bedingungsschlüssel verwenden, um den Zugriff auf Ihre KMS-Schlüssel anhand der VPC, in der sich der VPC-Endpunkt befindet, zu beschränken.

Die folgende Beispiel-Schlüsselrichtlinie erlaubt nur dann Befehle zur Verwaltung des KMS-Schlüssels, wenn sie von `vpc-12345678` stammen. Außerdem erlaubt sie nur Befehle, die den KMS-Schlüssel für kryptographische Operationen verwenden, wenn sie von `vpc-2b2b2b2b` stammen. Sie verwenden eine solche Richtlinie wie diese möglicherweise, wenn eine Anwendung in einer VPC ausgeführt wird, aber eine zweite, isolierte VPC für die Verwaltungsfunktionen genutzt wird.

Um eine Richtlinie wie diese zu verwenden, ersetzen Sie die AWS-Konto Platzhalter-ID und den VPC-Endpunkt IDs durch gültige Werte für Ihr Konto.

JSON

```
{
    "Id": "example-key-2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAdministrativeActionsFromVPC",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "kms:Create*",
                "kms:Enable*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
```

```
                "kms:Disable*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-12345678"
                }
            }
        },
        {
            "Sid": "AllowKeyUsageFromVPC2b2b2b2b",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-2b2b2b2b"
                }
            }
        },
        {
            "Sid": "AllowReadActionsEverywhere",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "kms:Describe*",
                "kms:List*",
                "kms:Get*"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Protokollierung von AWS KMS Anfragen, die einen VPC-Endpunkt verwenden

AWS CloudTrail protokolliert alle Operationen, die den VPC-Endpunkt verwenden. Wenn eine Anfrage an einen VPC-Endpunkt AWS KMS verwendet, erscheint die VPC-Endpunkt-ID in dem AWS CloudTrail Protokolleintrag, der die Anfrage aufzeichnet. Sie können die Endpunkt-ID verwenden, um die Nutzung Ihres AWS KMS VPC-Endpunkts zu überprüfen.

Ihre CloudTrail Protokolle enthalten jedoch keine Operationen, die von Prinzipalen in anderen Konten angefordert wurden, oder Anfragen für AWS KMS Operationen mit KMS-Schlüsseln und -Aliasnamen in anderen Konten. Um Ihre VPC zu schützen, werden Anfragen, die durch eine VPC-Endpunktrichtlinie abgelehnt wurden, aber andernfalls zugelassen worden wären, nicht aufgezeichnet. AWS CloudTrail

Dieser Beispiel-Protokolleintrag zeichnet z. B. eine GenerateDataKey-Anforderung auf, die den VPC-Endpunkt genutzt hat. Das Feld `vpcEndpointId` erscheint am Ende des Protokolleintrags.

```
{
  "eventVersion":"1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "accountId": "111122223333",
    "userName": "Alice"
  },
  "eventTime":"2018-01-16T05:46:57Z",
  "eventSource":"kms.amazonaws.com",
  "eventName":"GenerateDataKey",
  "awsRegion":"eu-west-1",
  "sourceIPAddress":"172.01.01.001",
  "userAgent":"aws-cli/1.14.23 Python/2.7.12 Linux/4.9.75-25.55.amzn1.x86_64
 botocore/1.8.27",
  "requestParameters":{
    "keyId":"1234abcd-12ab-34cd-56ef-1234567890ab",
    "numberOfBytes":128
  },
```

```
  "responseElements":null,
  "requestID":"a9fff0bf-fa80-11e7-a13c-afcabff2f04c",
  "eventID":"77274901-88bc-4e3f-9bb6-acf1c16f6a7c",
  "readOnly":true,
  "resources":[{
    "ARN":"arn:aws:kms:eu-
west-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId":"111122223333",
    "type":"AWS::KMS::Key"
  }],
  "eventType":"AwsApiCall",
  "recipientAccountId":"111122223333",
  "vpcEndpointId": "vpce-1234abcdf5678c90a"
}
```

Unterstützung für Dual-Stack-Endgeräte

AWS KMS bietet einen öffentlichen Dual-Stack-Endpunkt, der IPv4 sowohl IPv6 Clients als auch unterstützt. Ein Dual-Stack-Endpunkt ermöglicht es Clients, mit ihnen zu kommunizieren, indem sie entweder IPv4 oder IPv6 Adressen AWS KMS verwenden. Weitere Informationen zu den AWS KMS Endpunkten finden Sie unter AWS Key Management Service Endpunkte und Kontingente.

Der öffentliche AWS KMS Dual-Stack-Endpunkt `https://kms.`*`your-region`*`.api.aws` unterstützt sowohl IPv4 Clients als auch. IPv6 AWS KMS ist auch privat über IPv4 und IPv6 von Ihrer Virtual Private Cloud (VPC) aus zugänglich. AWS PrivateLink Weitere Informationen zum Erstellen von VPC-Endpunkten mit privater Schnittstelle für finden Sie AWS KMS unter. Stellen Sie eine Connect AWS KMS über einen VPC-Endpunkt her

Weitere Informationen IPv6 zur Adressierung für Sie VPCs finden Sie unter So funktioniert Amazon VPC im Amazon Virtual Private Cloud Cloud-Benutzerhandbuch. Weitere Informationen zur Konfiguration Ihrer VPC für den Dual-Stack-Modus finden Sie unter IP-Adressierung für Ihre VPCs und Subnetze im Amazon Virtual Private Cloud Cloud-Benutzerhandbuch.

Funktionen, die nicht verfügbar sind über IPv6

AWS KMS kann nicht IPv6 mit AWS CloudHSM Schlüsselspeichern oder externen Schlüsselspeichern kommunizieren. Diese Einschränkung verhindert nicht, dass Sie uns AWS KMS APIs anrufen IPv6.

AWS KMS Konzepte

Lernen Sie die in AWS Key Management Service (AWS KMS) verwendeten grundlegenden Begriffe und Konzepte kennen und erfahren Sie, wie sie zusammen zum Schutz Ihrer Daten beitragen.

Einführung in AWS KMS

AWS Key Management Service (AWS KMS) bietet eine Weboberfläche zur Generierung und Verwaltung kryptografischer Schlüssel und fungiert als Anbieter von kryptografischen Diensten zum Schutz von Daten. AWS KMS bietet traditionelle Schlüsselverwaltungsdienste, die in Dienste integriert sind, um einen konsistenten Überblick über die Schlüssel der Kunden zu bieten AWS, und zwar mit zentraler Verwaltung und Prüfung. AWS

AWS KMS umfasst eine Weboberfläche über die AWS Management Console Befehlszeile und RESTful API-Operationen zur Anforderung kryptografischer Operationen einer verteilten Flotte von FIPS 140-3-validierten Hardware-Sicherheitsmodulen (). HSMs Das AWS KMS HSM ist eine eigenständige Hardware-Kryptografie-Appliance mit mehreren Chips, die spezielle kryptografische Funktionen bereitstellt, um die Sicherheits- und Skalierbarkeitsanforderungen von zu erfüllen. AWS KMS Sie können Ihre eigene HSM-basierte kryptografische Hierarchie unter Schlüsseln einrichten, die Sie als AWS KMS keys verwaltet werden. Diese Schlüssel werden nur auf dem HSMs und nur für den Zeitraum im Speicher zur Verfügung gestellt, der für die Bearbeitung Ihrer kryptografischen Anfrage erforderlich ist. Sie können mehrere KMS-Schlüssel erstellen, die jeweils durch seine Schlüssel-ID dargestellt werden. Nur im Rahmen von AWS IAM-Rollen und -Konten, die von jedem Kunden verwaltet werden, können vom Kunden verwaltete KMS-Schlüssel erstellt, gelöscht oder zum Verschlüsseln, Entschlüsseln, Signieren oder Überprüfen von Daten verwendet werden. Sie können Zugriffskontrollen dafür definieren, wer die and/or Nutzung von KMS-Schlüsseln verwalten kann, indem Sie eine Richtlinie erstellen, die an den Schlüssel angehängt ist. Mithilfe dieser Richtlinien können Sie anwendungsspezifische Verwendungen für Ihre Schlüssel für jeden API-Vorgang definieren.

Darüber hinaus unterstützen die meisten AWS Dienste die Verschlüsselung ruhender Daten mithilfe von KMS-Schlüsseln. Mit dieser Funktion können Kunden steuern, wie und wann AWS Dienste auf verschlüsselte Daten zugreifen können, indem sie steuern, wie und wann auf KMS-Schlüssel zugegriffen werden kann.

AWS KMS ist ein mehrstufiger Dienst, der aus mit dem Internet verbundenen AWS KMS Hosts und einer Stufe von besteht. HSMs Die Gruppierung dieser mehrstufigen Hosts bildet den Stack. AWS KMS Alle Anfragen an AWS KMS müssen über das Transport Layer Security-Protokoll (TLS) gestellt und auf einem AWS KMS Host beendet werden. AWS KMS Hosts erlauben TLS nur mit einer Ciphersuite, die Perfect Forward Secrecy bietet. AWS KMS authentifiziert und autorisiert Ihre Anfragen mit denselben Anmelde- und Richtlinienmechanismen von AWS Identity and Access Management (IAM), die für alle anderen API-Operationen verfügbar sind. AWS

AWS KMS Ziele des Entwurfs

AWS KMS wurde entwickelt, um die folgenden Anforderungen zu erfüllen.

Haltbarkeit

Die Haltbarkeit kryptografischer Schlüssel ist so konzipiert, dass sie der Lebensdauer von Diensten mit der höchsten Haltbarkeit entspricht. AWS Ein einzelner kryptografischer Schlüssel kann große Datenmengen verschlüsseln, die sich über eine lange Zeit angesammelt haben.

Vertrauenswürdig

Die Verwendung von Schlüsseln ist durch Zugriffssteuerungsrichtlinien geschützt, die Sie definieren und verwalten. Es gibt keinen Mechanismus, um Klartext-KMS-Schlüssel zu exportieren. Die Vertraulichkeit Ihrer kryptografischen Schlüssel ist von entscheidender Bedeutung. Für die Durchführung administrativer Aktionen am sind mehrere Amazon-Mitarbeiter mit rollenspezifischem Zugriff auf quorumbasierte Zugriffskontrollen erforderlich. HSMs

Niedrige Latenz und hoher Durchsatz

AWS KMS bietet kryptografische Operationen mit Latenz- und Durchsatzniveaus, die für die Verwendung durch andere Dienste in geeignet sind. AWS

Unabhängige Regionen

AWS bietet unabhängige Regionen für Kunden, die den Datenzugriff in verschiedenen Regionen einschränken müssen. Die Schlüsselverwendung kann innerhalb eines AWS-Region isoliert werden.

Sichere Quelle von Zufallszahlen

Da starke Kryptografie von einer wirklich unvorhersehbaren Generierung von Zufallszahlen abhängt, AWS KMS bietet es eine qualitativ hochwertige und validierte Quelle für Zufallszahlen.

Audit

AWS KMS zeichnet die Verwendung und Verwaltung kryptografischer Schlüssel in AWS CloudTrail Protokollen auf. Mithilfe von AWS CloudTrail Protokollen können Sie die Verwendung Ihrer kryptografischen Schlüssel überprüfen, einschließlich der Verwendung von Schlüsseln durch AWS Dienste in Ihrem Namen.

Um diese Ziele zu erreichen, umfasst das AWS KMS System eine Reihe von AWS KMS Operatoren und Service-Host-Betreibern (zusammen „Betreiber“), die „Domains“ verwalten. Eine Domain ist eine regional definierte Gruppe von AWS KMS Servern und HSMs Operatoren. Jeder AWS KMS Operator verfügt über ein Hardware-Token, das ein privates und ein öffentliches key pair enthält, das zur Authentifizierung seiner Aktionen verwendet wird. HSMs Sie verfügen über ein zusätzliches privates und öffentliches key pair, um Verschlüsselungsschlüssel einzurichten, die die HSM-Statussynchronisierung schützen.

AWS KMS keys

Bei den KMS-Schlüsseln, die Sie für die Verwendung in Ihren eigenen kryptografischen Anwendungen erstellen und verwalten, handelt es sich um vom Kunden verwaltete Schlüssel. Vom Kunden verwaltete Schlüssel können auch in Verbindung mit AWS Diensten verwendet werden, die KMS-Schlüssel verwenden, um die Daten zu verschlüsseln, die der Dienst in Ihrem Namen speichert. Kundenverwaltete Schlüssel werden für Kunden empfohlen, die die volle Kontrolle über den Lebenszyklus und die Verwendung ihrer Schlüssel haben möchten. Es fallen monatliche Kosten an, wenn Sie einen vom Kunden verwalteten Schlüssel in Ihrem Konto haben. Darüber hinaus fallen für Anfragen, die den Schlüssel and/or verwalten, Nutzungskosten an. Weitere Informationen finden Sie unter AWS Key Management Service Preise.

Es gibt Fälle, in denen ein Kunde zwar möchte, dass ein AWS Dienst seine Daten verschlüsselt, er möchte aber nicht den Aufwand für die Verwaltung von Schlüsseln übernehmen und nicht für einen Schlüssel bezahlen möchte. An Von AWS verwalteter Schlüsselist ein KMS-Schlüssel, der in Ihrem Konto vorhanden ist, aber nur unter bestimmten Umständen verwendet werden kann. Insbesondere kann er nur im Kontext des AWS Dienstes verwendet werden, in dem Sie tätig sind, und er kann nur von Prinzipalen innerhalb des Kontos verwendet werden, in dem der Schlüssel vorhanden ist. Sie können nichts über den Lebenszyklus oder die Berechtigungen dieser Schlüssel verwalten. Wenn Sie Verschlüsselungsfunktionen in AWS Diensten verwenden, werden Sie vielleicht feststellen Von AWS verwaltete Schlüssel, dass sie einen Alias der Form „aws<service code>“ verwenden. Ein `aws/ebs` Schlüssel kann beispielsweise nur zum Verschlüsseln von EBS-Volumes und nur für Volumes

verwendet werden, die von IAM-Prinzipalen im selben Konto wie der Schlüssel verwendet werden. Stellen Sie sich einen vor Von AWS verwalteter Schlüssel , der nur von Benutzern in Ihrem Konto für Ressourcen in Ihrem Konto verwendet werden kann. Sie können Ressourcen, die unter oder verschlüsselt wurden, nicht Von AWS verwalteter Schlüssel mit anderen Konten teilen. Von AWS verwalteter Schlüssel Es ist zwar kostenlos, ein in Ihrem Konto zu speichern, aber die Nutzung dieses Schlüsseltyps durch den AWS Dienst, der dem Schlüssel zugewiesen ist, wird Ihnen in Rechnung gestellt.

Von AWS verwaltete Schlüssel sind ein veralteter Schlüsseltyp, der ab 2021 nicht mehr für neue AWS Dienste erstellt wird. Stattdessen verwenden neue (und ältere) AWS Dienste standardmäßig sogenannte An AWS-eigener Schlüsselzur Verschlüsselung von Kundendaten. An AWS-eigener Schlüssel ist ein KMS-Schlüssel, der sich in einem vom AWS Dienst verwalteten Konto befindet, sodass die Servicebetreiber den Lebenszyklus und die Nutzungsberechtigungen verwalten können. Durch die Verwendung AWS-eigene Schlüssel können AWS Dienste Ihre Daten transparent verschlüsseln und eine einfache konto- oder regionsübergreifende gemeinsame Nutzung von Daten ermöglichen, ohne dass Sie sich Gedanken über wichtige Berechtigungen machen müssen. Wird AWS-eigene Schlüssel für encryption-by-default Workloads verwendet, die einen einfacheren, automatisierteren Datenschutz bieten. Da diese Schlüssel Eigentum sind und von ihnen verwaltet werden AWS, fallen keine Gebühren für ihre Existenz oder Nutzung an. Sie können ihre Richtlinien nicht ändern, Sie können keine Aktivitäten mit diesen Schlüsseln überprüfen und Sie können sie nicht löschen. Verwenden Sie vom Kunden verwaltete Schlüssel, wenn Kontrolle wichtig ist, aber verwenden Sie AWS-eigene Schlüssel sie, wenn Komfort am wichtigsten ist.

	Vom Kunden verwaltete Schlüssel	Von AWS verwaltete Schlüssel	AWS-eigene Schlüssel
Schlüsselrichtlinie	Ausschließlich vom Kunden kontrolliert	Wird vom Service gesteuert; vom Kunden einsehbar	Ausschließlich kontrolliert und nur von dem AWS Dienst einsehbar, der Ihre Daten verschlüsselt
Protokollierung	CloudTrail Datenspeicher für Kundendaten oder Ereignisse	CloudTrail Datenspeicher für Kundendaten oder Ereignisse	Für den Kunden nicht sichtbar
Lebenszyklusmanagement	Der Kunde verwaltet Rotation, Löschung	AWS KMS verwaltet Rotation (jährlich	AWS-Service verwaltet Rotation,

	und regionalen Standort	), Löschung und regionalen Standort	Löschung und regionalen Standort
Preise	Monatliche Gebühr für das Vorhandensein von Schlüsseln (anteilig pro Stunde). Wird auch für die Nutzung von Schlüsseln berechnet	Keine monatliche Gebühr; dem Anrufer wird jedoch die API-Nutzung für diese Schlüssel in Rechnung gestellt	Keine Gebühren für den Kunden

Die von Ihnen erstellten KMS-Schlüssel sind kundenverwaltete Schlüssel. AWS-Services , die KMS-Schlüssel zur Entschlüsselung Ihrer Service-Ressourcen verwenden, erstellen oft Schlüssel für Sie. KMS-Schlüssel, die in Ihrem AWS Konto AWS-Services erstellt werden, sind Von AWS verwaltete Schlüssel. KMS-Schlüssel, die in einem Dienstkonto AWS-Services erstellt werden, sind AWS-eigene Schlüssel.

Typ des KMS-Schlüssels	Kann KMS-Schlüsselmetadaten anzeigen	Kann KMS-Schlüssel verwalten	Wird nur für meine verwendet AWS-Konto	Automatisches Rotieren	Preise
Kundenverwalteter Schlüssel	Ja	Ja	Ja	Optional.	Monatliche Gebühr (anteilig stündlich) Gebühr pro Nutzung
Von AWS verwalteter Schlüssel	Ja	Nein	Ja	Erforderlich Jedes Jahr (ungefähr 365 Tage).	Keine monatliche Gebühr Gebühr pro Nutzung

Typ des KMS-Schlüssels	Kann KMS-Schlüsselmetadaten anzeigen	Kann KMS-Schlüssel verwalten	Wird nur für meine verwendet AWS-Konto	Automatisches Rotieren	Preise
					(einige AWS-Services zahlen diese Gebühr für Sie)
AWS-eigener Schlüssel	Nein	Nein	Nein	Der AWS-Service verwaltet die Rotationsstrategie	Keine Gebühren

AWS Dienste, die in integriert werden, AWS KMS unterscheiden sich in ihrer Unterstützung für KMS-Schlüssel. Einige AWS Dienste verschlüsseln Ihre Daten standardmäßig mit einem AWS-eigener Schlüssel oder einem Von AWS verwalteter Schlüssel. Einige AWS Dienste unterstützen vom Kunden verwaltete Schlüssel. Andere AWS Dienste unterstützen alle Arten von KMS-Schlüsseln AWS-eigener Schlüssel, sodass Sie einen vom Kunden verwalteten Schlüssel einfach Von AWS verwalteter Schlüssel, sichtbar oder kontrollieren können. Ausführliche Informationen zu den Verschlüsselungsoptionen, die ein AWS Dienst bietet, finden Sie unter dem Thema Verschlüsselung im Ruhezustand im Benutzer- oder Entwicklerhandbuch für den Dienst.

Kundenverwaltete Schlüssel

Die von Ihnen erstellten KMS-Schlüssel sind kundenverwaltete Schlüssel. Kundenverwaltete Schlüssel sind KMS-Schlüssel in Ihrem AWS-Konto System, die Sie selbst erstellen, besitzen und verwalten. Sie haben die volle Kontrolle über diese KMS-Schlüssel. Dies gilt auch für die Festlegung und Verwaltung ihrer Schlüsselrichtlinien, IAM-Richtlinien und Erteilungen, ihre Aktivierung und Deaktivierung, die Drehung ihrer Verschlüsselungsinformationen, das Hinzufügen von Tags, das Erstellen von Aliassen, die sich auf die KMS-Schlüssel beziehen, und das Einplanen der KMS-Schlüssel zum Löschen.

Kundenverwaltete Schlüssel werden auf der Seite Customer managed keys (Kundenverwaltete Schlüssel) der AWS Management Console für AWS KMS angezeigt. Verwenden Sie den DescribeKeyVorgang, um einen vom Kunden verwalteten Schlüssel eindeutig zu identifizieren. Für kundenverwaltete Schlüssel ist der Wert des `KeyManager`-Felds der `DescribeKey`-Antwort `CUSTOMER`.

Sie können Ihren kundenverwalteten Schlüssel in kryptografischen Operationen verwenden und ihre Verwendung in AWS CloudTrail -Protokollen prüfen. Darüber hinaus bieten Ihnen viele mit AWS KMS integrierte AWS -Services die Möglichkeit, einen kundenverwalteten Schlüssel zum Schutz der Daten anzugeben, die für Sie gespeichert und verwaltet werden.

Für kundenverwaltete Schlüssel fällt eine monatliche Gebühr sowie eine Gebühr für die über das kostenlose Kontingent hinausgehende Nutzung an. Sie werden auf die AWS KMS Kontingente für Ihr Konto angerechnet. Weitere Informationen finden Sie unter AWS Key Management Service – Preise und Kontingente.

Von AWS verwaltete Schlüssel

Von AWS verwaltete Schlüsselsind KMS-Schlüssel in Ihrem Konto, die in Ihrem Namen von einem integrierten AWS Dienst erstellt, verwaltet und verwendet werden AWS KMS.

Bei einigen AWS Diensten können Sie einen Von AWS verwalteter Schlüssel oder einen vom Kunden verwalteten Schlüssel auswählen, um Ihre Ressourcen in diesem Dienst zu schützen. Generell gilt: Sofern Sie nicht selbst die Kontrolle über den Verschlüsselungsschlüssel haben müssen, der Ihre Ressourcen schützt, Von AWS verwalteter Schlüssel ist an eine gute Wahl. Sie müssen den Schlüssel oder seine Schlüsselrichtlinie nicht erstellen oder verwalten, und für einen Von AWS verwalteter Schlüssel gibt es keine monatliche Gebühr.

Sie sind berechtigt, sie Von AWS verwaltete Schlüssel in Ihrem Konto einzusehen, ihre wichtigsten Richtlinien einzusehen und ihre Verwendung in AWS CloudTrail Protokollen zu überprüfen. Sie können jedoch keine Eigenschaften von ändern Von AWS verwaltete Schlüssel, sie rotieren, ihre wichtigsten Richtlinien ändern oder ihre Löschung planen. Und Sie können sie nicht direkt für kryptografische Operationen verwenden Von AWS verwaltete Schlüssel . Der Dienst, der sie erstellt, verwendet sie in Ihrem Namen.

Die Richtlinien zur Ressourcenkontrolle in Ihrer Organisation gelten nicht für. Von AWS verwaltete Schlüssel

Von AWS verwaltete Schlüssel erscheinen auf der Von AWS verwaltete SchlüsselSeite des AWS Management Console Formulars AWS KMS. Sie können es auch Von AWS verwaltete

Schlüssel anhand ihrer Aliase identifizieren, die das Format haben`aws/service-name`, z. B. `aws/redshift` Verwenden Sie die Operation Von AWS verwaltete Schlüssel, um einen eindeutig zu identifizieren. DescribeKey Für Von AWS verwaltete Schlüssel ist der Wert des `KeyManager`-Felds der `DescribeKey`-Antwort `AWS`.

Alle Von AWS verwaltete Schlüssel werden jedes Jahr automatisch rotiert. Dieser Rotationsplan kann nicht geändert werden.

> **Note**
>
> Im Mai 2022 AWS KMS wurde der Rotationsplan Von AWS verwaltete Schlüssel von allen drei Jahren (ungefähr 1.095 Tage) auf jedes Jahr (ungefähr 365 Tage) geändert.

Es gibt keine monatliche Gebühr für. Von AWS verwaltete Schlüssel Für eine Nutzung, die über das kostenlose Kontingent hinausgeht, können Gebühren anfallen. Einige AWS Dienste übernehmen diese Kosten jedoch für Sie. Weitere Informationen finden Sie im Benutzerhandbuch oder Entwicklerhandbuch für den Service unter dem Thema Verschlüsselung im Ruhezustand. Für Einzelheiten vgl. AWS Key Management Service -Preise.

Von AWS verwaltete Schlüssel rechnen Sie die Anzahl der KMS-Schlüssel in jeder Region Ihres Kontos nicht auf die Ressourcenkontingente an. Wenn die KMS-Schlüssel jedoch im Namen eines Prinzipals in Ihrem Konto verwendet werden, werden sie auf die Anforderungskontingente angerechnet. Details hierzu finden Sie unter Kontingente.

AWS-eigene Schlüssel

AWS-eigene Schlüsselsind eine Sammlung von KMS-Schlüsseln, die ein AWS Dienst besitzt und verwaltet, sodass sie in mehreren Fällen verwendet AWS-Konten werden können. Sie AWS-eigene Schlüssel gehören zwar nicht zu Ihrem AWS-Konto, können aber von einem AWS Dienst verwendet werden AWS-eigener Schlüssel , um die Ressourcen in Ihrem Konto zu schützen.

Bei einigen AWS Diensten können Sie einen AWS-eigener Schlüssel oder einen vom Kunden verwalteten Schlüssel auswählen. Generell gilt: Sofern Sie nicht verpflichtet sind, den Verschlüsselungsschlüssel, der Ihre Ressourcen schützt, zu überprüfen oder zu kontrollieren, AWS-eigener Schlüssel ist an eine gute Wahl. AWS-eigene Schlüssel sind völlig kostenlos (keine monatlichen Gebühren oder Nutzungsgebühren), sie werden nicht auf die AWS KMS Kontingente für Ihr Konto angerechnet und sie sind einfach zu verwenden. Sie müssen den Schlüssel oder seine Schlüsselrichtlinie nicht erstellen oder pflegen.

Die Rotation von AWS-eigene Schlüssel variiert je nach Dienst. Informationen zur Rotation eines bestimmten AWS-eigener Schlüssel Dienstes finden Sie unter dem Thema Verschlüsselung im Ruhezustand im Benutzer- oder Entwicklerhandbuch für den Dienst.

AWS KMS key Hierarchie

Ihre Schlüsselhierarchie beginnt mit einem logischen Schlüssel der obersten Ebene, einem. AWS KMS key Ein KMS-Schlüssel stellt einen Container für Schlüsselmaterial der obersten Ebene dar und ist innerhalb des AWS -Service-Namespace mit einem Amazon-Ressourcennamen (ARN) eindeutig definiert. Der ARN enthält eine eindeutig generierte Schlüsselkennung, eine Schlüssel-ID. Ein KMS-Schlüssel wird auf der Grundlage einer vom Benutzer initiierten Anfrage über erstellt. AWS KMS AWS KMS Fordert bei Empfang die Erstellung eines ersten HSM-Backing-Schlüssels (HBK) an, der im KMS-Schlüsselcontainer platziert werden soll. Der HBK wird auf einem HSM in der Domäne generiert und ist so konzipiert, dass er niemals im Klartext aus dem HSM exportiert wird. Stattdessen wird der HBK unter HSM-verwalteten Domänenschlüsseln verschlüsselt exportiert. Diese exportierten Token HBKs werden als exportierte Schlüsseltoken () EKTs bezeichnet.

Der EKT wird in einen äußerst langlebigen Speicher mit geringer Latenz exportiert. Angenommen, Sie erhalten einen ARN für den logischen KMS-Schlüssel. Dies stellt für Sie die Spitze einer Schlüsselhierarchie oder eines kryptografischen Kontexts dar. Sie können mehrere KMS-Schlüssel in Ihrem Konto erstellen und Richtlinien für Ihre KMS-Schlüssel wie für jede andere AWS benannte Ressource festlegen.

Innerhalb der Hierarchie eines bestimmten KMS-Schlüssels kann man sich den HBK als eine Version des KMS-Schlüssels vorstellen. Wenn Sie den KMS-Schlüssel rotieren möchten AWS KMS, wird ein neuer HBK erstellt und dem KMS-Schlüssel als aktive HBK für den KMS-Schlüssel zugeordnet. Ältere Daten HBKs bleiben erhalten und können zum Entschlüsseln und Überprüfen zuvor geschützter Daten verwendet werden. Aber nur der aktive kryptografische Schlüssel kann verwendet werden, um neue Informationen zu schützen.

Sie können Anfragen stellen AWS KMS , um Ihre KMS-Schlüssel zum direkten Schutz von Informationen zu verwenden, oder zusätzliche HSM-generierte Schlüssel anfordern, die unter Ihrem KMS-Schlüssel geschützt sind. Diese Schlüssel werden Kundendatenschlüssel oder genannt. CDKs CDKs können verschlüsselt als Chiffretext (CT), im Klartext oder in beidem zurückgegeben werden. Alle mit einem KMS-Schlüssel verschlüsselten Objekte (entweder vom Kunden bereitgestellte Daten oder von HSM generierte Schlüssel) können nur auf einem HSM per Call Through entschlüsselt werden. AWS KMS

Der zurückgegebene Chiffretext oder die entschlüsselte Nutzlast werden niemals darin gespeichert. AWS KMS Die Informationen werden Ihnen über Ihre TLS-Verbindung an AWS KMS zurückgesendet. Dies gilt auch für Anrufe, die von AWS Diensten in Ihrem Namen getätigt werden.

Die Schlüsselhierarchie und die spezifischen Schlüsseleigenschaften werden in der folgenden Tabelle angezeigt.

Schlüssel	Beschreibung	Lebenszyklus
Domain-Schlüssel	Ein 256-Bit-AES-GCM-Schlüssel nur im Speicher eines HSM, der verwendet wird, um Versionen der KMS-Schlüssel, die HSM-Unter stützungsschlüssel einzuschließen.	Täglich gedreht [1]
HSM-Unterstützungsschlüssel	Ein symmetrischer 256-Bit-Schlüssel oder privater RSA- oder elliptischer Kurvenschlüssel, der zum Schutz von Kundendaten und -schlüsseln verwendet und unter Domänensc hlüsseln verschlüsselt gespeichert wird. Ein oder mehrere HSM-Unter stützungsschlüssel umfassen den KMS-Schlüssel, dargestellt durch die keyId.	Jährlich gedreht [2] (optionale Konfiguration)
Abgeleiteter Verschlüsselungssc hlüssel	Ein 256-Bit-AES-GCM-Schlüssel nur im Speicher eines HSM, der zum Verschlüsseln von Kundendaten und Schlüsseln verwendet wird. Abgeleite t von einem HBK für jede Verschlüs selung.	Wird einmal pro Verschlüsselung verwendet und beim Entschlüs seln regeneriert
Kunden-Datenschlüssel	Benutzerdefinierter symmetrischer oder asymmetrischer Schlüssel, der aus HSM in Klartext und Verschlüs selungstext exportiert wird.	Drehung und Nutzung durch Anwendung gesteuert

Schlüssel	Beschreibung	Lebenszyklus
	Unter einem HSM-Unterstützungs schlüssel verschlüsselt und über den TLS-Kanal an autorisierte Benutzer zurückgegeben.	

[Ich] AWS KMS könnte die Rotation der Domänenschlüssel von Zeit zu Zeit auf höchstens wöchentlich lockern, um die Aufgaben der Domänenverwaltung und -konfiguration zu berücksichtigen.

[2] Standardmäßig, die in AWS KMS Ihrem Namen Von AWS verwaltete Schlüssel erstellt und verwaltet werden, werden jährlich automatisch gewechselt.

Schlüsselkennungen () KeyId

Schlüsselbezeichner fungieren als Namen für Ihre KMS-Schlüssel. Sie helfen Ihnen, Ihre KMS-Schlüssel in der Konsole zu erkennen. Sie verwenden sie, um anzugeben, welche KMS-Schlüssel Sie in AWS KMS -API-Operationen, Schlüsselrichtlinien, IAM-Richtlinien und Erteilungen verwenden möchten. Die Schlüsselkennungswerte stehen in keinem Zusammenhang mit dem Schlüsselmaterial, das dem KMS-Schlüssel zugeordnet ist.

AWS KMS definiert mehrere Schlüsselkennungen. Wenn Sie einen KMS-Schlüssel erstellen, AWS KMS generiert er einen Schlüssel-ARN und eine Schlüssel-ID, die Eigenschaften des KMS-Schlüssels sind. Wenn Sie einen Alias erstellen, AWS KMS generiert er einen Alias-ARN auf der Grundlage des von Ihnen definierten Aliasnamens. Sie können den Schlüssel und die Alias-IDs in der AWS Management Console und in der AWS KMS API einsehen.

In der AWS KMS Konsole können Sie KMS-Schlüssel nach ihrem Schlüssel-ARN, ihrer Schlüssel-ID oder ihrem Aliasnamen anzeigen und filtern und nach Schlüssel-ID und Aliasnamen sortieren. Hilfestellung beim Suchen der Schlüsselbezeichner in der Konsole finden Sie unter the section called “Finden Sie die Schlüssel-ID und den Schlüssel-ARN”.

In der AWS KMS API haben die Parameter, die Sie zur Identifizierung eines KMS-Schlüssels verwenden, einen Namen `KeyId` oder eine Variante, z. B. `TargetKeyId` oder`DestinationKeyId`. Die Werte dieser Parameter sind jedoch nicht auf Schlüssel beschränkt IDs. Für einige eignet sich jeder gültige Schlüsselbezeichner. Informationen zu den Werten für die einzelnen Parameter finden Sie in der Parameterbeschreibung in der AWS Key Management Service API-Referenz.

> **Note**
>
> Achten Sie bei der Verwendung der AWS KMS API darauf, welche Schlüssel-ID Sie verwenden. APIs Für unterschiedliche sind unterschiedliche Schlüsselkennungen erforderlich. Verwenden Sie im Allgemeinen die vollständigste Schlüsselbezeichnung, die für Ihre Aufgabe praktisch ist.

AWS KMS unterstützt die folgenden Schlüsselkennungen.

Schüssel-ARN

Der Schlüssel-ARN ist der Amazon-Ressourcenname (ARN) eines KMS-Schlüssels. Er ist eine eindeutige, vollqualifizierte Kennung für den KMS-Schlüssel. Ein Schlüssel-ARN umfasst die AWS-Konto, Region und die Schlüssel-ID. Hilfestellung beim Suchen des Schlüssel-ARN eines KMS-Schlüssels finden Sie unter the section called “Finden Sie die Schlüssel-ID und den Schlüssel-ARN”.

Das Format eines Schlüssel-ARN lautet wie folgt:

```
arn:<partition>:kms:<region>:<account-id>:key/<key-id>
```

Es folgt ein Beispiel eines Schlüssel-ARNs für einen einzelregionalen KMS-Schlüssel.

```
arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Das *`key-id`* ARNs Schlüsselelement von Schlüsseln mit mehreren Regionen beginnt mit dem `mrk-` Präfix. Es folgt ein Beispiel eines Schlüssel-ARN für einen multiregionalen KMS-Schlüssel.

```
arn:aws:kms:us-west-2:111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab
```

Schlüssel-ID

Die Schlüssel-ID identifiziert einen KMS-Schlüssel innerhalb eines Kontos und einer Region eindeutig. Hilfestellung beim Suchen der Schlüssel-ID eines KMS-Schlüssels finden Sie unter the section called “Finden Sie die Schlüssel-ID und den Schlüssel-ARN”.

Es folgt ein Beispiel einer Schlüssel-ID für einen einzelregionalen KMS-Schlüssel.

```
1234abcd-12ab-34cd-56ef-1234567890ab
```

Der Schlüssel IDs von Schlüsseln für mehrere Regionen beginnt mit dem `mrk-` Präfix. Es folgt ein Beispiel einer Schlüssel-ID für einen multiregionalen KMS-Schlüssel.

```
mrk-1234abcd12ab34cd56ef1234567890ab
```

Alias-ARN

Der Alias-ARN ist der Amazon-Ressourcenname (ARN) eines AWS KMS Alias. Es ist ein eindeutiger, vollqualifizierter Bezeichner für den Alias und für den KMS-Schlüssel, den er repräsentiert. Ein Alias-ARN umfasst die AWS-Konto, Region und den Aliasnamen.

Ein Alias-ARN identifiziert zu einem bestimmten Zeitpunkt einen bestimmten KMS-Schlüssel. Da Sie jedoch den KMS-Schlüssel ändern können, der dem Alias zugeordnet ist, kann der Alias-ARN zu verschiedenen Zeiten unterschiedliche KMS-Schlüssel identifizieren. Hilfestellung beim Suchen des Alias-ARN eines KMS-Schlüssels finden Sie unter Suchen Sie den Aliasnamen und den Alias-ARN für einen KMS-Schlüssel.

Das Format eines Alias-ARN lautet wie folgt:

```
arn:<partition>:kms:<region>:<account-id>:alias/<alias-name>
```

Es folgt der Alias-ARN für einen fiktiven `ExampleAlias`.

```
arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias
```

Alias-Name

Der Aliasname besteht aus einer Zeichenfolge mit bis zu 256 Zeichen. Er identifiziert einen zugeordneten KMS-Schlüssel innerhalb eines Kontos und einer Region eindeutig. In der AWS KMS API beginnen Aliasnamen immer mit`alias/`. Hilfestellung beim Suchen des Aliasnamens eines KMS-Schlüssels finden Sie unter Suchen Sie den Aliasnamen und den Alias-ARN für einen KMS-Schlüssel.

Das Format eines Aliasnamens lautet wie folgt:

```
alias/<alias-name>
```

Zum Beispiel:

```
alias/ExampleAlias
```

Das `aws/`-Präfix für einen Aliasnamen ist für Von AWS verwaltete Schlüssel reserviert. Sie können keinen Alias mit diesem Präfix erstellen. Der Aliasname von Von AWS verwalteter Schlüssel für Amazon Simple Storage Service (Amazon S3) lautet beispielsweise wie folgt.

```
alias/aws/s3
```

Asymmetrische Tasten in AWS KMS

Ein asymmetrischer KMS-Schlüssel repräsentiert einen mathematisch verwandtes Schlüsselpaar mit einem öffentlichen und einem privaten Schlüssel. Sie können den öffentlichen Schlüssel jedem geben, auch wenn er nicht vertrauenswürdig sind, aber der private Schlüssel muss geheim gehalten werden.

Bei einem asymmetrischen KMS-Schlüssel wird der private Schlüssel erstellt AWS KMS und verlässt AWS KMS ihn nie unverschlüsselt. Um den privaten Schlüssel zu verwenden, müssen Sie anrufen. AWS KMS Sie können den öffentlichen Schlüssel darin verwenden, AWS KMS indem Sie die AWS KMS API-Operationen aufrufen. Oder Sie können den öffentlichen Schlüssel herunterladen und ihn außerhalb von verwenden AWS KMS.

Wenn Ihr Anwendungsfall eine Verschlüsselung außerhalb AWS von Benutzern erfordert, die nicht anrufen können AWS KMS, sind asymmetrische KMS-Schlüssel eine gute Wahl. Wenn Sie jedoch einen KMS-Schlüssel erstellen, um die Daten zu verschlüsseln, die Sie in einem AWS Dienst speichern oder verwalten, verwenden Sie einen KMS-Schlüssel mit symmetrischer Verschlüsselung. AWS Dienste, in die integriert sind, AWS KMS verwenden nur KMS-Schlüssel mit symmetrischer Verschlüsselung, um Ihre Daten zu verschlüsseln. Diese Services unterstützen keine Verschlüsselung mit asymmetrischen KMS-Schlüsseln.

AWS KMS unterstützt drei Arten von asymmetrischen KMS-Schlüsseln.

RSA-KMS-Schlüssel

Ein KMS-Schlüssel mit einem RSA-Schlüsselpaar für Verschlüsselung und Entschlüsselung oder Signierung und Überprüfung (aber nicht beides). AWS KMS unterstützt mehrere Schlüssellängen für unterschiedliche Sicherheitsanforderungen.

Technische Informationen zu den Verschlüsselungs- und Signierungsalgorithmen, die RSA-KMS-Schlüssel AWS KMS unterstützen, finden Sie unter RSA-Schlüsselspezifikationen.

KMS-Schlüssel mit elliptischer Kurve (ECC)

Ein KMS-Schlüssel mit einem key pair mit elliptischer Kurve zum Signieren und Überprüfen oder zum Ableiten von gemeinsamen Geheimnissen (aber nicht beides). AWS KMS unterstützt mehrere häufig verwendete Kurven.

Technische Informationen zu den Signaturalgorithmen, die ECC-KMS-Schlüssel AWS KMS unterstützen, finden Sie unter Schlüsselspezifikationen für elliptische Kurven.

ML-DSA-KMS-Schlüssel

Ein KMS-Schlüssel mit einem ML-DSA-Schlüsselpaar zum Signieren und Überprüfen. ML-DSA ist ein Postquanten-Kryptografie-Standard, der vom US-amerikanischen National Institute of Standards and Technology (NIST) zum Schutz vor Sicherheitsbedrohungen durch Quantencomputer entwickelt wurde. ML-DSA ist der empfohlene Algorithmus für digitale Signaturen für Unternehmen, die von digitalen Signaturalgorithmen RSA oder Elliptic Curve auf sichere Postquanten-Kryptografie umsteigen.

AWS KMS unterstützt mehrere Schlüssellängen für unterschiedliche Sicherheitsanforderungen. Technische Informationen zu den Signaturalgorithmen, die ML-DSA-KMS-Schlüssel AWS KMS unterstützen, finden Sie unter ML-DSA-Schlüsselspezifikation.

SM2 KMS-Schlüssel (nur Regionen China)

Ein KMS-Schlüssel mit einem SM2 key pair für Verschlüsselung und Entschlüsselung, Signierung und Überprüfung oder Ableitung gemeinsamer Geheimnisse (Sie müssen einen Key usage Typ auswählen).

Technische Informationen zu den Verschlüsselungs- und Signierungsalgorithmen, die SM2 KMS-Schlüssel AWS KMS unterstützen (nur Regionen China), finden Sie unter SM2 Schlüsselspezifikation.

Hilfe bei der Auswahl Ihrer asymmetrischen Schlüsselkonfiguration finden Sie unter Wählen Sie aus, welche Art von KMS-Schlüssel erstellt werden soll.

Regionen

Asymmetrische KMS-Schlüssel und asymmetrische Datenschlüsselpaare werden in allem AWS-Regionen unterstützt, was unterstützt wird. AWS KMS

Weitere Informationen

- Informationen zum Erstellen asymmetrischer KMS-Schlüssel finden Sie unter Erstellen eines asymmetrischen KMS-Schlüssels.
- Informationen zum Erstellen multiregionaler asymmetrischer KMS-Schlüssel finden Sie unter Primärschlüssel für mehrere Regionen erstellen.
- Informationen zum Signieren von Nachrichten und Überprüfen von Signaturen mit asymmetrischen KMS-Schlüsseln finden Sie unter Digitale Signierung mit dem neuen asymmetrischen Schlüssel-Feature von AWS KMS im AWS -Sicherheits-Blog.
- Informationen zu besonderen Überlegungen beim Löschen asymmetrischer KMS-Schlüssel finden Sie unter. Deleting asymmetric KMS keys
- Informationen zum Identifizieren und Anzeigen von asymmetrischen KMS-Schlüsseln finden Sie unter. Identifizieren Sie asymmetrische KMS-Schlüssel

HMAC-Schlüssel eingeben AWS KMS

Bei KMS-Schlüsseln (Hash-Based Message Authentication Code, HMAC) handelt es sich um symmetrische Schlüssel, die Sie zur Generierung und Überprüfung innerhalb von KMS-Schlüsseln verwenden. HMACs AWS KMS Das eindeutige Schlüsselmaterial, das mit jedem HMAC-KMS-Schlüssel verbunden ist, liefert den geheimen Schlüssel, den HMAC-Algorithmen benötigen. Sie können einen HMAC-KMS-Schlüssel mit `GenerateMac`- und `VerifyMac`-Operationen verwenden, um die Integrität und Authentizität von Daten in AWS KMS zu überprüfen.

HMAC-Algorithmen kombinieren eine kryptografische Hash-Funktion und einen gemeinsamen geheimen Schlüssel. Sie nehmen eine Nachricht und einen geheimen Schlüssel wie das Schlüsselmaterial in einem HMAC-KMS-Schlüssel und geben einen eindeutigen Code mit fester Größe oder Tag zurück. Wenn sich nur ein Zeichen der Nachricht ändert oder wenn der geheime Schlüssel nicht identisch ist, ist das resultierende Tag völlig anders. Durch die Anforderung eines geheimen Schlüssels bietet HMAC auch Authentizität. Ohne den geheimen Schlüssel ist es unmöglich, ein identisches HMAC-Tag zu generieren. HMACs werden manchmal als symmetrische Signaturen bezeichnet, weil sie wie digitale Signaturen funktionieren, aber einen einzigen Schlüssel sowohl für das Signieren als auch für die Überprüfung verwenden.

HMAC-KMS-Schlüssel und die AWS KMS verwendeten HMAC-Algorithmen entsprechen den in RFC 2104 definierten Industriestandards. Der AWS KMS GenerateMacVorgang generiert Standard-HMAC-Tags. HMAC-KMS-Schlüssel werden in AWS KMS Hardware-Sicherheitsmodulen generiert, die gemäß dem FIPS 140-3 Cryptographic Module Validation Program (außer in den Regionen China

(Peking) und China (Ningxia)) zertifiziert sind und niemals unverschlüsselt bleiben. AWS KMS Um einen HMAC-KMS-Schlüssel zu verwenden, müssen Sie AWS KMS aufrufen.

Sie können HMACs verwenden, um die Echtheit einer Nachricht zu ermitteln, z. B. eines JSON-Web-Token (JWT), tokenisierter Kreditkarteninformationen oder eines übermittelten Passworts. Sie können auch als sichere Funktionen zur Schlüsselableitung (KDFs) verwendet werden, insbesondere in Anwendungen, die deterministische Schlüssel benötigen.

HMAC-KMS-Schlüssel bieten einen Vorteil gegenüber Anwendungssoftware, da das Schlüsselmaterial ausschließlich innerhalb HMACs von Software generiert und verwendet wird AWS KMS, vorbehaltlich der Zugriffskontrollen, die Sie für den Schlüssel festlegen.

> **Tip**
>
> Bewährte Methoden empfehlen, die Zeit zu beschränken, in der ein Signiermechanismus, einschließlich eines HMAC, wirksam ist. Dies schreckt einen Angriff ab, bei dem der Akteur eine signierte Nachricht verwendet, um die Gültigkeit wiederholt oder lange nach dem Ersetzen der Nachricht festzustellen. HMAC-Tags enthalten keinen Zeitstempel, aber Sie können einen Zeitstempel in das Token oder die Nachricht aufnehmen, um zu erkennen, wann es Zeit ist, den HMAC zu aktualisieren.

Unterstützte kryptografische Operationen

HMAC-KMS-Schlüssel unterstützen nur die kryptografischen Operationen `GenerateMac` und `VerifyMac`. Sie können keine HMAC-KMS-Schlüssel verwenden, um Daten zu verschlüsseln oder Nachrichten zu signieren, oder eine andere Art von KMS-Schlüssel in HMAC-Operationen zu verwenden. Wenn Sie die `GenerateMac`-Operation verwenden, können Sie eine Meldung von bis zu 4 096 Bytes, einen HMAC-KMS-Schlüssel und den MAC-Algorithmus, der mit der HMAC-Schlüsselspezifikation kompatibel ist, bereitstellen und `GenerateMac` berechnet das HMAC-Tag. Um ein HMAC-Tag zu überprüfen, müssen Sie das HMAC-Tag und dieselbe Nachricht, den HMAC-KMS-Schlüssel und den MAC-Algorithmus angeben, der von `GenerateMac` verwendet wurde, um das ursprüngliche HMAC-Tag zu berechnen. Die `VerifyMac`-Operation berechnet das HMAC-Tag und überprüft, ob es mit dem bereitgestellten HMAC-Tag identisch ist. Wenn die eingegebenen und berechneten HMAC-Tags nicht identisch sind, schlägt die Überprüfung fehl.

HMAC-KMS-Schlüssel unterstützen kein(e) automatische Schlüsseldrehung und Sie können keinen HMAC-KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher erstellen.

Wenn Sie einen KMS-Schlüssel zum Verschlüsseln von Daten in einem AWS Dienst erstellen, verwenden Sie einen symmetrischen Verschlüsselungsschlüssel. Sie können keinen HMAC-KMS-Schlüssel verwenden.

Regionen

HMAC-KMS-Schlüssel werden in allen AWS-Regionen unterstützten Versionen AWS KMS unterstützt.

Weitere Informationen

- Informationen zum Erstellen von HMAC-KMS-Schlüsseln finden Sie unter. Erstellen Sie einen HMAC-KMS-Schlüssel
- Informationen zum Erstellen von HMAC-KMS-Schlüsseln für mehrere Regionen finden Sie unter. Schlüssel für mehrere Regionen eingeben AWS KMS
- Informationen zum Unterschied in der Standardschlüsselrichtlinie, die die AWS KMS Konsole für HMAC-KMS-Schlüssel festlegt, finden Sie unter. the section called “Erlaubt Schlüsselbenutzern die Verwendung eines KMS-Schlüssels für kryptografische Operationen”
- Informationen zum Identifizieren und Anzeigen von HMAC-KMS-Schlüsseln finden Sie unter. Identifizieren Sie HMAC-KMS-Schlüssel
- Weitere Informationen HMACs zur Erstellung von JSON-Web-Tokens finden Sie AWS KMS im AWS Sicherheits-Blog unter How to protect HMACs inside.
- Hören Sie sich einen Podcast: Introducing HMACs for AWS Key Management Service auf The Official AWS Podcast an.

ML-DSA gibt ein AWS KMS

AWS Key Management Service (AWS KMS) unterstützt den Module-Lattice Digital Signature Algorithm (ML-DSA) für kryptografische Signaturen nach dem Quantenverfahren. Diese Implementierung folgt dem Standard Federal Information Processing Standards (FIPS) 204, der zum Schutz vor future Bedrohungen durch Quantencomputer beiträgt. AWS KMS erstellt und schützt alle ML-DSA-Schlüssel und -Signaturoperationen in nach FIPS 140-3 Security Level 3 validierten Hardware-Sicherheitsmodulen. Um ein ausgewogenes Verhältnis zwischen Sicherheit und Leistung zu erreichen, AWS KMS bietet ML-DSA in drei verschiedene Sicherheitsstufen mit unterschiedlichen Schlüsselspezifikationen: ML_DSA_44, ML_DSA_65 und ML_DSA_87.

AWS KMS unterstützt unter Verwendung des Nachrichtentyps `RAW` asymmetrische Schlüsselsignaturen für Nachrichten mit einer Größe von bis zu 4 KB. Für größere Nachrichten müssen Sie die 64-Byte-Nachrichtendarstellung μ, die bei der ML-DSA-Signierung verwendet wird, extern berechnen, wie in NIST FIPS 204 Abschnitt 6.2 definiert. Verwenden Sie den `EXTERNAL_MU` Nachrichtentyp in der Operation AWS KMS Signieren, um diese vorverarbeitete 64-Byte-Nachricht anzugeben. Die durch das extern berechnete μ erzeugten Signaturen sind dieselben wie bei Verwendung derselben `RAW` Nachricht und desselben privaten Schlüssels. Beachten Sie, dass sich diese Signatur von der „Pre-Hash“ -ML-DSA oder HashML-DSA aus Abschnitt 5.4 von NIST FIPS 204 unterscheidet.

Weitere Hinweise zur Verwendung von ML-DSA und dem Nachrichtentyp EXTERNAL_MU finden Sie unter. ML-DSA — wichtige Spezifikationen

Ein Beispiel für die Verwendung von ML-DSA und dem Nachrichtentyp EXTERNAL_MU finden Sie unter. Offline-Überprüfung mit ML-DSA-Schlüsselpaaren

Schlüssel für mehrere Regionen eingeben AWS KMS

AWS KMS unterstützt Schlüssel für mehrere Regionen, AWS-Regionen die unterschiedlich sind und synonym verwendet werden können — als ob Sie denselben Schlüssel AWS KMS keys in mehreren Regionen hätten. Jeder Satz verwandter Schlüssel für mehrere Regionen hat dasselbe Schlüsselmaterial und dieselbe Schlüssel-ID, sodass Sie Daten in einem Schlüssel verschlüsseln AWS-Region und in einem anderen entschlüsseln können, AWS-Region ohne sie erneut zu verschlüsseln oder regionsübergreifend aufzurufen. AWS KMS

Wie alle KMS-Schlüssel bleiben Schlüssel für mehrere Regionen niemals unverschlüsselt. AWS KMS Sie können symmetrische oder asymmetrische multiregionale Schlüssel zur Verschlüsselung oder Signatur erstellen, multiregionale HMAC-Schlüssel zur Erstellung und Überprüfung von HMAC-Tags und multiregionale Schlüssel mit importiertem Schlüsselmaterial erstellen, oder mit von AWS KMS generiertem Schlüsselmaterial. Sie müssen jeden multiregionalen Schlüssel verwalten, unabhängig voneinander, einschließlich der Erstellung von Aliasen und Tags und der Festlegung ihrer wichtigsten Richtlinien und Berechtigungen sowie der selektiven Aktivierung und Deaktivierung. Sie können multiregionale Schlüssel in allen kryptografischen Operationen verwenden, die Sie mit einzelregionalen Schlüsseln ausführen können.

Multiregionale Schlüssel sind eine flexible und leistungsstarke Lösung für viele gängige Datensicherheitsszenarien.

Notfallwiederherstellung

In einer Sicherungs- und Wiederherstellungsarchitektur können Sie mit Schlüsseln für mehrere Regionen verschlüsselte Daten auch bei einem Ausfall unterbrechungsfrei verarbeiten. AWS-Region Daten, die in Backup-Regionen verwaltet werden, können im Backup-Bereich entschlüsselt werden, und Daten, die im Backup-Bereich neu verschlüsselt wurden, können in der primären Region entschlüsselt werden, wenn diese Region wiederhergestellt wird.

Globales Datenmanagement

Unternehmen, die global tätig sind, benötigen global verteilte Daten, die konsistent über AWS-Regionen verfügbar sind. Sie können multiregionale Schlüssel in allen Regionen erstellen, in denen sich Ihre Daten befinden. Anschließend können Sie die Schlüssel so verwenden, als wären sie einzelregionale Schlüssel, ohne die Latenz eines regionsübergreifenden Anrufs oder die Kosten für die erneute Verschlüsselung von Daten unter einem anderen Schlüssel in jeder Region.

Verteilte Signaturanwendungen

Anwendungen, die regionsübergreifende Signaturfunktionen erfordern, können asymmetrische Signaturschlüssel für mehrere Regionen verwenden, um identische digitale Signaturen konsistent und wiederholt in verschiedenen AWS-Regionen zu generieren.

Wenn Sie die Zertifikatsverkettung mit einem einzigen globalen Vertrauensspeicher (für eine einzelne Stammzertifizierungsstelle (CA)) und einem von der Stammzertifizierungsstelle CAs signierten Regional Intermediate verwenden, benötigen Sie keine Schlüssel für mehrere Regionen. Wenn Ihr System jedoch keine Zwischenzertifizierungen wie Anwendungssignaturen unterstützt CAs, können Sie Schlüssel für mehrere Regionen verwenden, um regionale Zertifizierungen einheitlich zu gestalten.

Aktiv/aktiv-Anwendungen, die sich über mehrere Regionen erstrecken

Einige Arbeitslasten und Anwendungen können sich über mehrere Regionen in aktiv/aktiv-Architekturen erstrecken. Für diese Anwendungen können multiregionale Schlüssel die Komplexität reduzieren, indem sie dasselbe Schlüsselmaterial für gleichzeitige Verschlüsselungs- und Entschlüsselungsoperationen für Daten bereitstellen, die möglicherweise über Regionsgrenzen hinweg verschoben werden.

Sie können Schlüssel für mehrere Regionen mit clientseitigen Verschlüsselungsbibliotheken wie dem AWS Encryption SDKAWSDatabase Encryption SDK und der clientseitigen Amazon S3 S3-Verschlüsselung verwenden.

AWS Dienste, die AWS KMS für Verschlüsselung im Ruhezustand oder digitale Signaturen integriert sind, behandeln Schlüssel mit mehreren Regionen derzeit so, als wären sie Schlüssel mit nur einer Region. Sie können Daten, die zwischen Regionen verschoben wurden, erneut verpacken oder neu verschlüsseln. Beispielsweise entschlüsselt und verschlüsselt die regionsübergreifende Replikation von Simple Storage Service (Amazon S3) Daten unter einem KMS-Schlüssel in der Zielregion, selbst wenn Objekte repliziert werden, die durch einen multiregionalen Schlüssel geschützt sind.

Multiregionale Schlüssel sind nicht global. Sie erstellen einen multiregionalen Primärschlüssel und replizieren ihn dann in Regionen, die Sie in einer AWS -Partition auswählen. Anschließend verwalten Sie den multiregionalen Schlüssel in jeder Region unabhängig. AWS Weder erstellt noch AWS KMS jemals automatisch Schlüssel für mehrere Regionen in Ihrem Namen oder repliziert sie in eine Region. Von AWS verwaltete Schlüssel, bei den KMS-Schlüsseln, die AWS Dienste in Ihrem Konto für Sie erstellen, handelt es sich immer um Schlüssel für einzelne Regionen.

In chinesischen Regionen können Sie die Schlüsselfunktion für mehrere Regionen verwenden, um KMS-Schlüssel innerhalb der Partition China Regions () `aws-cn` zu replizieren. Sie können beispielsweise einen Schlüssel von der Region China (Peking) in die Region China (Ningxia) replizieren oder umgekehrt. Durch die Replikation eines Schlüssels von einer China Region in eine andere erklären Sie sich damit einverstanden, den Schlüssel AWS Key Management Service der Zielregion zu verwenden und alle für die Zielregion geltenden Vertragsbedingungen einzuhalten. Sie können einen Schlüssel aus den Regionen Peking und Ningxia nicht in eine AWS Region außerhalb der Partition China Regions replizieren. Ebenso können Sie einen Schlüssel aus einer Region außerhalb der Partition China Regions nicht in die Regionen Peking und Ningxia replizieren.

Sie können einen vorhandenen einzelregionalen Schlüssel nicht in einen multiregionalen Schlüssel konvertieren. Dieses Merkmal stellt sicher, dass alle Daten, die mit vorhandenen einzelregionalen Schlüsseln geschützt sind, dieselben Datenresidenz- und Datensouveränitäts-Eigenschaften beibehalten.

Für die meisten Datensicherheitsanforderungen sind Standardschlüssel für AWS KMS einzelne Regionen aufgrund der regionalen Isolierung und Fehlertoleranz regionaler Ressourcen die am besten geeignete Lösung. Wenn Sie jedoch Daten in clientseitigen Anwendungen über mehrere Regionen hinweg verschlüsseln oder signieren müssen, sind multiregionale Schlüssel möglicherweise die Lösung.

Regionen

Schlüssel für mehrere Regionen werden in allem AWS-Regionen unterstützt, was unterstützt wird. AWS KMS

Preise und Kontingente

Jeder Schlüssel in einem Satz verwandter multiregionaler Schlüssel zählt als ein KMS-Schlüssel für Preise und Kontingente. AWS KMS -Kontingente werden separat für jede Region eines Kontos berechnet. Die Verwendung und Verwaltung der multiregionalen Schlüssel in jeder Region zählen zu den Kontingenten für diese Region.

Unterstützte KMS-Schlüsseltypen

Sie können die folgenden Typen von für mehrere Regionen geltenden KMS-Schlüsseln erstellen:

- KMS-Schlüssel zur symmetrischen Verschlüsselung
- Asymmetrische KMS-Schlüssel
- HMAC-KMS-Schlüssel
- KMS-Schlüssel mit importiertem Schlüsselmaterial

Sie können keine multiregionale Schlüssel in einem benutzerdefinierten Schlüsselspeicher verwenden.

Weitere Informationen

- Informationen zur Steuerung des Zugriffs auf KMS-Schlüssel mit mehreren Regionen finden Sie unter. Steuern Sie den Zugriff auf Schlüssel für mehrere Regionen
- Informationen zum Erstellen von primären KMS-Schlüsseln für mehrere Regionen eines beliebigen Typs finden Sie unter. Primärschlüssel für mehrere Regionen erstellen
- Informationen zum Erstellen von Replikat-KMS-Schlüsseln für mehrere Regionen finden Sie unter. Erstellen Sie Replikatschlüssel für mehrere Regionen
- Informationen zum Aktualisieren der primären Region finden Sie unter. Ändern Sie den Primärschlüssel in einer Reihe von Schlüsseln für mehrere Regionen
- Informationen zum Identifizieren und Anzeigen von KMS-Schlüsseln für mehrere Regionen finden Sie unterIdentifizieren Sie HMAC-KMS-Schlüssel.
- Informationen zu besonderen Überlegungen beim Löschen von KMS-Schlüsseln für mehrere Regionen finden Sie unter. Deleting multi-Region keys

Terminologie und Konzepte

Die folgenden Begriffe und Konzepte werden mit multiregionalen Schlüsseln verwendet.

Multiregionaler Schlüssel

Ein multiregionaler Schlüssel ist einer von einer Reihe von KMS-Schlüsseln mit derselben Schlüssel-ID und demselben Schlüsselmaterial (und anderen gemeinsam genutzten Eigenschaften) in verschiedenen AWS-Regionen. Jeder multiregionale Schlüssel ist ein voll funktionsfähiger KMS-Schlüssel, der vollständig unabhängig von seinen verwandten multiregionalen Schlüsseln verwendet werden kann. Da alle zugehörigen Schlüssel für mehrere Regionen dieselbe Schlüssel-ID und dasselbe Schlüsselmaterial haben, sind sie interoperabel, d. h. jeder zugehörige Schlüssel für mehrere Regionen AWS-Region kann Chiffretext entschlüsseln, der mit einem anderen zugehörigen Schlüssel für mehrere Regionen verschlüsselt wurde.

Sie legen die multiregionale Eigenschaft eines KMS-Schlüssels fest, wenn Sie ihn erstellen. Sie können die multiregionale Eigenschaft für einen vorhandenen Schlüssel nicht ändern. Einzelregionale Schlüssel können nicht in multiregionale Schlüssel konvertiert werden oder umgekehrt. Um vorhandene Workloads in multiregionale Szenarien zu verschieben, müssen Sie die Daten erneut verschlüsseln oder neue Signaturen mit neuen multiregionalen Schlüsseln erstellen.

Ein Schlüssel mit mehreren Regionen kann symmetrisch oder asymmetrisch sein und Schlüsselmaterial oder importiertes Schlüsselmaterial verwenden. AWS KMS Sie können keine multiregionale Schlüssel in einem benutzerdefinierten Schlüsselspeicher verwenden.

In einer Reihe von verwandten multiregionalen Schlüsseln gibt es zu jeder Zeit genau einen Primärschlüssel. Sie können Replikatschlüsseln dieses Primärschlüssels in anderen AWS-Regionen erstellen. Sie können auch die primäre Region aktualisieren, wodurch der Primärschlüssel in einen Replikatschlüssel und ein angegebener Replikatschlüssel in den Primärschlüssel geändert wird. Sie können jedoch jeweils nur einen Primärschlüssel oder Replikatschlüssel verwalten. AWS-Region Alle Regionen müssen sich in derselben AWS -Partition befinden.

Sie können mehrere Sätze von verwandten multiregionalen Schlüsseln in der gleichen oder in unterschiedlichen AWS-Regionen haben. Obwohl verwandte multiregionale Schlüssel interoperabel sind, sind unverwandte multiregionale Schlüssel nicht interoperabel.

Primärschlüssel

Ein Primärschlüssel mit mehreren Regionen ist ein KMS-Schlüssel, der in andere AWS-Regionen in derselben Partition repliziert werden kann. Jeder Satz von multiregionalen Schlüsseln hat nur einen Primärschlüssel.

Ein Primärschlüssel unterscheidet sich von einem Replikatschlüssel wie folgt:

- Nur ein Primärschlüssel kann repliziert werden.
- Der Primärschlüssel ist die Quelle für die gemeinsam genutzten Eigenschaften seiner Replikatschlüssel, einschließlich Schlüsselmaterial und Schlüssel-ID.
- Sie können die automatische Schlüsseldrehung nur für einen Primärschlüssel aktivieren oder deaktivieren.
- Sie können jederzeit das Löschen eines Primärschlüssels planen. Ein Primärschlüssel AWS KMS wird jedoch erst gelöscht, wenn alle seine Replikatschlüssel gelöscht sind.

Primär- und Replikatschlüssel unterscheiden sich jedoch nicht in kryptografischen Eigenschaften. Sie können einen Primärschlüssel und seine Replikatschlüssel austauschbar verwenden.

Sie müssen keinen Primärschlüssel replizieren. Sie können ihn genauso verwenden wie jeden KMS-Schlüssel und dabb replizieren, wenn es nützlich ist. Da multiregionale Schlüssel jedoch andere Sicherheitseigenschaften haben als einzelregionale Schlüssel, empfiehlt es sich, einen multiregionalen Schlüssel nur dann zu erstellen, wenn Sie ihn replizieren möchten.

Replikat-Schlüssel

Ein Replikatschlüssel mit mehreren Regionen ist ein KMS-Schlüssel, der dieselbe Schlüssel-ID und dasselbe Schlüsselmaterial wie sein Primärschlüssel und die zugehörigen Replikatschlüssel hat, der jedoch in einem anderen vorhanden ist. AWS-Region

Ein Replikatschlüssel ist ein voll funktionsfähiger KMS-Schlüssel mit eigener Schlüsselrichtlinie und den eigenen Erteilungen, Aliasen, Tags und anderen Eigenschaften. Es handelt sich nicht um eine Kopie oder einen Zeiger auf den Primärschlüssel oder einen sonstigen Schlüssel. Sie können einen Replikatschlüssel auch dann verwenden, wenn sein Primärschlüssel und alle verwandte Replikatschlüssel deaktiviert sind. Sie können auch einen Replikatschlüssel in einen Primärschlüssel und einen Primärschlüssel in einen Replikatschlüssel konvertieren. Sobald er erstellt wurde, stützt sich ein Replikatschlüssel auf seinen Primärschlüssel nur zur Schlüsseldrehung und zum Aktualisieren der primären Region.

Primär- und Replikatschlüssel unterscheiden sich nicht in kryptografischen Eigenschaften. Sie können einen Primärschlüssel und seine Replikatschlüssel austauschbar verwenden. Daten, die durch einen Primär- oder Replikatschlüssel verschlüsselt werden, können durch denselben Schlüssel oder durch einen beliebigen verwandten Primär- oder Replikatschlüssel entschlüsselt werden.

Replicate

Sie können einen Primärschlüssel mit mehreren Regionen in derselben Partition in einen anderen AWS-Region replizieren. Wenn Sie dies tun, AWS KMS wird in der angegebenen Region ein Replikatschlüssel für mehrere Regionen mit derselben Schlüssel-ID und anderen gemeinsamen Eigenschaften wie der Primärschlüssel erstellt. Dann transportiert es das Schlüsselmaterial sicher über die Regionsgrenze und ordnet es dem neuen Replikatschlüssel zu, alles innerhalb von AWS KMS.

Freigegebene Eigenschaften

Gemeinsam genutzte Eigenschaften sind Eigenschaften eines Primärschlüssels mit mehreren Regionen, die mit seinen Replikatschlüsseln gemeinsam genutzt werden. AWS KMS erstellt die Replikatschlüssel mit denselben gemeinsamen Eigenschaftswerten wie die des Primärschlüssels. Anschließend synchronisiert es regelmäßig die gemeinsamen Eigenschaftswerte des Primärschlüssels mit seinen Replikatschlüsseln. Sie können diese Eigenschaften nicht für einen Replikatschlüssel festlegen.

Im Folgenden sind die gemeinsamen Eigenschaften von multiregionalen Schlüsseln aufgeführt.

- Schlüssel-ID – (Das `Region`-Element des Schüssel-ARN unterscheidet sich.)
- Schlüsselmaterial
- Ursprung des Schlüsselmaterials
- Schlüsselspezifikation und Verschlüsselungsalgorithmen
- Schlüsselnutzung
- Automatische Schlüsseldrehung – Sie können die automatische Schlüsseldrehung nur für den Primärschlüssel aktivieren oder deaktivieren. Neue Replikatschlüssel werden mit allen Versionen des freigegebenen Schlüsselmaterials erstellt. Details hierzu finden Sie unter Rotating multi-Region keys.
- Rotation auf Anforderung — Sie können die Rotation auf Anforderung nur für den Primärschlüssel durchführen. Neue Replikatschlüssel werden mit allen Versionen des freigegebenen Schlüsselmaterials erstellt. Details hierzu finden Sie unter Rotating multi-Region keys.

Sie können sich auch die Primär- und Replikatbezeichnungen verwandter Mehrbereichsschlüssel als gemeinsame Eigenschaften vorstellen. Wenn Sie neue Replikatschlüssel erstellen oder den Primärschlüssel aktualisieren, wird die Änderung mit allen zugehörigen Multiregions-Schlüsseln

AWS KMS synchronisiert. Wenn diese Änderungen abgeschlossen sind, werden alle verwandte multiregionale Schlüssel ihren Primärschlüssel und die Replikatschlüssel korrekt auflisten.

Alle anderen Eigenschaften von multiregionalen Schlüsseln sind unabhängige Eigenschaften, einschließlich der Beschreibung, Schlüsselrichtlinie, Erteilungen, aktivierten und deaktivierten Schlüsselzustände, Aliasen und Tags. Sie können dieselben Werte für diese Eigenschaften für alle verwandte mutliregionale Schlüssel festlegen. Wenn Sie jedoch den Wert einer unabhängigen Eigenschaft ändern, wird dies von AWS KMS nicht synchronisiert.

Sie können die Synchronisierung der freigegebenen Eigenschaften Ihrer multiregionalen Schlüssel verfolgen. Suchen Sie in Ihrem AWS CloudTrail Protokoll nach dem Ereignis. SynchronizeMultiRegionKey

Sicherheitsaspekte für multiregionale Schlüssel

Verwenden Sie einen Schlüssel für AWS KMS mehrere Regionen nur, wenn Sie einen benötigen. Multiregionale Schlüssel bieten eine flexible und skalierbare Lösung für Workloads, die verschlüsselte Daten zwischen AWS-Regionen verschieben oder regionsübergreifenden Zugang benötigen. Verwenden Sie einen multiregionalen Schlüssel, wenn Sie geschützte Daten über Regionen hinweg freigeben, verschieben oder sichern müssen oder identische digitale Signaturen von Anwendungen erstellen müssen, die in verschiedenen Regionen ausgeführt werden.

Beim Erstellen eines multiregionalen Schlüssels wird das Schlüsselmaterial jedoch innerhalb AWS KMSüber die Grenzen der AWS-Region verschoben. Der Chiffretext, der von einem multiregionalen Schlüssel generiert wird, kann möglicherweise von mehreren verwandten Schlüsseln an mehreren geografischen Standorten entschlüsselt werden. Regional isolierte Services und Ressourcen bieten auch erhebliche Vorteile. Jede AWS-Region ist isoliert und unabhängig von den anderen Regionen. Regionen bieten Fehlertoleranz, Stabilität und Ausfallsicherheit und können auch die Latenz verkürzen. Sie ermöglichen das Erstellen redundanter Ressourcen, die verfügbar bleiben und von einem Ausfall in einer anderen Region nicht betroffen sind. Außerdem stellen sie sicher AWS KMS, dass jeder Chiffretext mit nur einem Schlüssel entschlüsselt werden kann.

Multiregionale Schlüssel werfen auch neue Sicherheitsbedenken auf.

- Die Steuerung des Zugriffs und die Durchsetzung von Datensicherheitsrichtlinien ist mit multiregionalen Schlüsseln komplexer. Sie müssen sicherstellen, dass die Richtlinie bei Schlüsseln in mehreren isolierten Regionen konsistent überwacht wird. Und Sie müssen Richtlinien verwenden, um Grenzen zu erzwingen, anstatt sich auf separate Schlüssel zu verlassen.

Sie müssen beispielsweise Richtlinienbedingungen für Daten festlegen, um zu verhindern, dass Lohn- und Gehaltsabrechnungsteams in einer Region Lohn- und Gehaltsabrechnungsdaten für eine andere Region lesen können. Außerdem müssen Sie die Zugriffssteuerung verwenden, um ein Szenario zu verhindern, in dem ein multiregionaler Schlüssel in einer Region die Daten eines Mandanten schützt und ein verwandter multiregionaler Schlüssel in einer anderen Region die Daten eines anderen Mandanten schützt.

- Die Prüfung von Schlüsseln über Regionen hinweg ist ebenfalls komplexer. Mit multiregionalen Schlüsseln müssen Sie Prüfungsaktivitäten über mehrere Regionen hinweg überprüfen und abstimmen, um ein vollständiges Verständnis der wichtigsten Aktivitäten für geschützte Daten zu erhalten.
- Die Einhaltung von Datenresidenz-Mandaten kann komplexer sein. Mit isolierten Regionen können Sie die Einhaltung von Datenresidenz und Datenhoheit sicherstellen. KMS-Schlüssel in einer bestimmten Region können sensible Daten nur in dieser Region entschlüsseln. Daten, die in einer Region verschlüsselt sind, können in jeder anderen Region vollständig geschützt und unzugänglich bleiben.

  Um die Datenresidenz und Datensouveränität mit Schlüsseln für mehrere Regionen zu überprüfen, müssen Sie Zugriffsrichtlinien implementieren und Ereignisse für mehrere Regionen zusammenstellen AWS CloudTrail .

Um Ihnen die Verwaltung der Zugriffskontrolle für Schlüssel mit mehreren Regionen zu erleichtern, ist die Berechtigung zum Replizieren eines Schlüssels für mehrere Regionen (kms: ReplicateKey) von der Standardberechtigung zum Erstellen von Schlüsseln (kms:) getrennt. CreateKey AWS KMS Unterstützt außerdem mehrere Richtlinienbedingungen für Schlüssel mit mehreren Regionen, z. B. die Erlaubnis`kms:MultiRegion`, Schlüssel für mehrere Regionen zu erstellen, zu verwenden oder zu verwalten, oder die die Regionen einschränkt`kms:ReplicaRegion`, in die ein Schlüssel für mehrere Regionen repliziert werden kann. Details hierzu finden Sie unter Steuern Sie den Zugriff auf Schlüssel für mehrere Regionen.

Funktionsweise von multiregionalen Schlüsseln

Sie beginnen mit der Erstellung eines symmetrischen oder asymmetrischen Primärschlüssels für mehrere Regionen in einem System, AWS-Region das dies AWS KMS unterstützt, z. B. US East (Nord-Virginia). Sie entscheiden nur, wenn Sie ihn erstellen, ob es sich bei einem Schlüssel um einen einzelregionalen Schlüssel oder um einen multiregionalen Schlüssel handelt. Sie können diese Eigenschaft später nicht ändern. Wie bei jedem KMS-Schlüssel, legen Sie eine Schlüsselrichtlinie für

den multiregionalen Schlüssel fest, und Sie können Berechtigungen erstellen und Aliase und Tags für die Kategorisierung und Autorisierung hinzufügen. (Diese sind unabhängige Eigenschaften, die nicht gemeinsam genutzt oder mit anderen Schlüsseln synchronisiert werden.) Sie können Ihren multiregionalen Primärschlüssel in kryptografischen Operationen zur Verschlüsselung oder Signatur verwenden.

Sie können einen Primärschlüssel für mehrere Regionen in der AWS KMS Konsole oder mithilfe der CreateKeyAPI erstellen, wobei der Parameter auf gesetzt ist`MultiRegion`. `true` Beachten Sie, dass multiregionale Schlüssel eine unverwechselbare Schlüssel-ID haben, die mit `mrk-` beginnt. Sie können das `mrk-` Präfix zur MRKs programmgesteuerten Identifizierung verwenden.

Wenn Sie möchten, können Sie den Primärschlüssel für mehrere Regionen in einen oder mehrere verschiedene Primärschlüssel AWS-Regionen in derselben AWS Partition replizieren, z. B. Europa (Irland). Wenn Sie dies tun, AWS KMS wird in der angegebenen Region ein Replikatschlüssel mit derselben Schlüssel-ID und anderen gemeinsamen Eigenschaften wie der Primärschlüssel erstellt. Dann transportiert es das Schlüsselmaterial sicher über die Regionsgrenze und ordnet es dem neuen KMS-Schlüssel in der Zielregion zu, alles innerhalb von AWS KMS. Das Ergebnis sind zwei verwandte multiregionale Schlüssel – ein Primärschlüssel und ein Replikatschlüssel – die austauschbar verwendet werden können.

Sie können einen Replikatschlüssel für mehrere Regionen in der AWS KMS Konsole oder mithilfe der API erstellen. ReplicateKey

Der resultierende multiregionale Replikatschlüssel ist ein voll funktionsfähiger KMS-Schlüssel mit denselben gemeinsam genutzten Eigenschaften wie der Primärschlüssel. In jeder anderen Hinsicht ist es ein unabhängiger KMS-Schlüssel mit eigener Beschreibung und Schlüsselrichtlinie und den eigenen Erteilungen, Aliasen und Tags. Das Aktivieren oder Deaktivieren eines multiregionalen Schlüssels hat keine Auswirkungen auf verwandte multiregionale Schlüssel. Sie können Primär- und Replikatschlüssel unabhängig von einander in kryptografischen Operationen verwenden oder deren Verwendung koordinieren. Beispielsweise können Sie Daten mit dem Primärschlüssel in der Region USA Ost (Nord-Virginia) verschlüsseln, die Daten in die Region Europa (Irland) verschieben und die Daten mit dem Replikatschlüssel entschlüsseln.

Verwandte multiregionale Schlüssel haben dieselbe Schlüssel-ID. Ihr Schlüssel ARNs (Amazon-Ressourcennamen) unterscheidet sich nur im Feld Region. Beispielsweise könnten der Primärschlüssel und die Replikatschlüssel für mehrere Regionen den folgenden Beispielschlüssel haben. ARNs Die Schlüssel-ID – das letzte Element im Schlüssel ARN – ist identisch. Beide

Schlüssel haben die unverwechselbare Schlüssel-ID von multiregionalen Schlüsseln, die mit mrk- beginnt.

```
Primary key: arn:aws:kms:us-
east-1:111122223333:key/mrk-1234abcd12ab34cd56ef12345678990ab
Replica key: arn:aws:kms:eu-
west-1:111122223333:key/mrk-1234abcd12ab34cd56ef12345678990ab
```

Für die Interoperabilität ist dieselbe Schlüssel-ID erforderlich. AWS KMS Bindet beim Verschlüsseln die Schlüssel-ID des KMS-Schlüssels an den Chiffretext, sodass der Chiffretext nur mit diesem KMS-Schlüssel oder einem KMS-Schlüssel mit derselben Schlüssel-ID entschlüsselt werden kann. Diese Funktion macht auch verwandte multiregionale Schlüssel leicht zu erkennen und erleichtert die austauschbare Verwendung. Wenn Sie sie beispielsweise in einer Anwendung verwenden, können Sie auf verwandte multiregionale Schlüssel anhand ihrer geteilten Schlüssel-ID verweisen. Geben Sie dann, falls erforderlich, die Region oder den ARN an, um sie zu unterscheiden.

Wenn sich Ihre Datenanforderungen ändern, können Sie den Primärschlüssel auf andere AWS-Regionen in derselben Partition replizieren, z. B. USA West (Oregon) und Asien-Pazifik (Sydney). Das Ergebnis sind vier zugehörige Schlüssel für mehrere Regionen mit demselben Schlüsselmaterial und demselben Schlüssel IDs, wie in der folgenden Abbildung dargestellt. Sie verwalten die Schlüssel unabhängig voneinander. Sie können sie unabhängig voneinander oder koordiniert verwenden. Beispielsweise können Sie Daten mit dem Replikatschlüssel in Asien-Pazifik (Sydney) verschlüsseln, die Daten nach USA West (Oregon) verschieben und mit dem Replikatschlüssel in USA West (Oregon) entschlüsseln.

Weitere Überlegungen für multiregionalen Schlüssel sind die folgenden:

Synchronisieren gemeinsam genutzter Eigenschaften — Wenn sich eine gemeinsame Eigenschaft der Schlüssel für mehrere Regionen ändert, AWS KMS wird die Änderung vom Primärschlüssel automatisch auf alle zugehörigen Replikatschlüssel synchronisiert. Sie können eine Synchronisation von gemeinsam genutzten Eigenschaften nicht anfordern oder erzwingen. AWS KMS erkennt und synchronisiert alle Änderungen für Sie. Sie können die Synchronisation jedoch überprüfen, indem Sie das SynchronizeMultiRegionKeyEreignis in den CloudTrail Protokollen verwenden.

Wenn Sie beispielsweise die automatische Schlüsselrotation für einen symmetrischen Primärschlüssel mit mehreren Regionen aktivieren, wird diese Einstellung auf alle zugehörigen Replikatschlüssel AWS KMS kopiert. Wenn das Schlüsselmaterial gedreht wird, wird die Drehung zwischen allen verwandten multiregionalen Schlüsseln synchronisiert, sodass sie weiterhin dasselbe

aktuelle Schlüsselmaterial haben und Zugriff auf alle älteren Versionen des Schlüsselmaterials haben. Wenn Sie einen neuen Replikatschlüssel erstellen, verfügt er über dasselbe aktuelle Schlüsselmaterial aller verwandten multiregionalen Schlüssel und Zugriff auf alle vorherigen Versionen des Schlüsselmaterials. Details hierzu finden Sie unter Rotating multi-Region keys.

Ändern des Primärschlüssels – Jeder Satz von multiregionalen Schlüsseln muss genau einen Primärschlüssel haben. Der Primärschlüssel ist der einzige Schlüssel, der repliziert werden kann. Er ist auch die Quelle der gemeinsamen Eigenschaften seiner Replikatschlüssel. Sie können jedoch den Primärschlüssel in ein Replikat ändern und einen der Replikatschlüssel als Primärschlüssel heraufstufen. Sie können dies tun, damit Sie einen multiregionalen Primärschlüssel aus einer bestimmten Region löschen oder den Primärschlüssel in einer Region näher an den Projektadministratoren platzieren können. Details hierzu finden Sie unter Ändern Sie den Primärschlüssel in einer Reihe von Schlüsseln für mehrere Regionen.

Löschen von Schlüsseln mit mehreren Regionen — Wie bei allen KMS-Schlüsseln müssen Sie das Löschen von Schlüsseln mit mehreren Regionen planen, bevor Sie sie löschen. AWS KMS Solange der Schlüssel zur Löschung ansteht, können Sie ihn nicht in kryptografischen Operationen verwenden. Ein Primärschlüssel mit mehreren Regionen AWS KMS wird jedoch erst gelöscht, wenn alle zugehörigen Replikatschlüssel gelöscht wurden. Details hierzu finden Sie unter Deleting multi-Region keys.

Schlüsselmaterial für AWS KMS Schlüssel importieren

Sie können einen AWS KMS keys (KMS-Schlüssel) mit den von Ihnen bereitgestellten Schlüsselmaterialien erstellen.

Ein KMS-Schlüssel ist eine logische Darstellung eines Datenschlüssels. Die Metadaten für einen KMS-Schlüssel enthalten die ID des Schlüsselmaterials, das zur Ausführung kryptografischer Operationen verwendet wird. Wenn Sie einen KMS-Schlüssel erstellen, wird standardmäßig das Schlüsselmaterial für diesen KMS-Schlüssel AWS KMS generiert. Aber Sie können einen KMS-Schlüssel ohne Schlüsselmaterial erstellen und dann Ihr eigenes Schlüsselmaterial in diesen KMS-Schlüssel importieren, ein Feature, das oft als Bring Your Own Key (BYOK, bringe deinen eigenen Schlüssel) bezeichnet wird.

> **Note**
>
> AWS KMS unterstützt nicht die Entschlüsselung von AWS KMS Chiffretext, der mit einem KMS-Schlüssel mit symmetrischer Verschlüsselung außerhalb von verschlüsselt wurde

AWS KMS, selbst wenn der Chiffretext unter einem KMS-Schlüssel mit importiertem Schlüsselmaterial verschlüsselt wurde. AWS KMS veröffentlicht das für diese Aufgabe erforderliche Chiffretextformat nicht, und das Format kann sich ohne vorherige Ankündigung ändern.

Wenn Sie importiertes Schlüsselmaterial verwenden, bleiben Sie für das Schlüsselmaterial verantwortlich und erlauben gleichzeitig, eine Kopie davon AWS KMS zu verwenden. Sie können dies aus den folgenden Gründe tun:

- Um nachzuweisen, dass das Schlüsselmaterial mit einer zufälligen Quelle generiert wurde, die Ihre Anforderungen erfüllt.
- Um Schlüsselmaterial aus Ihrer eigenen Infrastruktur mit AWS Services AWS KMS zu verwenden und den Lebenszyklus des darin enthaltenen Schlüsselmaterials zu verwalten AWS.
- Zur Verwendung vorhandener, etablierter Schlüssel AWS KMS, z. B. Schlüssel für Codesignatur, PKI-Zertifikatsignierung und per Zertifikat fixierte Anwendungen
- Um eine Ablaufzeit für das Schlüsselmaterial festzulegen AWS und es manuell zu löschen, es aber auch in future wieder verfügbar zu machen. Im Gegensatz dazu erfordert die Planung der Schlüssellöschung eine Wartezeit von 7 bis 30 Tagen. Danach können Sie den gelöschten KMS-Schlüssel nicht wiederherstellen.
- Die Originalkopie des Schlüsselmaterials zu besitzen und es AWS für zusätzliche Haltbarkeit und Notfallwiederherstellung während des gesamten Lebenszyklus des Schlüsselmaterials außerhalb des Betriebs aufzubewahren.
- Bei asymmetrischen Schlüsseln und HMAC-Schlüsseln werden beim Import kompatible und interoperable Schlüssel erstellt, die innerhalb und außerhalb von funktionieren. AWS

Unterstützte KMS-Schlüsseltypen

AWS KMS unterstützt importiertes Schlüsselmaterial für die folgenden Typen von KMS-Schlüsseln. Es ist jedoch nicht möglich, Schlüsselmaterial in KMS-Schlüsseln in benutzerdefinierten Schlüsselspeichern zu importieren.

- KMS-Schlüssel zur symmetrischen Verschlüsselung
- Asymmetrische KMS-Schlüssel (außer ML-DSA-Schlüssel)
- HMAC-KMS-Schlüssel

- Schlüssel für mehrere Regionen aller unterstützten Typen.

Regionen

Importiertes Schlüsselmaterial wird in allem AWS-Regionen unterstützt, was unterstützt wird. AWS KMS

In China Regionen unterscheiden sich die wichtigsten Materialanforderungen für symmetrische Verschlüsselung von KMS-Schlüsseln von denen in anderen Regionen. Details hierzu finden Sie unter Schritt 3: Verschlüsselung des Schlüsselmaterials.

Weitere Informationen

- Informationen zum Erstellen von KMS-Schlüsseln mit importiertem Schlüsselmaterial finden Sie unterErstellen Sie einen KMS-Schlüssel mit importiertem Schlüsselmaterial.
- Informationen zum Erstellen eines Alarms, der Sie benachrichtigt, wenn das importierte Schlüsselmaterial in einem KMS-Schlüssel bald abläuft, finden Sie unterEinen CloudWatch Alarm für den Ablauf von importiertem Schlüsselmaterial erstellen.
- Informationen zum erneuten Import von Schlüsselmaterial in einen KMS-Schlüssel finden Sie unter. Schlüsselmaterial erneut importieren
- Informationen zum Importieren von neuem Schlüsselmaterial in einen KMS-Schlüssel für die Rotation bei Bedarf finden Sie unter Neues Schlüsselmaterial importieren undFühren Sie die Schlüsselrotation bei Bedarf durch.
- Informationen zur Identifizierung und Anzeige von KMS-Schlüsseln mit importiertem Schlüsselmaterial finden Sie unterIdentifizieren Sie KMS-Schlüssel mit importiertem Schlüsselmaterial.
- Informationen zu besonderen Überlegungen beim Löschen von KMS-Schlüsseln mit importiertem Schlüsselmaterial finden Sie unterDeleting KMS keys with imported key material.

Besondere Überlegungen für importiertes Schlüsselmaterial

Bevor Sie sich für den Import von Schlüsselmaterial in entscheiden AWS KMS, sollten Sie sich mit den folgenden Eigenschaften von importiertem Schlüsselmaterial vertraut machen.

Sie generieren das Schlüsselmaterial

Sie sind verantwortlich für die Generierung des Schlüsselmaterials mit einer zufälligen Quelle, die Ihre Anforderungen erfüllt.

Sie sind verantwortlich für Verfügbarkeit und Langlebigkeit

AWS KMS wurde entwickelt, um importiertes Schlüsselmaterial hochverfügbar zu halten. Die Haltbarkeit von importiertem Schlüsselmaterial wird jedoch AWS KMS nicht auf dem gleichen Niveau gehalten wie das AWS KMS generierte Schlüsselmaterial. Details hierzu finden Sie unter Schützen von importiertem Schlüsselmaterial.

Sie können das Schlüsselmaterial löschen.

Sie können importiertes Schlüsselmaterial aus einem KMS-Schlüssel löschen, wodurch der KMS-Schlüssel sofort unbrauchbar wird. Beim Importieren von Schlüsselmaterial in einen KMS-Schlüssel können Sie außerdem bestimmen, ob der Schlüssel abläuft und die Ablaufzeit festlegen. Wenn die Ablaufzeit erreicht ist, AWS KMS wird das Schlüsselmaterial gelöscht. Ohne Schlüsselmaterial kann der KMS-Schlüssel nicht in kryptografischen Operationen verwendet werden. Um den Schlüssel wiederherzustellen, müssen Sie das gleiche Schlüsselmaterial erneut in den Schlüssel importieren.

Sie können das Schlüsselmaterial für asymmetrische Schlüssel, HMAC-Schlüssel und Schlüssel mit mehreren Regionen nicht ändern

Beim Importieren von Schlüsselmaterial in einen KMS-Schlüssel wird der KMS-Schlüssel dauerhaft diesem Schlüsselmaterial zugeordnet. Sie können dasselbe Schlüsselmaterial erneut importieren, aber kein anderes Schlüsselmaterial in diesen KMS-Schlüssel importieren. Außerdem ist es nicht möglich, die automatische Schlüsseldrehung für einen KMS-Schlüssel mit importiertem Schlüsselmaterial zu aktivieren. Sie können einen KMS-Schlüssel mit importiertem Schlüsselmaterial jedoch manuell drehen.

Sie können bei Bedarf eine Rotation für symmetrische Verschlüsselungsschlüssel mit nur einer Region durchführen

Symmetrische Verschlüsselungsschlüssel für einzelne Regionen mit importiertem Schlüsselmaterial unterstützen die Rotation nach Bedarf. Sie können mehrere Schlüsselmaterialien in diese Schlüssel importieren und das aktuelle Schlüsselmaterial mithilfe der Rotation bei Bedarf aktualisieren. Das aktuelle Schlüsselmaterial wird sowohl für die Verschlüsselung als auch für die Entschlüsselung verwendet, andere (nicht aktuelle) Schlüsselmaterialien können jedoch nur zur Entschlüsselung verwendet werden.

Sie können die Herkunft des Schlüsselmaterials nicht ändern

KMS-Schlüssel, die für importiertes Schlüsselmaterial entwickelt wurden, haben einen Ursprungswert von`EXTERNAL`, der nicht geändert werden kann. Sie können einen KMS-Schlüssel für importiertes Schlüsselmaterial nicht konvertieren, um Schlüsselmaterial aus einer anderen

Quelle zu verwenden, einschließlich. AWS KMS Ebenso können Sie einen KMS-Schlüssel mit Schlüsselmaterial nicht in einen AWS KMS KMS-Schlüssel konvertieren, der für importiertes Schlüsselmaterial entworfen wurde.

Sie können kein Schlüsselmaterial exportieren

Sie können kein Schlüsselmaterial exportieren, das Sie importiert haben. AWS KMS Sie können das importierte Schlüsselmaterial in keiner Form an Sie zurücksenden. Sie müssen eine Kopie Ihres importierten Schlüsselmaterials außerhalb von AWS, vorzugsweise in einem Schlüsselmanager, wie z. B. einem Hardware-Sicherheitsmodul (HSM), aufbewahren, damit Sie das Schlüsselmaterial erneut importieren können, falls Sie es löschen oder wenn es abläuft.

Sie können multiregionale Schlüssel mit importiertem Schlüsselmaterial erstellen

Multi-Region mit importiertem Schlüsselmaterial haben die Eigenschaften von KMS-Schlüsseln mit importiertem Schlüsselmaterial und können zwischen AWS-Regionen interoperieren. Um einen multiregionalen Schlüssel mit importiertem Schlüsselmaterial zu erstellen, müssen Sie dasselbe Schlüsselmaterial in den KMS-Schlüssel und in jeden Replikatschlüssel importieren. Symmetrische Verschlüsselungsschlüssel für mehrere Regionen unterstützen keine Rotation nach Bedarf.

Asymmetrische Schlüssel und HMAC-Schlüssel sind portabel und interoperabel

Sie können Ihr asymmetrisches Schlüsselmaterial und Ihr HMAC-Schlüsselmaterial außerhalb von verwenden, um mit AWS KMS Schlüsseln AWS zu interagieren, die dasselbe importierte Schlüsselmaterial enthalten.

Im Gegensatz zum AWS KMS symmetrischen Chiffretext, der untrennbar mit dem im Algorithmus verwendeten KMS-Schlüssel verbunden ist, werden Standard-HMAC- und asymmetrische Formate für Verschlüsselung, Signierung und MAC-Generierung AWS KMS verwendet. Dadurch sind die Schlüssel übertragbar und unterstützen herkömmliche Escrow-Key-Szenarien.

Wenn Ihr KMS-Schlüssel Schlüsselmaterial importiert hat, können Sie das importierte Schlüsselmaterial außerhalb von verwenden, um die folgenden Operationen auszuführen. AWS

- HMAC-Schlüssel – Sie können ein HMAC-Tag, das durch den HMAC-KMS-Schlüssel generiert wurde, mit importiertem Schlüsselmaterial überprüfen. Sie können den HMAC-KMS-Schlüssel auch zusammen mit dem importierten Schlüsselmaterial verwenden, um ein HMAC-Tag zu verifizieren, das mit dem Schlüsselmaterial außerhalb von generiert wurde. AWS
- Asymmetrische Verschlüsselungsschlüssel — Sie können Ihren privaten asymmetrischen Verschlüsselungsschlüssel außerhalb von verwenden, AWS um einen durch den KMS-Schlüssel verschlüsselten Chiffretext mit dem entsprechenden öffentlichen Schlüssel zu

entschlüsseln. Sie können Ihren asymmetrischen KMS-Schlüssel auch verwenden, um einen asymmetrischen Chiffretext zu entschlüsseln, der außerhalb von generiert wurde. AWS

- Asymmetrische Signaturschlüssel — Sie können Ihren asymmetrischen KMS-Schlüssel mit importiertem Schlüsselmaterial verwenden, um digitale Signaturen zu überprüfen, die mit Ihrem privaten Signaturschlüssel außerhalb von generiert wurden. AWS Sie können Ihren asymmetrischen öffentlichen Signaturschlüssel auch außerhalb von verwenden, um Signaturen AWS zu überprüfen, die mit Ihrem asymmetrischen KMS-Schlüssel generiert wurden.
- Schlüssel für asymmetrische Schlüsselvereinbarungen — Sie können Ihren KMS-Schlüssel für die asymmetrische Schlüsselvereinbarung mit importiertem Schlüsselmaterial verwenden, um gemeinsam genutzte Geheimnisse mit einem Peer außerhalb von abzuleiten. AWS

Wenn Sie dasselbe Schlüsselmaterial in verschiedene KMS-Schlüssel desselben AWS-Region importieren, sind diese Schlüssel ebenfalls interoperabel. Um interoperable KMS-Schlüssel in verschiedenen Sprachen zu erstellen AWS-Regionen, erstellen Sie einen Schlüssel für mehrere Regionen mit importiertem Schlüsselmaterial.

Symmetrische Verschlüsselungsschlüssel sind nicht portabel oder interoperabel

Die daraus resultierenden symmetrischen Chiffretexte sind weder portabel noch AWS KMS interoperabel. AWS KMS veröffentlicht nicht das symmetrische Chiffretext-Format, das für die Portabilität erforderlich ist, und das Format kann sich ohne vorherige Ankündigung ändern.

- AWS KMS kann symmetrische Chiffretexte, die Sie außerhalb von verschlüsseln, nicht entschlüsseln AWS, selbst wenn Sie importiertes Schlüsselmaterial verwenden.
- AWS KMS unterstützt nicht die Entschlüsselung von AWS KMS symmetrischem Chiffretext außerhalb von AWS KMS, selbst wenn der Chiffretext unter einem KMS-Schlüssel mit importiertem Schlüsselmaterial verschlüsselt wurde.
- KMS-Schlüssel mit demselben importierten Schlüsselmaterial sind nicht interoperabel. Der symmetrische Chiffretext, der Chiffretext AWS KMS generiert, der für jeden KMS-Schlüssel spezifisch ist. Dieses Geheimtextformat garantiert, dass nur der KMS-Schlüssel, der die Daten verschlüsselt hat, diese entschlüsseln kann.

Außerdem können Sie keine AWS Tools wie die clientseitige Verschlüsselung AWS Encryption SDKoder die clientseitige Amazon S3 S3-Verschlüsselung verwenden, um symmetrische Chiffretexte zu entschlüsseln AWS KMS .

Daher können Sie Schlüssel mit importiertem Schlüsselmaterial nicht zur Unterstützung von Schlüsseltreuhandvereinbarungen verwenden, bei denen ein autorisierter Dritter mit eingeschränktem Zugriff auf Schlüsselmaterial bestimmte Chiffretexte außerhalb von

entschlüsseln kann. AWS KMS Um die Treuhandfunktion für Schlüssel zu unterstützen, verwenden Sie die AWS Encryption SDK, um Ihre Nachricht mit einem Schlüssel zu verschlüsseln, der unabhängig von AWS KMS ist.

Schützen von importiertem Schlüsselmaterial

Das Schlüsselmaterial, das Sie importieren, ist während des Transports und im Ruhezustand geschützt. Bevor Sie das Schlüsselmaterial importieren, verschlüsseln (oder „verpacken“) Sie das Schlüsselmaterial mit dem öffentlichen Schlüssel eines RSA-Schlüsselpaars, das in AWS KMS Hardware-Sicherheitsmodulen (HSMs) generiert wurde, die im Rahmen des FIPS 140-3 Cryptographic Module Validation Program validiert wurden. Sie können das Schlüsselmaterial direkt mit dem öffentlichen Schlüssel zur Verpackung verschlüsseln oder das Schlüsselmaterial mit einem symmetrischen AES-Schlüssel verschlüsseln und anschließend den symmetrischen AES-Schlüssel mit dem öffentlichen RSA-Schlüssel verschlüsseln.

AWS KMS Entschlüsselt das Schlüsselmaterial nach Erhalt mit dem entsprechenden privaten Schlüssel in einem AWS KMS HSM und verschlüsselt es erneut unter einem symmetrischen AES-Schlüssel, der nur im flüchtigen Speicher des HSM vorhanden ist. Ihr Schlüsselmaterial verlässt zu keiner Zeit Ihr HSM im Klartext. Es wird nur entschlüsselt, während es verwendet wird, und nur innerhalb. AWS KMS HSMs

Die Verwendung Ihres KMS-Schlüssels mit importiertem Schlüsselmaterial hängt ausschließlich von den Zugriffskontrollrichtlinien ab, die Sie für den KMS-Schlüssel festlegen. Darüber hinaus können Sie Aliase und Tags verwenden, um den Zugriff auf den KMS-Schlüssel zu identifizieren und zu kontrollieren. Sie können den Schlüssel aktivieren und deaktivieren, ihn anzeigen und überwachen, indem Sie Dienste wie AWS CloudTrail verwenden.

Sie behalten jedoch die einzige ausfallsichere Kopie Ihres Schlüsselmaterials. Im Gegenzug für dieses zusätzliche Maß an Kontrolle sind Sie für die Haltbarkeit und allgemeine Verfügbarkeit des importierten Schlüsselmaterials verantwortlich. AWS KMS ist darauf ausgelegt, importiertes Schlüsselmaterial hochverfügbar zu halten. Die Haltbarkeit von importiertem Schlüsselmaterial wird jedoch AWS KMS nicht auf dem gleichen Niveau gehalten wie das AWS KMS generierte Schlüsselmaterial.

Dieser Unterschied in der Haltbarkeit ist in den folgenden Fällen von Bedeutung:

- Wenn Sie eine Ablaufzeit für Ihr importiertes Schlüsselmaterial festlegen, AWS KMS wird das Schlüsselmaterial nach Ablauf gelöscht. AWS KMS löscht den KMS-Schlüssel oder seine

Metadaten nicht. Sie können einen CloudWatch Amazon-Alarm erstellen, der Sie benachrichtigt, wenn importiertes Schlüsselmaterial sich dem Ablaufdatum nähert.

Sie können kein Schlüsselmaterial löschen, das für einen KMS-Schlüssel AWS KMS generiert wurde, und Sie können nicht festlegen, dass AWS KMS Schlüsselmaterial abläuft.

- Wenn Sie importiertes Schlüsselmaterial manuell AWS KMS löschen, wird das Schlüsselmaterial gelöscht, jedoch nicht der KMS-Schlüssel oder seine Metadaten. Im Gegensatz dazu erfordert die Planung der Schlüssellöschung eine Wartezeit von 7 bis 30 Tagen. Danach werden der KMS-Schlüssel, seine Metadaten und sein Schlüsselmaterial AWS KMS dauerhaft gelöscht.
- Im unwahrscheinlichen Fall bestimmter regionaler Ausfälle, die Auswirkungen haben AWS KMS (z. B. ein vollständiger Stromausfall), AWS KMS kann Ihr importiertes Schlüsselmaterial nicht automatisch wiederhergestellt werden. Der KMS-Schlüssel und seine Metadaten AWS KMS können jedoch wiederhergestellt werden.

Sie müssen eine Kopie des importierten Schlüsselmaterials außerhalb eines von AWS Ihnen kontrollierten Systems aufbewahren. Wir empfehlen, eine exportierbare Kopie des importierten Schlüsselmaterials in einem Schlüsselverwaltungssystem, z. B. einem HSM, zu speichern. Als bewährte Methode sollten Sie einen Verweis auf den KMS-Schlüssel-ARN und die von generierte Schlüsselmaterial-ID AWS KMS zusammen mit der exportierbaren Kopie des Schlüsselmaterials speichern. Wenn Ihr importiertes Schlüsselmaterial gelöscht wird oder abläuft, ist der zugehörige KMS-Schlüssel unbrauchbar, bis Sie dasselbe Schlüsselmaterial erneut importieren. Wenn Ihr importiertes Schlüsselmaterial dauerhaft verloren geht, ist jeder unter dem KMS-Schlüssel verschlüsselte Geheimtext nicht wiederherstellbar.

> **Important**
>
> Symmetrischen Verschlüsselungsschlüsseln mit einer Region können mehrere Schlüsselmaterialien zugeordnet sein. Der gesamte KMS-Schlüssel wird unbrauchbar, sobald Sie eines dieser Schlüsselmaterialien löschen oder wenn eines dieser Schlüsselmaterialien abläuft (es sei denn, das gelöschte oder ablaufende Schlüsselmaterial ist abgelaufen). `PENDING_ROTATION` Sie müssen alle abgelaufenen oder gelöschten Schlüsselmaterialien, die mit einem solchen Schlüssel verknüpft sind, erneut importieren, bevor der Schlüssel für kryptografische Operationen verwendet werden kann.

KMS-Schlüssel in einem CloudHSM-Schlüsselspeicher

Sie können den in einem AWS CloudHSM Schlüsselspeicher erstellen, anzeigen, verwalten, verwenden und das AWS KMS keys Löschen von Dateien planen. Die dafür verwendeten Verfahren ähneln denjenigen für andere KMS-Schlüssel. Der einzige Unterschied besteht darin, dass Sie bei der Erstellung des AWS CloudHSM KMS-Schlüssels einen Schlüsselspeicher angeben. AWS KMS Erstellt dann nicht extrahierbares Schlüsselmaterial für den KMS-Schlüssel im AWS CloudHSM Cluster, der dem AWS CloudHSM Schlüsselspeicher zugeordnet ist. Wenn Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher verwenden, werden die kryptografischen Operationen HSMs im Cluster ausgeführt.

Unterstützte Features

Zusätzlich zu den in diesem Abschnitt beschriebenen Verfahren können Sie mit KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher wie folgt vorgehen:

- Verwenden Sie Schlüsselrichtlinien, IAM-Richtlinien und Erteilungen zur Autorisierung des Zugriffs auf die KMS-Schlüssel.
- Aktivieren und Deaktivieren der KMS-Schlüssel.
- Zuweisen von Tags und Erstellen von Aliassen und Autorisieren des Zugriffs auf die KMS-Schlüssel mithilfe der attributbasierten Zugriffskontrolle (ABAC)
- Verwenden Sie die KMS-Schlüssel, um die folgenden kryptografischen Operationen durchzuführen:
  - Encrypt
  - Decrypt
  - GenerateDataKey
  - GenerateDataKeyWithoutPlaintext
  - ReEncrypt

  Die Operationen, die asymmetrische Datenschlüsselpaare generieren, GenerateDataKeyPairund GenerateDataKeyPairWithoutPlaintext, werden in benutzerdefinierten Schlüsselspeichern nicht unterstützt.
- Verwenden Sie die KMS-Schlüssel mit AWS -Services, die in AWS KMS integriert werden können und kundenverwaltete KMS-Schlüssel unterstützen.
- Verfolgen Sie die Verwendung Ihrer KMS-Schlüssel in AWS CloudTrail Protokollen und CloudWatch Amazon-Überwachungstools.

Nicht unterstützte Funktionen

- AWS CloudHSM Schlüsselspeicher unterstützen nur KMS-Schlüssel mit symmetrischer Verschlüsselung. Sie können keine HMAC-KMS-Schlüssel, asymmetrische KMS-Schlüssel oder asymmetrische Datenschlüsselpaare in einem Schlüsselspeicher erstellen. AWS CloudHSM
- Sie können Schlüsselmaterial nicht in einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher importieren. AWS KMS generiert das Schlüsselmaterial für den KMS-Schlüssel im AWS CloudHSM Cluster.
- Sie können die automatische Rotation des Schlüsselmaterials für einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher nicht aktivieren oder deaktivieren.

Verwendung von KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher

Wenn Sie Ihren KMS-Schlüssel in einer Anfrage verwenden, identifizieren Sie den KMS-Schlüssel anhand seiner ID oder seines Alias. Sie müssen den AWS CloudHSM Schlüsselspeicher oder den AWS CloudHSM Cluster nicht angeben. Die Antwort enthält die gleichen Felder, die auch für alle anderen KMS-Schlüssel mit symmetrischer Verschlüsselung zurückgegeben werden.

Wenn Sie jedoch einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher verwenden, wird der kryptografische Vorgang vollständig innerhalb des AWS CloudHSM Clusters ausgeführt, der dem AWS CloudHSM Schlüsselspeicher zugeordnet ist. Die Produktion verwendet das Schlüsselmaterial im Cluster, das dem ausgewählten KMS-Schlüssel zugeordnet ist.

Damit dies möglich ist, sind die folgenden Bedingungen erforderlich.

- Der Schlüsselstatus des KMS-Schlüssels muss `Enabled` lauten. Um den Schlüsselstatus zu ermitteln, verwenden Sie das Statusfeld in der AWS KMS Konsole oder das `KeyState` Feld in der DescribeKeyAntwort.
- Der AWS CloudHSM Schlüsselspeicher muss mit seinem AWS CloudHSM Cluster verbunden sein. Sein Status muss in der AWS KMS Konsole oder `ConnectionState` in der DescribeCustomKeyStoresAntwort stehen`CONNECTED`.
- Der AWS CloudHSM Cluster, der dem benutzerdefinierten Schlüsselspeicher zugeordnet ist, muss mindestens ein aktives HSM enthalten. Verwenden Sie die Konsole, die AWS KMS Konsole oder HSMs den DescribeClustersVorgang, um die Anzahl der AWS CloudHSM aktiven Benutzer im Cluster zu ermitteln.
- Der AWS CloudHSM Cluster muss das Schlüsselmaterial für den KMS-Schlüssel enthalten. Wenn das Schlüsselmaterial aus dem Cluster gelöscht wurden oder ein HSM aus einer Sicherung erstellt wurde, in der die Schlüsselinformationen nicht enthalten waren, schlägt die kryptografische Produktion fehl.

Wenn diese Bedingungen nicht erfüllt sind, schlägt der kryptografische Vorgang fehl und es wird eine `KMSInvalidStateException` Ausnahme AWS KMS zurückgegeben. In der Regel müssen Sie nur die Verbindung zum AWS CloudHSM Schlüsselspeicher erneut herstellen. Weitere Informationen finden Sie unter Beheben eines fehlerhaften KMS-Schlüssels.

Beachten Sie bei der Verwendung der KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher, dass sich die KMS-Schlüssel in jedem AWS CloudHSM Schlüsselspeicher ein benutzerdefiniertes Schlüsselspeicher-Anforderungskontingent für kryptografische Operationen teilen. Wenn Sie das Kontingent überschreiten, wird a AWS KMS `ThrottlingException` zurückgegeben. Wenn der AWS CloudHSM Cluster, der dem AWS CloudHSM Schlüsselspeicher zugeordnet ist, zahlreiche Befehle verarbeitet, auch solche, die nichts mit dem AWS CloudHSM Schlüsselspeicher zu tun haben, erhalten Sie möglicherweise eine noch geringere Rate. `ThrottlingException` Wenn Sie eine `ThrottlingException` für eine Anforderung erhalten, verringern Sie Ihre Anforderungsrate und führen Sie die Befehle erneut aus. Details zum Anforderungskontingent für benutzerdefinierte Schlüsselspeicher finden Sie unter Anforderungskontingente für benutzerdefinierte Schlüsselspeicher.

Weitere Informationen

- Weitere Informationen zu AWS CloudHSM Schlüsselspeichern finden Sie unterAWS CloudHSM wichtige Geschäfte.
- Informationen zum Erstellen von KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher finden Sie unterErstellen Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher.
- Informationen zum Identifizieren und Anzeigen von KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher finden Sie unterIdentifizieren Sie KMS-Schlüssel in AWS CloudHSM Schlüsselspeichern.
- Informationen zu KMS-Schlüsseln und Schlüsselmaterial in einem AWS CloudHSM Schlüsselspeicher finden Sie unterSuchen Sie KMS-Schlüssel und Schlüsselmaterial in einem AWS CloudHSM Schlüsselspeicher.
- Informationen zu besonderen Überlegungen beim Löschen von KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher finden Sie unter Löschen von KMS-Schlüsseln aus einem AWS CloudHSM Schlüsselspeicher.

KMS-Schlüssel in externen Schlüsselspeichern

Zum Erstellen, Anzeigen, Verwalten, Verwenden und Planen der Löschung von KMS-Schlüsseln in einem externen Schlüsselspeicher verwenden Sie ganz ähnliche Verfahren wie für andere

KMS-Schlüssel. Wenn Sie jedoch einen KMS-Schlüssel in einem externen Schlüsselspeicher erstellen, geben Sie einen externen Schlüsselspeicher und einen externen Schlüssel an. Wenn Sie einen KMS-Schlüssel in einem externen Schlüsselspeicher verwenden, werden die Ver- und Entschlüsselungsvorgänge von Ihrem externen Schlüsselmanager unter Verwendung des angegebenen externen Schlüssels durchgeführt.

AWS KMS kann keine kryptografischen Schlüssel in Ihrem externen Schlüsselmanager erstellen, anzeigen, aktualisieren oder löschen. AWS KMS greift niemals direkt auf Ihren externen Schlüsselmanager oder einen externen Schlüssel zu. Alle Anforderungen für kryptografische Vorgänge werden vom Proxy Ihres externen Schlüsselspeichers vermittelt. Zur Verwendung eines KMS-Schlüssels in einem externen Schlüsselspeicher muss der externe Schlüsselspeicher, der den KMS-Schlüssel hostet, mit seinem externen Schlüsselspeicher-Proxy verbunden sein.

Unterstützte Features

Neben den in diesem Abschnitt beschriebenen Verfahren können Sie mit KMS-Schlüsseln in einem externen Schlüsselspeicher folgende Aktionen ausführen:

- Verwenden von Schlüsselrichtlinien, IAM-Richtlinien und Erteilungen zur Steuerung des Zugriffs auf die KMS-Schlüssel
- Aktivieren und Deaktivieren der KMS-Schlüssel. Diese Aktionen wirken sich nicht auf den externen Schlüssel in Ihrem externen Schlüsselmanager aus.
- Zuweisen von Tags und Erstellen von Aliassen und Autorisieren des Zugriffs auf die KMS-Schlüssel mithilfe der attributbasierten Zugriffskontrolle (ABAC)
- Verwenden Sie die KMS-Schlüssel, um die folgenden kryptografischen Operationen durchzuführen:
  - Encrypt
  - Decrypt
  - GenerateDataKey
  - GenerateDataKeyWithoutPlaintext
  - ReEncrypt

  Die Operationen, die asymmetrische Datenschlüsselpaare generieren, GenerateDataKeyPairund GenerateDataKeyPairWithoutPlaintext, werden in benutzerdefinierten Schlüsselspeichern nicht unterstützt.
- Verwenden der KMS-Schlüssel mit AWS-Services -Services, die in AWS KMS integriert werden können und Unterstützen von kundenverwalteten KMS-Schlüsseln

Nicht unterstützte Funktionen

- Externe Schlüsselspeicher unterstützen nur KMS-Schlüssel mit symmetrischer Verschlüsselung. Sie können keine HMAC-KMS-Schlüssel oder asymmetrische KMS-Schlüssel in einem externen Schlüsselspeicher erstellen.
- GenerateDataKeyPairund GenerateDataKeyPairWithoutPlaintextwerden für KMS-Schlüssel in einem externen Schlüsselspeicher nicht unterstützt.
- Sie können keine AWS::KMS::Key AWS CloudFormation Vorlage verwenden, um einen externen Schlüsselspeicher oder einen KMS-Schlüssel in einem externen Schlüsselspeicher zu erstellen.
- Multiregionale Schlüssel werden in einem externen Schlüsselspeicher nicht unterstützt.
- KMS-Schlüssel mit importiertem Schlüsselmaterial werden in einem externen Schlüsselspeicher nicht unterstützt.
- Automatische Schlüsselrotation wird für KMS-Schlüssel in einen externen Schlüsselspeicher nicht unterstützt.

KMS-Schlüssel in einem externen Schlüsselspeicher verwenden

Wenn Sie Ihren KMS-Schlüssel in einer Anforderung verwenden, identifizieren Sie den KMS-Schlüssel durch Schlüssel-ID, Schlüssel-ARN, Alias oder Alias-ARN. Sie müssen den externen Schlüsselspeicher nicht angeben. Die Antwort enthält die gleichen Felder, die auch für alle anderen KMS-Schlüssel mit symmetrischer Verschlüsselung zurückgegeben werden. Wenn Sie jedoch einen KMS-Schlüssel in einem externen Schlüsselspeicher verwenden, werden die Ver- und Entschlüsselungsvorgänge von Ihrem externen Schlüsselmanager unter Verwendung des externen Schlüssels ausgeführt, der dem KMS-Schlüssel zugeordnet ist.

Um sicherzustellen, dass Chiffretext, der mit einem KMS-Schlüssel in einem externen Schlüsselspeicher verschlüsselt wurde, mindestens so sicher ist wie jeder Chiffretext, der mit einem Standard-KMS-Schlüssel verschlüsselt wurde, AWS KMS verwendet doppelte Verschlüsselung. Daten werden zunächst mithilfe von Schlüsselmaterial verschlüsselt. AWS KMS AWS KMS Dann werden sie von Ihrem externen Schlüsselmanager mit dem externen Schlüssel für den KMS-Schlüssel verschlüsselt. Um doppelt verschlüsselten Geheimtext zu entschlüsseln, wird der Geheimtext zunächst von Ihrem externen Schlüsselmanager mit dem externen Schlüssel für den KMS-Schlüssel entschlüsselt. Anschließend werden sie AWS KMS unter Verwendung des AWS KMS Schlüsselmaterials für den KMS-Schlüssel entschlüsselt.

Damit dies möglich ist, sind die folgenden Bedingungen erforderlich.

- Der Schlüsselstatus des KMS-Schlüssels muss `Enabled` lauten. Den Status des Schlüssels finden Sie im Feld Status für vom Kunden verwaltete Schlüssel, die AWS KMS Konsole oder das `KeyState` Feld in der DescribeKeyAntwort.
- Der externe Schlüsselspeicher, der den KMS-Schlüssel hostet, muss mit seinem externen Schlüsselspeicher-Proxy verbunden sein, d. h. der Verbindungsstatus des externen Schlüsselspeichers muss `CONNECTED` sein.

  Sie können den Verbindungsstatus auf der Seite Externe Schlüsselspeicher in der AWS KMS Konsole oder in der DescribeCustomKeyStoresAntwort einsehen. Der Verbindungsstatus des externen Schlüsselspeicher wird auch auf der Detailseite für den KMS-Schlüssel in der AWS KMS -Konsole angezeigt. Wählen Sie auf der Detailseite die Registerkarte Cryptographic configuration (Kryptografische Konfiguration) und sehen Sie im Feld Connection state (Verbindungsstatus) im Abschnitt Custom key store (Benutzerdefinierter Schlüsselspeicher) nach.

  Wenn der Verbindungsstatus `DISCONNECTED` ist, müssen Sie zuerst eine Verbindung herstellen. Wenn der Verbindungsstatus `FAILED` lautet, müssen Sie das Problem lösen, den externen Schlüsselspeicher trennen und ihn dann verbinden. Detaillierte Anweisungen finden Sie unter Externe Schlüsselspeicher Connect und trennen.
- Der Proxy des externen Schlüsselspeichers muss in der Lage sein, den externen Schlüssel zu finden.
- Der externe Schlüssel muss aktiviert sein und er muss die Verschlüsselung und Entschlüsselung durchführen.

  Der Status des externen Schlüssels ist unabhängig von Änderungen des Schlüsselstatus des KMS-Schlüssels, einschließlich der Aktivierung und Deaktivierung des KMS-Schlüssels, und wird von diesen nicht beeinflusst. Ebenso ändert das Deaktivieren oder Löschen des externen Schlüssels nicht den Schlüsselstatus des KMS-Schlüssels, aber kryptografische Vorgänge, die den zugehörigen KMS-Schlüssel verwenden, schlagen fehl.

Wenn diese Bedingungen nicht erfüllt sind, schlägt der kryptografische Vorgang fehl und es wird eine `KMSInvalidStateException` Ausnahme AWS KMS zurückgegeben. Möglicherweise müssen Sie den externen Schlüsselspeicher erneut verbinden oder Tools Ihres externen Schlüsselmanagers verwenden, um Ihren externen Schlüssel neu zu konfigurieren oder zu reparieren. Weitere Informationen finden Sie unter the section called “Fehlerbehebung bei externen Schlüsselspeichern”.

Bei der Verwendung von KMS-Schlüsseln in einem externen Schlüsselspeicher ist zu beachten, dass die KMS-Schlüssel in jedem externen Schlüsselspeicher gemeinsam ein Anforderungskontingent für benutzerdefinierte Schlüsselspeicher für kryptografische Vorgänge nutzen. Wenn Sie das Kontingent überschreiten, wird a AWS KMS `ThrottlingException` zurückgegeben. Details zum Anforderungskontingent für benutzerdefinierte Schlüsselspeicher finden Sie unter Anforderungskontingente für benutzerdefinierte Schlüsselspeicher.

Weitere Informationen

- Weitere Informationen zu externen Schlüsselspeichern finden Sie unterExterne Schlüsselspeicher.
- Weitere Informationen über Schlüsselmaterial in externen Schlüsselspeichern finden Sie unterExterner Schlüssel.
- Informationen zum Erstellen von KMS-Schlüsseln in einem externen Schlüsselspeicher finden Sie unterErstellen Sie einen KMS-Schlüssel in externen Schlüsselspeichern.
- Informationen zum Identifizieren und Anzeigen von KMS-Schlüsseln in einem externen Schlüsselspeicher finden Sie unterIdentifizieren Sie KMS-Schlüssel in externen Schlüsselspeichern.
- Informationen zu besonderen Überlegungen beim Löschen von KMS-Schlüsseln in einem externen Schlüsselspeicher finden Sie unter Löschen von KMS-Schlüsseln aus einem externen Schlüsselspeicher.

AWS KMS Grundlagen der Kryptografie

AWS KMS verwendet konfigurierbare kryptografische Algorithmen, sodass das System schnell von einem zugelassenen Algorithmus oder Modus zu einem anderen migrieren kann. Der anfängliche Standardsatz kryptografischer Algorithmen wurde aufgrund ihrer Sicherheitseigenschaften und Leistung aus den vom Federal-Information-Processing-Standard-(FIPS)-geprüften Algorithmen ausgewählt.

Weitere Informationen zu bevorzugten und akzeptablen kryptografischen Algorithmen finden Sie unter. Unterstützte kryptografische Algorithmen

Entropie und Zufallszahlengenerierung

AWS KMS Die Schlüsselgenerierung erfolgt in der. AWS KMS HSMs Sie HSMs implementieren einen hybriden Zufallszahlengenerator, der NIST SP800-90A Deterministic Random Bit Generator

(DRBG) CTR_DRBG using AES-256 verwendet. Es wird mit einem nicht-deterministischen Zufallsbitgenerator mit 384 Bit Entropie gesät und mit zusätzlicher Entropie aktualisiert, um Prognosewiderstand bei jedem Aufruf von kryptografischem Material zu bieten.

Symmetrische Schlüsseloperationen (nur Verschlüsselung)

Alle darin verwendeten Befehle zur Verschlüsselung mit symmetrischen Schlüsseln HSMs verwenden die Advanced Encryption Standards (AES), im Galois Counter Mode (GCM) werden 256-Bit-Schlüssel verwendet. Die analogen Aufrufe zum Entschlüsseln verwenden die umgekehrte Funktion.

AES-GCM ist ein authentifiziertes Verschlüsselungsschema. Neben der Verschlüsselung von Klartext zur Erzeugung von Verschlüsselungstext berechnet es ein Authentifizierungs-Tag über den Verschlüsselungstext und alle zusätzlichen Daten, für die eine Authentifizierung erforderlich ist (zusätzlich authentifizierte Daten oder AAD). Mit dem Authentifizierungs-Tag wird sichergestellt, dass die Daten aus der angeblichen Quelle stammen und dass der Verschlüsselungstext und AAD nicht geändert wurden.

Oft wird die AWS Erwähnung des AAD in unseren Beschreibungen weggelassen, insbesondere wenn es um die Verschlüsselung von Datenschlüsseln geht. Durch umgebenden Text wird in diesen Fällen impliziert, dass die zu verschlüsselnde Struktur zwischen dem zu verschlüsselnden Klartext und dem zu schützenden Klartext-AAD aufgeteilt ist.

AWS KMS bietet Ihnen die Möglichkeit, Schlüsselmaterial in eine zu importieren, AWS KMS key anstatt sich auf die Generierung des Schlüsselmaterials AWS KMS zu verlassen. Dieses importierte Schlüsselmaterial kann mit RSAES-OAEP verschlüsselt werden, um den Schlüssel beim Transport zum HSM zu schützen. AWS KMS Die RSA-Schlüsselpaare werden am generiert. AWS KMS HSMs Das importierte Schlüsselmaterial wird auf einem AWS KMS HSM entschlüsselt und unter AES-GCM erneut verschlüsselt, bevor es vom Dienst gespeichert wird.

Asymmetrische Schlüsseloperationen (Verschlüsselung, digitale Signatur und Signaturüberprüfung)

AWS KMS unterstützt die Verwendung asymmetrischer Schlüsseloperationen sowohl für die Verschlüsselung als auch für die digitale Signatur und für Schlüsselvereinbarungen. Asymmetrische Schlüsseloperationen basieren auf einem mathematisch verwandten Paar aus öffentlichem und privatem Schlüssel, das Sie zur Verschlüsselung und Entschlüsselung, Signierung und Signaturüberprüfung oder zum Ableiten gemeinsamer Geheimnisse verwenden können. Der private Schlüssel wird niemals unverschlüsselt verlassen. AWS KMS Sie können den öffentlichen Schlüssel

intern verwenden, AWS KMS indem Sie die AWS KMS API-Operationen aufrufen, oder Sie können den öffentlichen Schlüssel herunterladen und ihn außerhalb von AWS KMS verwenden.

AWS KMS unterstützt die folgenden asymmetrischen Chiffren.

- RSA-OAEP (zur Verschlüsselung) und RSA-PSS und RSA-PKCS-#1-v1_5 (zum Signieren und Verifizieren) – Unterstützt RSA-Schlüssellängen (in Bit): 2048, 3072 und 4096 für unterschiedliche Sicherheitsanforderungen.
- Elliptische Kurve (ECC) — Wird zum Signieren und Verifizieren oder zum Ableiten von gemeinsamen Geheimnissen verwendet, aber nicht für beides. Unterstützt ECC-Kurven: NIST P256, P384, P521, SECP 256k1.
- ML-DSA — Wird zum Signieren und Verifizieren verwendet. Unterstützte ML-DSA-Schlüsselspezifikationen sind: ML_DSA_44, ML_DSA_65 und ML_DSA_87.
- SM2 (Nur Regionen in China) — Wird zur Verschlüsselung und Entschlüsselung, zum Signieren und Überprüfen oder zum Ableiten von gemeinsamen Geheimnissen verwendet, aber Sie müssen eine Schlüsselverwendung wählen. Unterstützt SM2 PKE für die Verschlüsselung und SM2 DSA für das Signieren.

Schlüsselableitungsfunktionen

Eine Funktion zur Schlüsselableitung wird verwendet, um zusätzliche Schlüssel aus einem anfänglichen Geheimnis oder Schlüssel abzuleiten. AWS KMS verwendet eine Schlüsselableitungsfunktion (KDF), um Schlüssel pro Anruf für jede Verschlüsselung unter einem abzuleiten. AWS KMS keyAlle KDF-Operationen verwenden das KDF im Zählermodus unter Verwendung von HMAC [] mit [0]. FIPS197 SHA256 FIPS18 Der abgeleitete 256-Bit-Schlüssel wird mit AES-GCM verwendet, um Kundendaten und Schlüssel zu verschlüsseln oder zu entschlüsseln.

AWS KMS interne Verwendung digitaler Signaturen

Digitale Signaturen werden auch verwendet, um Befehle und Kommunikationen zwischen AWS KMS Entitäten zu authentifizieren. Alle Dienstentitäten verfügen über ein Schlüsselpaar des elliptischen Kurvendigitalsignaturalgorithmus (ECDSA). Sie führen ECDSA gemäß der Definition in: Verwendung von Algorithmen der Elliptic Curve Cryptography (ECC) in der kryptografischen Nachrichtensyntax (CMS) und X9.62-2005: Kryptografie mit öffentlichen Schlüsseln für die Finanzdienstleistungsbranche: Der Elliptic Curve Digital Signature Algorithm (ECDSA) durch. Die Unternehmen verwenden den in Federal Information Processing Standards Publications, FIPS PUB

180-4, definierten sicheren Hash-Algorithmus, bekannt als. SHA384 Die Schlüssel werden auf der Kurve secp384r1 (NIST-P384) generiert.

Umschlagverschlüsselung

Wenn Sie Daten verschlüsseln, sind die Daten geschützt. Sie müssen aber jetzt den Verschlüsselungsschlüssel schützen. Eine Strategie besteht darin, ihn zu verschlüsseln. Envelope-Verschlüsselung bezeichnet das Verschlüsseln von Klartextdaten mit einem Datenschlüssel und die anschließende Verschlüsselung des Datenschlüssels mit einem anderen Schlüssel.

Sie können auch den Datenverschlüsselungsschlüssel mit einem anderen Verschlüsselungsschlüssel verschlüsseln und diesen Verschlüsselungsschlüssel mit einem weiteren Verschlüsselungsschlüssel verschlüsseln. Doch schließlich muss ein Schlüssel als Klartext verbleiben, damit Sie die Schlüssel und die Daten entschlüsseln können. Dieser Top-Level-Klartext-Verschlüsselungsschlüssel zum Verschlüsseln von Schlüsseln wird als Stammschlüssel bezeichnet.

AWS KMS hilft Ihnen dabei, Ihre Verschlüsselungsschlüssel zu schützen, indem sie sicher gespeichert und verwaltet werden. Der in AWS KMS den AWS KMS FIPS 140-3 Security Level 3-validierten Hardware-Sicherheitsmodulen gespeicherte Root-Schlüssel AWS KMS keys, auch bekannt als, darf diese niemals unverschlüsselt verlassen. Um einen KMS-Schlüssel zu verwenden, müssen Sie anrufen. AWS KMS

Eine grundlegende Konstruktion, die in vielen kryptographischen Systemen verwendet wird, ist die Envelope-Verschlüsselung. Bei der Envelope-Verschlüsselung werden zwei oder mehr kryptografische Schlüssel verwendet, um eine Nachricht zu sichern. Typischerweise wird ein Schlüssel von einem längerfristigen statischen Schlüssel k abgeleitet, und ein anderer Schlüssel ist ein Schlüssel pro Nachricht, msgKey, der erzeugt wird, um die Nachricht zu verschlüsseln. Der Umschlag (Envelope) wird durch Verschlüsseln der Nachricht gebildet: ciphertext = Encrypt(msgKey, message) . Anschließend wird der Nachrichtenschlüssel mit dem statischen Langzeitschlüssel verschlüsselt: encKey = Encrypt(k, msgKey) . Schließlich werden die beiden Werte (encKey, Ciphertext) in eine einzelne Struktur oder eine Envelope-veschlüsselte Nachricht verpackt.

Der Empfänger mit Zugriff auf k kann die Envelope-Nachricht öffnen, indem er zuerst den verschlüsselten Schlüssel entschlüsselt und dann die Nachricht entschlüsselt.

AWS KMS bietet die Möglichkeit, diese längerfristigen statischen Schlüssel zu verwalten und den Prozess der Umschlagverschlüsselung Ihrer Daten zu automatisieren.

Zusätzlich zu den im AWS KMS Service bereitgestellten Verschlüsselungsfunktionen bietet das AWS Encryption SDK clientseitige Envelope-Verschlüsselungsbibliotheken. Sie können diese Bibliotheken verwenden, um Ihre Daten und die Verschlüsselungsschlüssel zu schützen, die zum Verschlüsseln dieser Daten verwendet werden.

Envelope-Verschlüsselung bietet mehrere Vorteile:

- Schutz von Datenschlüsseln

  Wenn Sie einen Datenschlüssel verschlüsseln, müssen Sie sich nicht darum sorgen, wo Sie den verschlüsselten Schlüssel speichern, da die Sicherheit dieses Datenschlüssels inhärent durch Verschlüsselung geschützt ist. Sie können den verschlüsselten Datenschlüssel sicher neben den verschlüsselten Daten speichern.
- Verschlüsselung der gleichen Daten unter mehreren Schlüsseln

  Verschlüsselungsoperationen können zeitaufwendig sein, insbesondere wenn die zu verschlüsselnden Daten große Objekte sind. Statt Rohdaten mit verschiedenen Schlüsseln mehrmals neu zu verschlüsseln, können Sie einfach die Datenschlüssel, die die Rohdaten schützen, neu verschlüsseln.
- Kombination der Stärken mehrerer Algorithmen

  Im Allgemeinen sind symmetrische Schlüsselalgorithmen schneller und erzeugen kleinere Verschlüsselungstexte als öffentliche Schlüsselalgorithmen. Algorithmen mit öffentlichem Schlüssel unterstützen jedoch eine inhärente Rollentrennung und eine einfachere Schlüsselverwaltung. Die Envelope-Verschlüsselung vereint die Stärken aller Strategien.

Kryptografische Operationen

Bei AWS KMSkryptografischen Vorgängen handelt es sich um API-Operationen, die KMS-Schlüssel zum Schutz von Daten verwenden. Da KMS-Schlüssel im System verbleiben AWS KMS, müssen Sie anrufen, AWS KMS um einen KMS-Schlüssel für einen kryptografischen Vorgang zu verwenden.

Um kryptografische Operationen mit KMS-Schlüsseln durchzuführen, verwenden Sie den AWS SDKs, AWS Command Line Interface (AWS CLI) oder den. AWS -Tools für PowerShell Sie können keine kryptografischen Operationen in der AWS KMS -Konsole ausführen. Beispiele für das Aufrufen der kryptografischen Operationen in mehreren Programmiersprachen finden Sie unter Codebeispiele für die AWS KMS Verwendung AWS SDKs.

In der folgenden Tabelle sind die AWS KMS kryptografischen Operationen aufgeführt. Außerdem werden die Anforderungen an den Schlüsseltyp und die Schlüsselnutzung für KMS-Schlüssel angezeigt, die in der Operation verwendet werden.

Operation	Schlüsseltyp	Schlüsselnutzung
Decrypt	Symmetrisch oder asymmetrisch	`ENCRYPT_DECRYPT`
DeriveSharedSecret	Asymmetrisch	`KEY_AGREEMENT`
Encrypt	Symmetrisch oder asymmetrisch	`ENCRYPT_DECRYPT`
GenerateDataKey	Symmetrisch	`ENCRYPT_DECRYPT`
GenerateDataKeyPair	Symmetrisch [1] - nicht unterstützt bei KMS-Schlüsseln in benutzerdefinierten Schlüsselspeichern.	`ENCRYPT_DECRYPT`
GenerateDataKeyPairWithoutPlaintext	Symmetrisch [1] - nicht unterstützt bei KMS-Schlüsseln in benutzerdefinierten Schlüsselspeichern.	`ENCRYPT_DECRYPT`
GenerateDataKeyWithoutPlaintext	Symmetrisch	`ENCRYPT_DECRYPT`
GenerateMac	HMAC	`GENERATE_VERIFY_MAC`
GenerateRandom	Bei dieser Operation wird kein KMS-Schlüssel verwendet.	N/A

Operation	Schlüsseltyp	Schlüsselnutzung
ReEncrypt	Symmetrisch oder asymmetrisch	`ENCRYPT_DECRYPT`
Sign	Asymmetrisch	`SIGN_VERIFY`
Verify	Asymmetrisch	`SIGN_VERIFY`
VerifyMac	HMAC	`GENERATE_VERIFY_MAC`

[1] Generiert ein asymmetrisches Datenschlüsselpaar, das durch einen symmetrischen KMS-Schlüssel geschützt ist.

Informationen zu den Berechtigungen für kryptografische Operationen finden Sie unter the section called “Berechtigungsreferenz”.

Damit alle Benutzer AWS KMS schnell reagieren und einen hohen Funktionsumfang haben, werden AWS KMS Kontingente für die Anzahl der pro Sekunde aufgerufenen kryptografischen Operationen eingerichtet. Details hierzu finden Sie unter the section called “Gemeinsame Kontingente für kryptografische Operationen”.

Unterstützte kryptografische Algorithmen

In der folgenden Tabelle sind die kryptografischen Algorithmen, Chiffren, Modi und Schlüsselgrößen zusammengefasst, die in allen Diensten zum AWS Schutz Ihrer Daten eingesetzt werden. Dies sollte nicht als erschöpfende Liste aller verwendeten Kryptographien angesehen werden. AWS Die Algorithmen lassen sich in zwei Kategorien einteilen: „Bevorzugte“ Algorithmen, die Industriestandards erfüllen und die Interoperabilität fördern, und „Akzeptabel“, die aus Kompatibilitätsgründen in bestimmten Anwendungen verwendet werden können, aber nicht bevorzugt werden. Sie sollten die folgenden Informationen berücksichtigen, wenn Sie Ihre kryptografischen Entscheidungen für Ihre Verschlüsselungsanwendungsfälle treffen.

Weitere Informationen zu den in AWS bereitgestellten kryptografischen Algorithmen finden Sie unter Kryptografiealgorithmen und. AWS-Services

Kryptografische Algorithmen

In den folgenden Tabellen sind die kryptografischen Algorithmen, Chiffren, Modi und Schlüsselgrößen zusammengefasst, die in allen Diensten zum Schutz AWS Ihrer Daten eingesetzt werden. Sie sollten nicht als vollständige Liste aller verfügbaren Kryptografieoptionen in betrachtet werden. AWS Die Algorithmen lassen sich in zwei Kategorien einteilen:

- Bevorzugte Algorithmen erfüllen die AWS Sicherheits- und Leistungsstandards.
- Zulässige Algorithmen können aus Kompatibilitätsgründen in einigen Anwendungen verwendet werden, werden aber nicht bevorzugt.

Asymmetrische Kryptografie

In der folgenden Tabelle sind die unterstützten asymmetrischen Algorithmen für Verschlüsselung, Schlüsselvereinbarung und digitale Signaturen aufgeführt.

Typ	Algorithmus	Status
Verschlüsselung	RSA-OAEP (2048- oder 3072-Bit-Modul)	Akzeptabel
Verschlüsselung	HPKE (P-256 oder P-384, HKDF und AES-GCM)	Akzeptabel
Wichtiges Abkommen	ML-KEM-768 oder ML-KEM-1024	Bevorzugt (quantenresistent)
Wichtiges Abkommen	ECDH (E) mit P-384	Akzeptabel
Wichtige Vereinbarung	ECDH (E) mit P-256, P-521 oder X25519	Akzeptabel
Wichtiges Abkommen	ECDH (E) mit Brainpool P256R1, Brainpool P384R1 oder Brainpool P512R1	Akzeptabel
Signatures (Signaturen)	ML-DSA-65 oder ML-DSA-87	Bevorzugt (quantenresistent)

Typ	Algorithmus	Status
Signatures (Signaturen)	SLH-DSA	Bevorzugtes (quantenr esistentes Signieren) software/ firmware
Signatures (Signaturen)	ECDSA mit P-384	Akzeptabel
Signatures (Signaturen)	ECDSA mit P-256, P-521 oder Ed25519	Akzeptabel
Signatures (Signaturen)	RSA-2048 oder RSA-3072	Akzeptabel

Symmetrische Kryptografie

In der folgenden Tabelle sind die unterstützten symmetrischen Algorithmen für Verschlüsselung, authentifizierte Verschlüsselung und Schlüsselumhüllung aufgeführt.

Typ	Algorithmus	Status
Authentifizierte Verschlüs selung	AES-GCM-256	Bevorzugt
Authentifizierte Verschlüs selung	AES-GCM-128	Akzeptabel
Authentifizierte Verschlüs selung	ChaCha20/Poly1305	Akzeptabel
Verschlüsselungsmodi	AES-XTS-256 (für Blockspei cher)	Bevorzugt
Verschlüsselungsmodi	AES-CBC/CTR (nicht authentif izierte Modi)	Akzeptabel
Verpackung der Schlüssel	AES-GCM-256	Bevorzugt
Verpackung der Schlüssel	AES-KW oder AES-KWP mit 256-Bit-Schlüsseln	Akzeptabel

Kryptografische Funktionen

In der folgenden Tabelle sind die unterstützten Algorithmen für Hashing, Schlüsselableitung, Nachrichtenauthentifizierung und Passwort-Hashing aufgeführt.

Typ	Algorithmus	Status
Hashing	SHA2-384	Bevorzugt
Hashing	SHA2-256	Akzeptabel
Hashing	SHA3	Akzeptabel
Wichtige Ableitung	HKDF_Expand oder HKDF mit -256 SHA2	Bevorzugt
Ableitung von Schlüsseln	Zählermodus KDF mit HMAC- -256 SHA2	Akzeptabel
Authentifizierungscode für Nachrichten	HMAC- -384 SHA2	Bevorzugt
Authentifizierungscode für Nachrichten	HMAC-256 SHA2	Akzeptabel
Authentifizierungscode für Nachrichten	KMAC	Akzeptabel
Passwort-Hashing	verschlüsseln mit SHA384	Bevorzugt
Passwort-Hashing	PBKDF2	Akzeptabel

KMS-Schlüsselzugriff und Berechtigungen

Für die Verwendung AWS KMS benötigen Sie Anmeldeinformationen, mit denen Sie Ihre Anfragen authentifizieren AWS können. Die Anmeldeinformationen müssen Berechtigungen für den Zugriff auf AWS Ressourcen AWS KMS keys und Aliase enthalten. Kein AWS Prinzipal hat Berechtigungen für einen KMS-Schlüssel, es sei denn, diese Berechtigung wird ausdrücklich erteilt und niemals verweigert. Es gibt keine implizite oder automatische Berechtigung zur Verwendung oder Verwaltung eines KMS-Schlüssels.

Um den Zugriff auf Ihre KMS-Schlüssel zu steuern, können Sie die folgenden Richtlinienmechanismen verwenden.

- Schlüsselrichtlinie – Jeder KMS-Schlüssel besitzt eine Schlüsselrichtlinie. Schlüsselrichtlinien sind der primäre Mechanismus zur Steuerung des Zugriffs auf KMS-Schlüssel. Sie können einzig die Schlüsselrichtlinie zum Steuern des Zugriffs verwenden, was bedeutet, dass der vollständige Umfang des Zugriffs auf den KMS-Schlüssel in einem einzigen Dokument definiert wird (der Schlüsselrichtlinie). Weitere Informationen zur Verwendung von Schlüsselrichtlinien finden Sie unter Schlüsselrichtlinien.
- Verwendung von IAM-Richtlinien in Kombination mit der Schlüsselrichtlinie – Sie können IAM-Richtlinien in Kombination mit der Schlüsselrichtlinie verwenden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Diese Art der Zugriffssteuerung ermöglicht Ihnen die Verwaltung aller Berechtigungen für Ihre IAM-Identitäten in IAM. Um mit einer IAM-Richtlinie den Zugriff auf einen KMS-Schlüssel zu ermöglichen, muss die Schlüsselrichtlinie dies explizit erlauben. Weitere Informationen zur Verwendung von IAM-Richtlinien finden Sie unter IAM-Richtlinien.
- Erteilungen – Sie können Erteilungen in Kombination mit der Schlüsselrichtlinie und IAM-Richtlinien verwenden, um den Zugriff auf einen KMS-Schlüssel zu ermöglichen. Diese Art der Zugriffssteuerung ermöglicht es Ihnen, Zugriff auf den KMS-Schlüssel in der Schlüsselrichtlinie zu gewähren und Identitäten die Berechtigung zur Delegierung ihres Zugriffs zu erteilen. Weitere Informationen zur Verwendung von Zugriffserteilungen finden Sie unter Zuschüsse in AWS KMS.

KMS-Schlüsselrichtlinien

Der Zugriff auf Ihre AWS KMS Ressourcen lässt sich in erster Linie mithilfe von Richtlinien verwalten. Richtlinien sind Dokumente, in denen beschrieben wird, welche Prinzipale auf welche Ressourcen zugreifen können. Richtlinien, die mit einer IAM-Identität verknüpft sind, werden als identitätsbasierte Richtlinien (oder IAM-Richtlinien) bezeichnet, und Richtlinien, die mit anderen Arten von Ressourcen

verknüpft sind, werden als Ressourcenrichtlinien bezeichnet. AWS KMS Ressourcenrichtlinien für KMS-Schlüssel werden als Schlüsselrichtlinien bezeichnet.

Alle KMS-Schlüssel verfügen über eine Schlüsselrichtlinie. Wenn Sie keine angeben, AWS KMS wird eine für Sie erstellt. Die verwendete Standardschlüsselrichtlinie unterscheidet sich je nachdem, ob Sie den Schlüssel in der AWS KMS Konsole erstellen oder die AWS KMS API verwenden. AWS KMS Wir empfehlen Ihnen, die Standard-Schlüsselrichtlinie so zu bearbeiten, dass sie den Anforderungen Ihrer Organisation für Berechtigungen mit den geringsten Rechten entspricht.

Sie können nur die Schlüsselrichtlinie verwenden, um den Zugriff zu steuern, wenn sich der Schlüssel und der IAM-Prinzipal in demselben AWS Konto befinden. Das bedeutet, dass der gesamte Zugriffsumfang auf den KMS-Schlüssel in einem einzigen Dokument (der Schlüsselrichtlinie) definiert ist. Wenn jedoch ein Anrufer in einem Konto auf einen Schlüssel in einem anderen Konto zugreifen muss, können Sie die Schlüsselrichtlinie nicht allein verwenden, um den Zugriff zu gewähren. Im kontoübergreifenden Szenario muss dem Benutzer oder der Rolle des Anrufers eine IAM-Richtlinie zugewiesen werden, die es dem Anrufer ausdrücklich ermöglicht, den API-Aufruf durchzuführen.

Sie können IAM-Richtlinien auch in Kombination mit wichtigen Richtlinien und Zuschüssen verwenden, um den Zugriff auf einen KMS-Schlüssel zu kontrollieren. Um eine IAM-Richtlinie zur Steuerung des Zugriffs auf einen KMS-Schlüssel zu verwenden, muss die Schlüsselrichtlinie dem Konto die Erlaubnis zur Verwendung von IAM-Richtlinien erteilen. Sie können entweder eine wichtige Richtlinienerklärung angeben, die IAM-Richtlinien aktiviert, oder Sie können in der Schlüsselrichtlinie explizit zulässige Prinzipale angeben.

Achten Sie beim Schreiben von Richtlinien darauf, dass Sie über strenge Kontrollen verfügen, die einschränken, wer die folgenden Aktionen ausführen kann:

- Aktualisieren, erstellen und löschen Sie IAM- und KMS-Schlüsselrichtlinien
- Hängen Sie IAM-Richtlinien an Benutzer, Rollen und Gruppen an und trennen Sie sie
- Hängen Sie KMS-Schlüsselrichtlinien an Ihre KMS-Schlüssel an und trennen Sie sie

KMS-Schlüsselzuschüsse

AWS KMS Unterstützt neben IAM und wichtigen Richtlinien auch Zuschüsse. Zuschüsse bieten eine flexible und leistungsstarke Möglichkeit, Berechtigungen zu delegieren. Mithilfe von Zuschüssen können Sie zeitgebundenen KMS-Schlüsselzugriff auf IAM-Prinzipale in Ihrem AWS Konto oder in anderen Konten gewähren. AWS Wir empfehlen, zeitgebundenen Zugriff zu gewähren, wenn Sie

die Namen der Principals zum Zeitpunkt der Erstellung der Richtlinien nicht kennen oder wenn sich die Principals, für die Zugriff erforderlich ist, häufig ändern. Der Prinzipal des Empfängers kann sich in demselben Konto wie der KMS-Schlüssel oder in einem anderen Konto befinden. Wenn sich der Prinzipal und der KMS-Schlüssel in unterschiedlichen Konten befinden, müssen Sie zusätzlich zum Zuschuss eine IAM-Richtlinie angeben. Zuschüsse erfordern zusätzliche Verwaltung, da Sie eine API aufrufen müssen, um den Zuschuss zu erstellen und den Zuschuss zurückzuziehen oder zu widerrufen, wenn er nicht mehr benötigt wird.

Die wichtigsten Richtlinien in AWS KMS

Eine Schlüsselrichtlinie ist eine Ressourcenrichtlinie für eine. AWS KMS key Schlüsselrichtlinien sind die primäre Methode zur Zugriffssteuerung für KMS-Schlüssel. Jeder KMS-Schlüssel muss genau eine Schlüsselrichtlinie haben. Die Anweisungen im Schlüsselrichtliniendokument legen fest, wer über eine Berechtigung zur Verwendung des KMS-Schlüssels verfügt, und wie diese Verwendung erfolgen kann. Sie können den Zugriff auf den KMS-Schlüssel auch mithilfe von IAM-Richtlinien und Erteilungen steuern, jeder KMS-Schlüssel muss aber über ein Schlüsselrichtliniendokument verfügen.

Kein AWS Hauptbenutzer, auch nicht der Root-Benutzer oder der Ersteller des Schlüssels, hat Berechtigungen für einen KMS-Schlüssel, es sei denn, sie sind in einer Schlüsselrichtlinie, IAM-Richtlinie oder Grant ausdrücklich erlaubt und niemals verweigert.

Wenn die Schlüsselrichtlinie es nicht ausdrücklich erlaubt, können Sie keine IAM-Richtlinien verwenden, um den Zugriff auf einen KMS-Schlüssel zu erlauben. Ohne Berechtigung der Schlüsselrichtlinie haben IAM-Richtlinien, die Berechtigungen erlauben, keine Auswirkungen. (Sie können eine IAM-Richtlinie verwenden, um eine Berechtigung für einen KMS-Schlüssel ohne Berechtigung einer Schlüsselrichtlinie zu verweigern.) Die Standardschlüsselrichtlinie aktiviert IAM-Richtlinien. Um IAM-Richtlinien in Ihrer Schlüsselrichtlinie zu aktivieren, fügen Sie die unter Erlaubt den Zugriff auf das AWS-Konto und aktiviert IAM-Richtlinien beschriebene Richtlinienanweisung hinzu .

Im Gegensatz zu IAM-Richtlinien, die global sind, sind Schlüsselrichtlinien regional. Eine wichtige Richtlinie steuert den Zugriff nur auf einen KMS-Schlüssel in derselben Region. Es hat keine Auswirkungen auf KMS-Schlüssel in anderen Regionen.

Themen

- Erstellen einer Schlüsselrichtlinie
- Standardschlüsselrichtlinie

- Wichtige Richtlinien anzeigen
- Ändern Sie eine wichtige Richtlinie
- Berechtigungen für AWS Dienste in wichtigen Richtlinien

Erstellen einer Schlüsselrichtlinie

Sie können wichtige Richtlinien in der AWS KMS Konsole oder mithilfe von AWS KMS API-Operationen wie CreateKeyReplicateKey, und und erstellen und PutKeyPolicyverwalten.

Wenn Sie einen KMS-Schlüssel in der AWS KMS Konsole erstellen, führt Sie die Konsole Schritt für Schritt durch die Erstellung einer Schlüsselrichtlinie, die auf der Standardschlüsselrichtlinie für die Konsole basiert. Wenn Sie `CreateKey` oder verwenden `ReplicateKey` APIs, wenn Sie keine Schlüsselrichtlinie angeben, APIs wenden diese die Standardschlüsselrichtlinie für programmgesteuert erstellte Schlüssel an. Wenn Sie die `PutKeyPolicy`-API verwenden müssen Sie eine Schlüsselrichtlinie angeben.

Jedes Richtliniendokument kann eine oder mehrere Richtlinienanweisungen enthalten. Das folgende Beispiel zeigt ein gültiges Schlüsselrichtliniendokument mit einer Richtlinienanweisung.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DescribePolicyStatement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/Alice"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:KeySpec": "SYMMETRIC_DEFAULT"
        }
      }
    }
  ]
```

```
}
```

Themen

- Schlüsselrichtlinienformat
- Elemente in einer Schlüsselrichtlinie
- Beispiel für eine Schlüsselrichtlinie

Schlüsselrichtlinienformat

Ein Schlüsselrichtliniendokument muss den folgenden Regeln entsprechen:

- Bis zu 32 Kilobytes (32 768 Bytes)
- Das `Sid`-Element in einer Schlüsselrichtlinienanweisung kann Leerzeichen enthalten. (Leerzeichen sind im `Sid`-Element eines IAM-Richtliniendokuments untersagt.)

Ein Schlüsselrichtliniendokument darf nur die folgenden Zeichen enthalten:

- Druckbare ASCII-Zeichen
- Die druckbaren Zeichen im zusätzlichen Zeichensatz Basic Latin und Latin-1 Supplement
- Die Sonderzeichen Tabulator (`\u0009`), Zeilenvorschub (`\u000A`) und Wagenrücklauf (`\u000D`)

Elemente in einer Schlüsselrichtlinie

Ein Schlüsselrichtliniendokument muss die folgenden Elemente besitzen:

Version

Gibt die Schlüsselrichtliniendokumentversion an. Wir empfehlen, die Version auf `2012-10-17` (neueste Version) einzustellen.

Statement

Fügt die Richtlinienanweisungen bei. Ein Schlüsselrichtliniendokument muss mindestens eine Anweisung enthalten.

Jede Schlüsselrichtlinienanweisung kann aus bis zu sechs Elementen bestehen. Die Elemente `Effect`, `Principal`, `Action`, und `Resource` sind erforderlich.

Sid

(Optional) Der Anweisungsbezeichner (`Sid`) ist eine beliebigen Zeichenfolge, die Sie zur Beschreibung der Anweisung verwenden können. Der `Sid` in einer Schlüsselrichtlinie kann Leerzeichen enthalten. (Sie können keine Leerzeichen in ein `Sid`-Element einer IAM-Richtlinie aufnehmen.)

Auswirkung

(Erforderlich) Gibt an, ob die Berechtigungen in der Richtlinienanweisung zugelassen oder verweigert werden. Gültige Werte sind `Allow` oder `Deny`. Wenn Sie den Zugriff auf einen KMS-Schlüssel nicht explizit erlauben, wird er implizit verweigert. Sie können den Zugriff auf einen KMS-Schlüssel auch explizit verweigern. So können Sie zum Beispiel sicherstellen, dass Benutzer nicht darauf zugreifen können, auch wenn der Zugriff durch eine andere Richtlinie gestattet wird.

Auftraggeber

(Erforderlich) Der Prinzipal ist die Identität, die die in der Richtlinienanweisung angegebenen Berechtigungen erhält. Sie können IAM-Benutzer AWS-Konten, IAM-Rollen und einige AWS Dienste als Prinzipale in einer Schlüsselrichtlinie angeben. IAM-Benutzergruppen sind kein gültiger Prinzipal in irgendeinem Richtlinientyp.

Ein Sternchenwert, z. B. `"AWS": "*"`, steht für alle AWS -Identitäten in allen Konten.

> **Important**
>
> Setzen Sie den Prinzipal nicht auf ein Sternchen (*) in einer Schlüsselrichtlinienanweisung, die Berechtigungen erlaubt, es sei denn, Sie verwenden Bedingungen, um die Schlüsselrichtlinie einzuschränken. Ein Sternchen gibt jede Identität in jeder AWS-Konto Berechtigung zur Verwendung des KMS-Schlüssels an, sofern dies nicht in einer anderen Richtlinienanweisung ausdrücklich verweigert wird. Benutzer in anderen Ländern AWS-Konten können Ihren KMS-Schlüssel immer dann verwenden, wenn sie über entsprechende Berechtigungen in ihrem eigenen Konto verfügen.

> **Note**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen,

die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Wenn der Prinzipal in einer wichtigen Richtlinienanweisung ein AWS-Konto -Prinzipal ist, das als `arn:aws:iam::111122223333:root"` ausgedrückt wird, erteilt die Richtlinienanweisung keinem IAM-Prinzipal die Berechtigung. Stattdessen erteilt es die AWS-Konto Erlaubnis, IAM-Richtlinien zu verwenden, um die in der Schlüsselrichtlinie angegebenen Berechtigungen zu delegieren. (Ein Prinzipal im `arn:aws:iam::111122223333:root"`-Format repräsentiert nicht die AWS -Stammbenutzer des Kontos, trotz der Verwendung von „root“ in der Kontokennung. Der Kontoprinzipal repräsentiert jedoch das Konto und seine Administratoren, einschließlich des Account-Root-Benutzers.)

Handelt es sich bei dem Prinzipal um einen anderen Prinzipal AWS-Konto oder dessen Prinzipale, sind die Berechtigungen nur wirksam, wenn das Konto in der Region mit dem KMS-Schlüssel und der Schlüsselrichtlinie aktiviert ist. Informationen zu Regionen, die standardmäßig nicht aktiviert sind („Opt-In-Regionen“), finden Sie unter Verwalten von AWS-Regionen in Allgemeine AWS-Referenz.

Um einem anderen AWS-Konto oder seinen Prinzipalen die Verwendung eines KMS-Schlüssels zu ermöglichen, müssen Sie in einer Schlüsselrichtlinie und in einer IAM-Richtlinie im anderen Konto die entsprechenden Berechtigungen erteilen. Details hierzu finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

Aktion

(Erforderlich) Geben Sie die zugelassenen oder verweigerten API-Operationen an. Die `kms:Encrypt` Aktion entspricht beispielsweise dem Vorgang AWS KMS Verschlüsseln. Sie können mehr als eine Aktion in einer Richtlinienanweisung auflisten. Weitere Informationen finden Sie unter Berechtigungsreferenz.

> **Note**
>
> Wenn das erforderliche `Action`-Element in einer Schlüsselrichtlinienanweisung fehlt, hat die Richtlinienanweisung keine Wirkung. Eine wichtige Richtlinienanweisung ohne ein `Action` Element gilt für keinen KMS-Schlüssel.

Wenn das `Action` Element einer wichtigen Richtlinienanweisung fehlt, meldet die AWS KMS Konsole korrekt einen Fehler, aber die CreateKeyund PutKeyPolicy APIs sind erfolgreich, obwohl die Richtlinienanweisung unwirksam ist.

Ressource

(Erforderlich) In einer Schlüsselrichtlinie ist der Wert des Ressourcen-Elements `"*"`, was "dieser KMS-Schlüssel" bedeutet. Das Sternchen (`"*"`) identifiziert den KMS-Schlüssel, an den die Schlüsselrichtlinie angefügt ist.

> **Note**
>
> Wenn das erforderliche `Resource`-Element in einer Schlüsselrichtlinienanweisung fehlt, hat die Richtlinienanweisung keine Wirkung. Eine Schlüsselrichtlinie ohne `Resource`-Element gilt für keinen KMS-Schlüssel.
> Wenn das `Resource` Element einer wichtigen Grundsatzerklärung fehlt, meldet die AWS KMS Konsole zwar korrekt einen Fehler, aber der CreateKeyFehler ist PutKeyPolicy APIs erfolgreich, obwohl die Grundsatzerklärung unwirksam ist.

Bedingung

(Optional) Bedingungen geben an, welche Anforderungen erfüllt werden müssen, damit eine Schlüsselrichtlinie wirksam wird. AWS Kann anhand von Bedingungen den Kontext einer API-Anfrage auswerten, um festzustellen, ob die Richtlinienerklärung zutrifft oder nicht.

Um Bedingungen anzugeben, verwenden Sie vordefinierte Bedingungsschlüssel. AWS KMS unterstützt AWS globale Bedingungsschlüssel und AWS KMS Bedingungsschlüssel. Zur Unterstützung der attributebasierten Zugriffskontrolle (ABAC) AWS KMS stellt es Bedingungsschlüssel bereit, die den Zugriff auf einen KMS-Schlüssel auf der Grundlage von Tags und Aliasen steuern. Details hierzu finden Sie unter ABAC für AWS KMS.

Das Format für eine Bedingung lautet:

```
"Condition": {"condition operator": {"condition key": "condition value"}}
```

wie beispielsweise:

```
"Condition": {"StringEquals": {"kms:CallerAccount": "111122223333"}}
```

Weitere Informationen zur AWS Richtliniensyntax finden Sie unter AWS IAM-Richtlinienreferenz im IAM-Benutzerhandbuch.

Beispiel für eine Schlüsselrichtlinie

Das folgende Beispiel zeigt eine vollständige Schlüsselrichtlinie für einen KMS-Schlüssel mit symmetrischer Verschlüsselung. Sie können es als Nachschlagewerk verwenden, wenn Sie sich mit den wichtigsten Konzepten in diesem Kapitel befassen. Diese Schlüsselrichtlinie vereint die Beispiel-Richtlinienanweisungen des vorherigen Abschnitts Standardschlüsselrichtlinie in einer einzigen Schlüsselrichtlinie, wodurch Folgendes erreicht wird:

- Ermöglicht im Beispiel AWS-Konto 111122223333 vollen Zugriff auf den KMS-Schlüssel. Damit können das Konto und seine Administratoren, einschließlich des Root-Benutzers des Kontos (in Notfällen), IAM-Richtlinien verwenden, um den Zugriff auf den KMS-Schlüssel zu gestatten.
- Erlaubt der `ExampleAdminRole`-IAM-Rolle den KMS-Schlüssel zu verwalten.
- Erlaubt der `ExampleUserRole`-IAM-Rolle, den KMS-Schlüssel zu verwenden.

JSON

```
{
    "Id": "key-consolepolicy",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "AllowKeyAdministratorsAccess",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::111122223333:role/ExampleAdminRole"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion",
                "kms:RotateKeyOnDemand"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowKeyUse",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleUserRole"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowAttachmentPersistentResources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleUserRole"
            },
            "Action": [
                "kms:CreateGrant",
```

```
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

Standardschlüsselrichtlinie

Beim Erstellen eines KMS-Schlüssels können Sie die Schlüsselrichtlinie für den neuen KMS-Schlüssel angeben. Wenn Sie keinen angeben, AWS KMS erstellt er einen für Sie. Die verwendete Standardschlüsselrichtlinie unterscheidet sich je nachdem, ob Sie den Schlüssel in der AWS KMS Konsole erstellen oder die AWS KMS API verwenden. AWS KMS

Die Standard-Schlüsselrichtlinie beim programmgesteuerten Erstellen eines KMS-Schlüssels

Wenn Sie einen KMS-Schlüssel programmgesteuert mit der AWS KMS API erstellen (auch mithilfe von AWS SDKs, AWS Command Line Interfaceoder AWS -Tools für PowerShell) und Sie keine Schlüsselrichtlinie angeben, wird eine sehr einfache Standardschlüsselrichtlinie AWS KMS angewendet. Diese Standardschlüsselrichtlinie enthält eine Richtlinienanweisung, die demjenigen, der den KMS-Schlüssel besitzt AWS-Konto , die Erlaubnis erteilt, IAM-Richtlinien zu verwenden, um Zugriff auf alle AWS KMS Operationen mit dem KMS-Schlüssel zu gewähren. Weitere Informationen zu dieser Richtlinienanweisung finden Sie unter Erlaubt den Zugriff auf das AWS-Konto und aktiviert IAM-Richtlinien.

Standardschlüsselrichtlinie beim Erstellen eines KMS-Schlüssels mit AWS Management Console

Wenn Sie einen KMS-Schlüssel mit dem erstellen AWS Management Console, beginnt die Schlüsselrichtlinie mit der Richtlinienanweisung, die den Zugriff auf die IAM-Richtlinien ermöglicht AWS-Konto und diese aktiviert. Die Konsole fügt dann eine wichtige Administratoranweisung, eine Schlüsselbenutzeranweisung und (bei den meisten Schlüsseltypen) eine Anweisung hinzu, die es Prinzipalen ermöglicht, den KMS-Schlüssel mit anderen AWS Diensten zu verwenden. Sie können die Funktionen der AWS KMS Konsole verwenden, um die IAM-Benutzer und die

IAM-Rollen anzugeben und anzugeben, AWS-Konten wer Schlüsseladministratoren und welche Hauptbenutzer sind (oder beides).

Berechtigungen

- Erlaubt den Zugriff auf das AWS-Konto und aktiviert IAM-Richtlinien
- Erlaubt Schlüsseladministratoren die Verwaltung des KMS-Schlüssels
- Erlaubt Schlüsselbenutzern die Verwendung des KMS-Schlüssels
  - Erlaubt Schlüsselbenutzern die Verwendung eines KMS-Schlüssels für kryptografische Operationen
  - Erlaubt Schlüsselbenutzern die Verwendung des KMS-Schlüssel mit AWS -Services

Erlaubt den Zugriff auf das AWS-Konto und aktiviert IAM-Richtlinien

Die folgende standardmäßige Schlüsselrichtlinienanweisung ist kritisch.

- Sie gewährt AWS-Konto demjenigen, der den KMS-Schlüssel besitzt, vollen Zugriff auf den KMS-Schlüssel.

  Im Gegensatz zu anderen AWS Ressourcenrichtlinien erteilt eine AWS KMS Schlüsselrichtlinie nicht automatisch Berechtigungen für das Konto oder eine seiner Identitäten. Um Kontoadministratoren die Berechtigung zu erteilen, muss die Schlüsselrichtlinie eine explizite Erklärung enthalten, die diese Berechtigung wie diese bereitstellt.
- Zusätzlich zur Schlüsselrichtlinie kann das Konto IAM-Richtlinien verwenden, um den Zugriff auf den KMS-Schlüssel zu gestatten.

  Ohne diese Berechtigung sind IAM-Richtlinien, die den Zugriff auf den Schlüssel ermöglichen, wirkungslos, obwohl IAM-Richtlinien, die den Zugriff auf den Schlüssel verweigern, weiterhin wirksam sind.
- Es reduziert das Risiko, dass der Schlüssel unüberschaubar wird, indem es den Kontoadministratoren, einschließlich dem Account-Root-Benutzer, die nicht gelöscht werden kann, die Zugriffsberechtigung erteilt.

Die folgende wichtige Richtlinienanweisung ist die gesamte Standardschlüsselrichtlinie für KMS-Schlüssel, die programmgesteuert erstellt wurden. Dies ist die erste Richtlinienanweisung in der Standard-Schlüsselrichtlinie für KMS-Schlüssel, die in der AWS KMS Konsole erstellt wurde.

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:root"
   },
  "Action": "kms:*",
  "Resource": "*"
}
```

Erlaubt es IAM-Richtlinien, den Zugriff auf den KMS-Schlüssel zu steuern.

Die oben abgebildete Grundsatzerklärung gibt AWS-Konto demjenigen, der den Schlüssel besitzt, die Erlaubnis, IAM-Richtlinien sowie wichtige Richtlinien zu verwenden, um alle Aktionen (`kms:*`) für den KMS-Schlüssel zuzulassen.

Der Grundsatz in dieser wichtigen Richtlinienanweisung ist der Kontoprinzipal, der durch einen ARN in diesem Format repräsentiert wird: `arn:aws:iam::account-id:root`. Der Kontoprinzipal repräsentiert das AWS Konto und seine Administratoren.

Wenn der Prinzipal in einer Schlüsselrichtlinienausweisung der Kontoprinzipal ist, erteilt die Richtlinienerklärung keinen IAM-Identitäten die Berechtigung, den KMS-Schlüssel zu verwenden. Stattdessen erlaubt es dem Konto, IAM-Richtlinien zu verwenden, um die in der Richtlinienanweisung angegebenen Berechtigungen zu delegieren. Diese standardmäßige Schlüsselrichtlinienausweisung ermöglicht es dem Konto, IAM-Richtlinien zu verwenden, um Berechtigungen für alle Aktionen zu delegieren (`kms:*`) auf dem KMS-Schlüssel.

verringert das Risiko, dass der KMS-Schlüssel nicht mehr verwaltet werden kann.

Im Gegensatz zu anderen AWS Ressourcenrichtlinien erteilt eine AWS KMS Schlüsselrichtlinie dem Konto oder einem seiner Hauptbenutzer nicht automatisch Berechtigungen. Um jedem Auftraggeber die Berechtigung zu erteilen, einschließlich der-Kontoauftraggeber verwenden, müssen Sie eine Schlüsselrichtlinienanweisung verwenden, die die Berechtigung explizit bereitstellt. Sie müssen dem Kontoprinzipal oder einem Prinzipal keinen Zugriff auf den KMS-Schlüssel gewähren. Wenn Sie jedoch Zugriff auf den Kontoprinzipal gewähren, können Sie verhindern, dass der Schlüssel nicht überschaubar wird.

Angenommen, Sie erstellen eine Schlüsselrichtlinie, die nur einem Benutzer Zugriff auf den KMS-Schlüssel gewährt. Wenn Sie diesen Benutzer dann löschen, wird der Schlüssel unüberschaubar

und Sie müssen den AWS -Support kontaktieren, um wieder Zugriff auf den KMS-Schlüssel zu erhalten.

Mit der oben abgebildeten Grundsatzerklärung wird die Kontrolle über den Schlüssel zum Kontoprinzipal erteilt, der das Konto AWS-Konto und seine Administratoren, einschließlich des Root-Benutzers, repräsentiert. Der Account-Root-Benutzer ist der einzige Prinzipal, der nur gelöscht werden kann, wenn Sie den AWS-Konto löschen. Die Best Practices von IAM raten davon ab, im Namen des Account-Root-Benutzers zu handeln, außer im Notfall. Möglicherweise müssen Sie jedoch als Account-Root-Benutzer fungieren, wenn Sie alle anderen Benutzer und Rollen mit Zugriff auf den KMS-Schlüssel löschen.

Erlaubt Schlüsseladministratoren die Verwaltung des KMS-Schlüssels

Die von der Konsole erstellte Standard-Schlüsselrichtlinie erlaubt es Ihnen, IAM-Benutzer und -Rollen im Konto auszuwählen und diese zu Schlüsseladministratoren zu machen. Diese Anweisung wird Schlüsseladministratoren-Anweisung genannt. Schlüsseladministratoren sind dazu berechtigt, den KMS-Schlüssel zu verwalten, sie verfügen jedoch nicht über die Berechtigungen, den KMS-Schlüssel in kryptografischen Operationen zu verwenden. Sie können IAM-Benutzer und -Rollen zu der Liste der Schlüsseladministratoren hinzufügen, wenn Sie den KMS-Schlüssel in der Standard- oder Richtlinienansicht erstellen.

> **Warning**
>
> Da Schlüsseladministratoren berechtigt sind, die Schlüsselrichtlinie zu ändern und Zuweisungen zu erstellen, können sie sich selbst und anderen AWS KMS Berechtigungen gewähren, die in dieser Richtlinie nicht spezifiziert sind.
> Prinzipale, die über die Berechtigung zum Verwalten von Tags und Aliasen verfügen, können auch den Zugriff auf einen KMS-Schlüssel steuern. Details hierzu finden Sie unter ABAC für AWS KMS.

> **Note**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Das folgende Beispiel zeigt die Schlüsseladministratoren-Anweisung in der Standardansicht der AWS KMS -Konsole.

Nachfolgend sehen Sie ein Beispiel für eine Schlüsseladministratoren-Anweisung in der Richtlinienansicht der AWS KMS -Konsole. Diese Aussage von Schlüsseladministratoren bezieht sich auf einen KMS-Schlüssel mit symmetrischer Verschlüsselung für eine einzelne Region.

> **Note**
>
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter der Anweisungs-ID Schlüsseladministratoren hinzu. `"Allow access for Key Administrators"` Eine Änderung dieser Anweisungs-ID kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Anweisung vornehmen.

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS":"arn:aws:iam::111122223333:role/ExampleAdminRole"},
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:TagResource",
    "kms:UntagResource",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion",
    "kms:RotateKeyOnDemand"
  ],
  "Resource": "*"
}
```

Die standardmäßige Schlüsseladministratoren-Anweisung für den geläufigsten KMS-Schlüssel, ein KMS-Schlüssel mit symmetrischer Verschlüsselung für eine Region, erteilt die folgenden Berechtigungen. Ausführliche Informationen zu den einzelnen Berechtigungen finden Sie unter AWS KMS Berechtigungen.

Wenn Sie die AWS KMS Konsole verwenden, um einen KMS-Schlüssel zu erstellen, fügt die Konsole die von Ihnen angegebenen Benutzer und Rollen dem `Principal` Element in der Anweisung für wichtige Administratoren hinzu.

Viele dieser Berechtigungen enthalten das Platzhalterzeichen (`*`), sodass alle Berechtigungen zugelassen werden, die mit dem angegebenen Verb beginnen. Wenn neue API-Operationen AWS KMS hinzugefügt werden, dürfen Schlüsseladministratoren diese daher automatisch verwenden. Sie müssen Ihre Schlüsselrichtlinien nicht aktualisieren, um die neuen Operationen einzubeziehen. Wenn Sie Ihre Schlüsseladministratoren lieber auf bestimmte API-Operationen beschränken möchten, können Sie Ihre Schlüsselrichtlinie ändern.

`kms:Create*`

Erlaubt `kms:CreateAlias` und `kms:CreateGrant`. (Die `kms:CreateKey`-Berechtigung ist nur in einer IAM-Richtlinie gültig.)

`kms:Describe*`

Erlaubt `kms:DescribeKey`. Die Berechtigung `kms:DescribeKey` ist erforderlich, um die Seite mit den Schlüsseldetails für einen KMS-Schlüssel in der AWS Management Console anzuzeigen.

`kms:Enable*`

Erlaubt `kms:EnableKey`. Für KMS-Schlüssel mit symmetrischer Verschlüsselung erlaubt sie auch `kms:EnableKeyRotation`.

`kms:List*`

Erlaubt `kms:ListGrants`, `kms:ListKeyPolicies` und `kms:ListResourceTags`. (Die Berechtigungen `kms:ListAliases` und `kms:ListKeys`, die zum Anzeigen von KMS-Schlüsseln in der AWS Management Console erforderlich sind, sind nur in IAM-Richtlinien gültig.)

`kms:Put*`

Erlaubt `kms:PutKeyPolicy`. Diese Berechtigung erlaubt Schlüsseladministratoren, die Schlüsselrichtlinie für diesen KMS-Schlüssel zu ändern.

`kms:Update*`

Erlaubt `kms:UpdateAlias` und `kms:UpdateKeyDescription`. Bei multiregionalen Schlüsseln erlaubt sie `kms:UpdatePrimaryRegion` für diesen KMS-Schlüssel.

`kms:Revoke*`

Erlaubt `kms:RevokeGrant`, wodurch Schlüsseladministratoren eine Erteilung löschen können, auch wenn sie kein ausscheidender Prinzipal in der Erteilung sind.

`kms:Disable*`

Erlaubt `kms:DisableKey`. Für KMS-Schlüssel mit symmetrischer Verschlüsselung erlaubt sie auch `kms:DisableKeyRotation`.

`kms:Get*`

Erlaubt `kms:GetKeyPolicy` und `kms:GetKeyRotationStatus`. Für KMS-Schlüssel mit importiertem Schlüsselmaterial erlaubt sie `kms:GetParametersForImport`. Für asymmetrische KMS-Schlüssel erlaubt sie `kms:GetPublicKey`. Die Berechtigung `kms:GetKeyPolicy` ist erforderlich, um die Schlüsselrichtlinie eines KMS-Schlüssels in der AWS Management Console anzuzeigen.

`kms:Delete*`

Erlaubt `kms:DeleteAlias`. Für Schlüssel mit importiertem Schlüsselmaterial erlaubt sie `kms:DeleteImportedKeyMaterial`. Die Berechtigung `kms:Delete*` erlaubt Schlüsseladministratoren nicht das Löschen des KMS-Schlüssels (`ScheduleKeyDeletion`).

`kms:TagResource`

Erlaubt `kms:TagResource`, sodass Schlüsseladministratoren Tags zum KMS-Schlüssel hinzufügen können. Da Tags auch verwendet werden können, um den Zugriff auf den KMS-Schlüssel zu steuern, können Administratoren mit dieser Berechtigung den Zugriff auf den KMS-Schlüssel gewähren oder verweigern. Details hierzu finden Sie unter ABAC für AWS KMS.

`kms:UntagResource`

Erlaubt `kms:UntagResource`, sodass Schlüsseladministratoren Tags aus dem KMS-Schlüssel löschen können. Da Tags verwendet werden können, um den Zugriff auf den Schlüssel zu steuern, können Administratoren mit dieser Berechtigung den Zugriff auf den KMS-Schlüssel gewähren oder verweigern. Details hierzu finden Sie unter ABAC für AWS KMS.

`kms:ScheduleKeyDeletion`

Erlaubt `kms:ScheduleKeyDeletion`, sodass Schlüsseladministratoren diesen KMS-Schlüssel löschen können. Um diese Berechtigung zu löschen, deaktivieren Sie die Option Allow key administrators to delete this key (Schlüsseladministratoren das Löschen dieses Schlüssels erlauben).

`kms:CancelKeyDeletion`

Erlaubt `kms:CancelKeyDeletion`, sodass Schlüsseladministratoren das Löschen dieses KMS-Schlüssels abbrechen können. Um diese Berechtigung zu löschen, deaktivieren Sie die Option Allow key administrators to delete this key (Schlüsseladministratoren das Löschen dieses Schlüssels erlauben).

`kms:RotateKeyOnDemand`

Ermöglicht es Schlüsseladministratoren `kms:RotateKeyOnDemand`, bei Bedarf eine Rotation des Schlüsselmaterials in diesem KMS-Schlüssel durchzuführen.

AWS KMS fügt der Standardanweisung für Schlüsseladministratoren die folgenden Berechtigungen hinzu, wenn Sie Schlüssel für spezielle Zwecke erstellen.

`kms:ImportKeyMaterial`

Die Berechtigung `kms:ImportKeyMaterial` erlaubt es Schlüsseladministratoren, Schlüsselmaterial in den KMS-Schlüssel zu importieren. Diese Berechtigung ist nur in der Schlüsselrichtlinie enthalten, wenn Sie einen KMS-Schlüssel ohne Schlüsselmaterial erstellen.

`kms:ReplicateKey`

Diese `kms:ReplicateKey`Berechtigung ermöglicht es Schlüsseladministratoren, ein Replikat eines Primärschlüssels für mehrere Regionen in einer anderen Region zu erstellen. AWS Diese Berechtigung ist nur dann in der Schlüsselrichtlinie enthalten, wenn Sie einen multiregionalen Primär- oder Replikatschlüssel erstellen.

`kms:UpdatePrimaryRegion`

Die Berechtigung `kms:UpdatePrimaryRegion` erlaubt es Schlüsseladministratoren, einen multiregionalen Replikatschlüssel zu einem multiregionalen Primärschlüssel zu ändern. Diese Berechtigung ist nur dann in der Schlüsselrichtlinie enthalten, wenn Sie einen multiregionalen Primär- oder Replikatschlüssel erstellen.

Erlaubt Schlüsselbenutzern die Verwendung des KMS-Schlüssels

Die Standardschlüsselrichtlinie, die die Konsole für KMS-Schlüssel erstellt, ermöglicht es Ihnen, IAM-Benutzer und IAM-Rollen innerhalb des Kontos sowie externe AWS-Konten Benutzer auszuwählen und sie zu Schlüsselbenutzern zu machen.

Die Konsole fügt der Schlüsselrichtlinie für Schlüsselbenutzer zwei Richtlinienanweisungen hinzu.

- Direkte Verwendung des KMS-Schlüssels – Die erste Schlüsselrichtlinienanweisung gibt Schlüsselbenutzern die Berechtigung, den KMS-Schlüssel direkt für alle unterstützten kryptografischen Operationen für diesen KMS-Schlüsseltyp zu verwenden.
- Verwenden Sie den KMS-Schlüssel mit AWS Diensten — Die zweite Richtlinienanweisung gibt Schlüsselbenutzern die Erlaubnis, AWS Diensten, die in integriert sind, die Verwendung des KMS-Schlüssels in ihrem Namen zu gestatten, um Ressourcen wie Amazon S3 S3-Buckets und Amazon DynamoDB-Tabellen zu schützen. AWS KMS

Sie können IAM-Benutzer, IAM-Rollen und andere AWS-Konten zur Liste der Hauptbenutzer hinzufügen, wenn Sie den KMS-Schlüssel erstellen. Sie können auch die Liste der Schlüsselrichtlinien in der Standardansicht der Konsole wie in der folgenden Abbildung dargestellt bearbeiten. Die Standardansicht für Schlüsselrichtlinien finden Sie auf der Seite mit den Schlüsseldetails. Weitere Informationen darüber, wie Benutzern in anderen AWS-Konten Ländern die Verwendung des KMS-Schlüssels gestattet wird, finden Sie unter. Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben

> **Note**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Die Standard-Schlüsselbenutzer-Anweisung für einen symmetrischen Schlüssel für eine Region gewährt die folgenden Berechtigungen. Ausführliche Informationen zu den einzelnen Berechtigungen finden Sie unter AWS KMS Berechtigungen.

Wenn Sie die AWS KMS Konsole verwenden, um einen KMS-Schlüssel zu erstellen, fügt die Konsole die Benutzer und Rollen, die Sie angeben, dem `Principal` Element in den einzelnen Schlüsselbenutzeranweisungen hinzu.

> **Note**
>
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter den Anweisungskennungen `"Allow use of the key"` und `"Allow attachment of persistent resources"` Schlüsselbenutzer hinzu. Das Ändern dieser Anweisungskennungen kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Aussage vornehmen.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:role/ExampleRole",
    "arn:aws:iam::444455556666:root"
  ]},
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
},
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:role/ExampleRole",
    "arn:aws:iam::444455556666:root"
  ]},
  "Action": [
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
```

```
}
```

Erlaubt Schlüsselbenutzern die Verwendung eines KMS-Schlüssels für kryptografische Operationen

Schlüsselbenutzer haben die Berechtigung, den KMS-Schlüssel direkt in allen kryptografischen Operationen zu verwenden, die vom KMS-Schlüssel unterstützt werden. Sie können den DescribeKeyVorgang auch verwenden, um detaillierte Informationen über den KMS-Schlüssel in der AWS KMS Konsole oder mithilfe der AWS KMS API-Operationen abzurufen.

Standardmäßig fügt die AWS KMS Konsole der Standardschlüsselrichtlinie wichtige Benutzeranweisungen wie die in den folgenden Beispielen hinzu. Da sie unterschiedliche API-Operationen unterstützen, weichen die jeweiligen Aktionen in den Richtlinienanweisungen für KMS-Schlüssel mit symmetrischer Verschlüsselung, HMAC-KMS-Schlüssel, asymmetrische KMS-Schlüssel für die Verschlüsselung öffentlicher Schlüssel und asymmetrische KMS-Schlüssel für Signatur und Verifizierung geringfügig voneinander ab.

KMS-Schlüssel mit symmetrischer Verschlüsselung

Die Konsole fügt der Schlüsselrichtlinie für KMS-Schlüssel mit symmetrischer Verschlüsselung die folgende Anweisung hinzu.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey",
    "kms:Encrypt",
    "kms:GenerateDataKey*",
    "kms:ReEncrypt*"
  ],
  "Resource": "*"
}
```

HMAC-KMS-Schlüssel

Die Konsole fügt der Schlüsselrichtlinie für HMAC-KMS-Schlüssel die folgende Anweisung hinzu.

```
{
```

```
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:DescribeKey",
    "kms:GenerateMac",
    "kms:VerifyMac"
  ],
  "Resource": "*"
}
```

Asymmetrische KMS-Schlüssel für die Verschlüsselung öffentlicher Schlüssel

Die Konsole fügt die folgende Anweisung zur Schlüsselrichtlinie für asymmetrische KMS-Schlüssels mit der Schlüsselnutzung Encrypt and decrypt (Verschlüsseln und Entschlüsseln) hinzu.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:DescribeKey",
    "kms:GetPublicKey"
  ],
  "Resource": "*"
}
```

Asymmetrische KMS-Schlüssel für Signatur und Verifizierung

Die Konsole fügt die folgende Anweisung zur Schlüsselrichtlinie für asymmetrische KMS-Schlüssels mit der Schlüsselnutzung Sign and verify (Signieren und Überprüfen) hinzu.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
```

```
  "Action": [
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:Sign",
    "kms:Verify"
  ],
  "Resource": "*"
}
```

Asymmetrische KMS-Schlüssel zum Ableiten von gemeinsamen Geheimnissen

Die Konsole fügt der Schlüsselrichtlinie für asymmetrische KMS-Schlüssel die folgende Anweisung hinzu, wobei die Schlüsselvereinbarung als Schlüssel verwendet wird.

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:DeriveSharedSecret"
  ],
  "Resource": "*"
}
```

Die Aktionen in diesen Anweisungen geben den Schlüsselbenutzern die folgenden Berechtigungen.

`kms:Encrypt`

Erlaubt es Schlüsselbenutzern, Daten mit diesem KMS-Schlüssel zu verschlüsseln.

`kms:Decrypt`

Erlaubt es Schlüsselbenutzern, Daten mit diesem KMS-Schlüssel zu entschlüsseln.

`kms:DeriveSharedSecret`

Ermöglicht es Schlüsselbenutzern, gemeinsame Geheimnisse mit diesem KMS-Schlüssel abzuleiten.

`kms:DescribeKey`

Erlaubt Schlüsselbenutzern das Abrufen detaillierter Information über diesen KMS-Schlüssel, einschließlich der dazugehörigen IDs, des Erstellungsdatums und des Schlüsselstatus. Außerdem können die Hauptbenutzer Details zum KMS-Schlüssel in der AWS KMS Konsole anzeigen.

`kms:GenerateDataKey*`

Erlaubt Schlüsselbenutzern, einen symmetrischen Datenschlüssel oder ein asymmetrisches Datenschlüsselpaar für clientseitige kryptografische Operationen anzufordern. Die Konsole verwendet das Platzhalterzeichen *, um die Erlaubnis für die folgenden API-Operationen darzustellen: GenerateDataKey, GenerateDataKeyWithoutPlaintextGenerateDataKeyPair, und GenerateDataKeyPairWithoutPlaintext. Diese Berechtigungen gelten nur für die symmetrischen KMS-Schlüssel, die die Datenschlüssel verschlüsseln.

km: GenerateMac

Ermöglicht es Schlüsselbenutzern, einen HMAC-KMS-Schlüssel zu verwenden, um ein HMAC-Tag zu generieren.

km: GetPublicKey

Erlaubt Schlüsselbenutzern das Herunterladen des öffentlichen Schlüssels des asymmetrischen KMS-Schlüssels. Parteien, mit denen Sie diesen öffentlichen Schlüssel teilen, können Daten außerhalb von AWS KMS verschlüsseln. Diese Chiffretexte können jedoch nur durch Aufrufen der Produktion Decrypt in AWS KMS entschlüsselt werden.

km: * ReEncrypt

Erlaubt es Schlüsselbenutzern, Daten erneut zu verschlüsseln, die ursprünglich mit diesem KMS-Schlüssel verschlüsselt wurden, oder diesen KMS-Schlüssel zu verwenden, um zuvor verschlüsselte Daten erneut zu verschlüsseln. Der ReEncryptVorgang erfordert Zugriff sowohl auf Quell- als auch auf Ziel-KMS-Schlüssel. Um dies zu erreichen, können Sie die `kms:ReEncryptFrom`-Berechtigung für den Quell-KMS-Schlüssel und die `kms:ReEncryptTo`-Berechtigung für den Ziel-KMS-Schlüssel erlauben. Der Einfachheit halber erlaubt die Konsole jedoch `kms:ReEncrypt*` (mit dem Platzhalterzeichen `*`) für beide KMS-Schlüssel.

kms:Sign

Erlaubt es Schlüsselbenutzern, Nachrichten mit diesem KMS-Schlüssel zu signieren.

kms:Verify

Erlaubt es Schlüsselbenutzern, Signaturen mit diesem KMS-Schlüssel zu überprüfen.

km: VerifyMac

Ermöglicht es Schlüsselbenutzern, einen HMAC-KMS-Schlüssel zu verwenden, um ein HMAC-Tag zu verifizieren.

Erlaubt Schlüsselbenutzern die Verwendung des KMS-Schlüssel mit AWS -Services

Die standardmäßige Schlüsselrichtlinie in der Konsole gewährt Schlüsselbenutzern außerdem die Zugriffsberechtigungen, die sie zum Schutz ihrer Daten in AWS Diensten benötigen, die Grants verwenden. AWS Dienste verwenden häufig Zuschüsse, um spezifische und eingeschränkte Berechtigungen zur Verwendung eines KMS-Schlüssels zu erhalten.

Diese wichtige Richtlinienerklärung ermöglicht es dem Hauptbenutzer, Berechtigungen für den KMS-Schlüssel zu erstellen, einzusehen und zu widerrufen, aber nur, wenn die Anfrage für den Gewährungsvorgang von einem AWS Dienst stammt, in den der Dienst integriert ist AWS KMS. Die GrantIsFor AWSResource Richtlinienbedingung kms: erlaubt es dem Benutzer nicht, diese Grant-Operationen direkt aufzurufen. Wenn der Schlüsselbenutzer dies zulässt, kann ein AWS Dienst im Namen des Benutzers eine Gewährung einrichten, die es dem Dienst ermöglicht, den KMS-Schlüssel zum Schutz der Benutzerdaten zu verwenden.

Schlüsselbenutzer benötigen diese Erteilungs-Berechtigungen, um ihren KMS-Schlüssel mit integrierten Services zu verwenden, aber diese Berechtigungen sind nicht ausreichend. Schlüsselbenutzer benötigen außerdem die Berechtigung, die integrierten Services zu nutzen. Einzelheiten darüber, wie Sie Benutzern Zugriff auf einen AWS Dienst gewähren, der integriert werden kann AWS KMS, finden Sie in der Dokumentation zum integrierten Dienst.

```
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

Schlüsselbenutzer können diese Berechtigungen beispielsweise auf folgende Weise für den KMS-Schlüssel verwenden.

- Verwenden Sie diesen KMS-Schlüssel mit Amazon Elastic Block Store (Amazon EBS) und Amazon Elastic Compute Cloud (Amazon EC2), um ein verschlüsseltes EBS-Volume an eine EC2 Instance anzuhängen. Der Schlüsselbenutzer erteilt Amazon implizit die EC2 Erlaubnis, den KMS-Schlüssel zu verwenden, um das verschlüsselte Volume an die Instance anzuhängen. Weitere Informationen finden Sie unter So verwendet Amazon Elastic Block Store (Amazon EBS) AWS KMS.
- Verwenden Sie diesen KMS-Schlüssel zusammen mit Amazon Redshift, um einen verschlüsselten Cluster zu starten. Der Schlüsselbenutzer erteilt Amazon Redshift implizit die Berechtigung, den KMS-Schlüssel zu verwenden, um den verschlüsselten Cluster zu starten und verschlüsselte Snapshots zu erstellen. Weitere Informationen finden Sie unter So verwendet Amazon Redshift AWS KMS.
- Verwenden Sie diesen KMS-Schlüssel mit anderen AWS -Diensten, die in AWS KMS integriert sind und Berechtigungen zum Erstellen, Verwalten oder Verwenden verschlüsselter Ressourcen mit diesen Diensten verwenden.

Die Standard-Berechtigung ermöglicht es Schlüsselbenutzern, alle integrierten Services zu verwenden, die Zugriffserteilungen nutzen. Sie können jedoch eine benutzerdefinierte Schlüsselrichtlinie erstellen, die die Berechtigung auf bestimmte AWS Dienste beschränkt. Weitere Informationen erhalten Sie unter km: ViaService Bedingungsschlüssel.

Wichtige Richtlinien anzeigen

Sie können die wichtigsten Richtlinien für einen vom AWS KMS Kunden verwalteten Schlüssel oder einen Von AWS verwalteter Schlüsselin Ihrem Konto mithilfe der AWS KMS Konsole oder über den GetKeyPolicyVorgang in der AWS KMS API einsehen. Sie können diese Methoden nicht verwenden, um die Schlüsselrichtlinie eines KMS-Schlüssels in einem anderen AWS-Konto anzuzeigen.

Weitere Informationen zu den AWS KMS wichtigsten Richtlinien finden Sie unterDie wichtigsten Richtlinien in AWS KMS. Informationen zum Bestimmen, welche Benutzer und Rollen Zugriff auf ein KMS-Schlüssel haben, finden Sie unter the section called “Bestimmen des Zugriffs”.

Die AWS KMS Konsole verwenden

Autorisierte Benutzer können die Schlüsselrichtlinie für einen Von AWS verwalteter Schlüssel oder einen vom Kunden verwalteten Schlüssel auf der Registerkarte Key policy (Schlüsselrichtlinie) der AWS Management Console anzeigen.

Um die Schlüsselrichtlinie für einen KMS-Schlüssel in der anzeigen zu können AWS Management Console, benötigen Sie die GetKeyPolicy Berechtigungen kms: DescribeKey, kms: und kms:. ListAliases

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Schlüssel in Ihrem Konto anzuzeigen, das für Sie AWS erstellt und verwaltet wird, wählen AWS Sie im Navigationsbereich die Option Verwaltete Schlüssel aus. Zum Anzeigen der Schlüssel in Ihrem Konto, die Sie erstellen und verwalten, wählen Sie im Navigationsbereich Customer managed keys (Vom Kunden verwaltete Schlüssel) aus.
4. Wählen Sie in der Liste der KMS-Schlüssel den Alias oder die Schlüssel-ID des KMS-Schlüssels aus, den Sie untersuchen möchten.
5. Wählen Sie die Registerkarte Key policy (Schlüsselrichtlinie).

   Auf der Registerkarte Key policy (Schlüsselrichtlinie) wird möglicherweise das Schlüsselrichtliniendokument angezeigt. Dies ist eine Richtlinienansicht. In den Schlüsselrichtlinienanweisungen sehen Sie die Prinzipale, denen die Schlüsselrichtlinie Zugriff auf den KMS-Schlüssel gewährt hat, und Sie können die Aktionen anzeigen, die sie ausführen können.

   Das folgende Beispiel zeigt die Richtlinienansicht für die Standardschlüsselrichtlinie.

   Oder, wenn Sie den KMS-Schlüssel in der erstellt haben AWS Management Console, wird Ihnen die Standardansicht mit Abschnitten für Schlüsseladministratoren, Schlüssellöschung und Schlüsselbenutzer angezeigt. Wählen Sie zum Anzeigen der Schlüsselrichtliniendokuments Switch to policy view (Zur Richtlinienansicht wechseln).

   Das folgende Beispiel zeigt die Standardansicht für die Standardschlüsselrichtlinie.

Verwenden der AWS KMS API

Um die Schlüsselrichtlinie für einen KMS-Schlüssel in Ihrem abzurufen AWS-Konto, verwenden Sie den GetKeyPolicyVorgang in der AWS KMS API. Sie können diesen Vorgang nicht verwenden, um eine Schlüsselrichtlinie in einem anderen Konto anzuzeigen.

Im folgenden Beispiel wird der get-key-policyBefehl in AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jedes AWS SDK verwenden, um diese Anfrage zu stellen.

Beachten Sie, dass der Parameter `PolicyName` erforderlich ist, obwohl `default` der einzige gültige Wert ist. Außerdem fordert dieser Befehl die Ausgabe als Text anstelle im JSON-Format an, damit sie leichter zu lesen ist.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige aus Ihrem Konto.

```
$ aws kms get-key-policy --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --policy-name
 default --output text
```

Die Antwort sollte der folgenden ähneln, die die Standardschlüsselrichtlinie zurückgibt.

JSON

```
{
  "Version" : "2012-10-17",
  "Id" : "key-consolepolicy-3",
  "Statement" : [ {
  "Sid" : "EnableIAMUserPermissions",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : "kms:*",
    "Resource" : "*"
  } ]
}
```

Ändern Sie eine wichtige Richtlinie

Sie können die Schlüsselrichtlinie für einen KMS-Schlüssel in Ihrem ändern, AWS-Konto indem Sie die PutKeyPolicyOperation AWS Management Console oder verwenden. Sie können diese Methoden nicht verwenden, um die Schlüsselrichtlinie eines KMS-Schlüssels in einem anderen AWS-Konto zu ändern.

Beachten Sie beim Ändern einer Schlüsselrichtlinie die folgenden Regeln:

- Sie können die Schlüsselrichtlinie für einen Von AWS verwalteter Schlüssel oder einen vom Kunden verwalteten Schlüssel anzeigen, aber ändern können Sie nur die Schlüsselrichtlinie für einen vom Kunden verwalteten Schlüssel. Die Richtlinien von Von AWS verwaltete Schlüssel werden von dem AWS Dienst erstellt und verwaltet, der den KMS-Schlüssel in Ihrem Konto erstellt hat. Sie können die Schlüsselrichtlinie für einen AWS-eigener Schlüssel nicht anzeigen oder ändern.
- Sie können IAM-Benutzer, IAM-Rollen und AWS-Konten in der Schlüsselrichtlinie hinzufügen oder entfernen und die Aktionen ändern, die für diese Prinzipale zulässig oder verweigert sind. Weitere Informationen zu den Möglichkeiten, um Prinzipale und Berechtigungen in einer Schlüsselrichtlinie festzulegen, finden Sie unter Schlüsselrichtlinien.
- Sie können keine IAM-Gruppen zu einer Schlüsselrichtlinie hinzufügen, aber Sie können mehrere IAM-Benutzer und IAM-Rollen hinzufügen. Weitere Informationen finden Sie unter Mehreren IAM-Prinzipalen Zugriff auf einen KMS-Schlüssel gewähren.
- Wenn Sie externe Richtlinien AWS-Konten zu einer Schlüsselrichtlinie hinzufügen, müssen Sie auch IAM-Richtlinien in den externen Konten verwenden, um IAM-Benutzern, Gruppen oder Rollen in diesen Konten Berechtigungen zu erteilen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.
- Das sich ergebende Schlüssel-Richtliniendokument darf 32 KB (32.768 Bytes) nicht überschreiten.

So Ändern Sie eine Schlüsselrichtlinie:

Es gibt drei Möglichkeiten zum Ändern einer Schlüsselrichtlinie, die in den folgenden Abschnitten erläutert werden.

Themen

- Verwendung der AWS Management Console -Standardansicht
- Verwenden Sie die AWS Management Console Richtlinienansicht

- Verwenden der AWS KMS API

Verwendung der AWS Management Console -Standardansicht

Sie können die Konsole verwenden, um eine Schlüsselrichtlinie mit einer grafischen Benutzeroberfläche mit dem Namen Standardansicht zu ändern.

Wenn die folgenden Schritte nicht dem entsprechen, was Sie in der Konsole sehen, kann dies bedeuten, dass diese Schlüsselrichtlinie nicht durch die Konsole erstellt wurde. Es kann auch bedeuten, dass die Schlüsselrichtlinie auf eine Weise geändert wurde, die von der Standardansicht der Konsole nicht unterstützt wird. Befolgen Sie in diesem Fall die Schritte unter Verwenden Sie die AWS Management Console Richtlinienansicht oder Verwenden der AWS KMS API.

1. Zeigen Sie die Schlüsselrichtlinie für einen kundenverwalteten KMS-Schlüssel wie unter Die AWS KMS Konsole verwenden beschrieben an. (Sie können die wichtigsten Richtlinien von Von AWS verwaltete Schlüssel nicht ändern.)
2. Entscheiden Sie, was geändert werden soll.
   - Um Schlüsseladministratoren hinzuzufügen oder zu entfernen und festzulegen, ob sie den KMS-Schlüssel löschen dürfen oder nicht, verwenden Sie die Steuerelemente im Bereich Key Administrators (Schlüsseladministratoren) der Seite. Schlüsseladministratoren verwalten den KMS-Schlüssel. Dies umfasst die Aktivierung und Deaktivierung, das Festlegen der Schlüsselrichtlinie und das Aktivieren der Schlüsseldrehung.
   - Verwenden Sie die Steuerelemente im Abschnitt Hauptbenutzer der Seite, um Schlüsselbenutzer hinzuzufügen oder AWS-Konten zu entfernen und externen Benutzern die Verwendung des KMS-Schlüssels zu gestatten oder zu verbieten. Schlüsselbenutzer können den KMS-Schlüssel in kryptografischen Produktionen verwenden, wie beispielsweise der Verschlüsselung, Entschlüsselung, erneuten Verschlüsselung und Generierung von Datenschlüsseln.

Verwenden Sie die AWS Management Console Richtlinienansicht

Sie können die Konsole verwenden, um ein Schlüsselrichtliniendokument über die Richtlinienansicht der Konsole zu ändern.

1. Zeigen Sie die Schlüsselrichtlinie für einen kundenverwalteten KMS-Schlüssel wie unter Die AWS KMS Konsole verwenden beschrieben an. (Sie können die wichtigsten Richtlinien von nicht ändern Von AWS verwaltete Schlüssel.)

2. Wählen Sie im Abschnitt Key Policy (Schlüsselrichtlinie) die Option Switch to policy view (Zur Richtlinienansicht wechseln) aus.
3. Wählen Sie Bearbeiten aus.
4. Entscheiden Sie, was geändert werden soll.
   - Um einen neuen Kontoauszug hinzuzufügen, wählen Sie Neuen Kontoauszug hinzufügen. Anschließend können Sie die Aktionen, Grundsätze und Bedingungen für Ihre neue wichtige Grundsatzerklärung aus den im Statement Builder-Menü aufgelisteten Optionen auswählen oder die Elemente der Grundsatzerklärung manuell eingeben.
   - Um eine Aussage aus Ihrer wichtigsten Richtlinie zu entfernen, wählen Sie die Erklärung aus und klicken Sie dann auf Entfernen. Überprüfen Sie die ausgewählte Richtlinienerklärung und bestätigen Sie, dass Sie sie entfernen möchten. Wenn Sie entscheiden, dass Sie mit dem Entfernen der Erklärung nicht fortfahren möchten, wählen Sie Abbrechen.
   - Um eine bestehende wichtige Richtlinienerklärung zu bearbeiten, wählen Sie die Erklärung aus. Anschließend können Sie das Statement Builder-Bedienfeld verwenden, um bestimmte Elemente auszuwählen, die Sie ändern möchten, oder die Aussage manuell bearbeiten.
5. Wählen Sie Änderungen speichern aus.

Verwenden der AWS KMS API

Sie können den PutKeyPolicyVorgang verwenden, um die Schlüsselrichtlinie eines KMS-Schlüssels in Ihrem zu ändern AWS-Konto. Sie können diese API nicht für einen KMS-Schlüssel in einem anderen AWS-Konto verwenden.

1. Verwenden Sie den GetKeyPolicyVorgang, um das vorhandene wichtige Richtliniendokument abzurufen, und speichern Sie das wichtige Richtliniendokument anschließend in einer Datei. Beispielcode in mehreren Programmiersprachen finden Sie unter Verwendung `GetKeyPolicy` mit einem AWS SDK oder CLI.
2. Öffnen Sie das Schlüsselrichtliniendokument in Ihrem bevorzugten Texteditor, bearbeiten Sie das Schlüsselrichtliniendokument und speichern Sie dann die Datei.
3. Verwenden Sie den PutKeyPolicyVorgang, um das aktualisierte Dokument mit den wichtigsten Richtlinien auf den KMS-Schlüssel anzuwenden. Beispielcode in mehreren Programmiersprachen finden Sie unter Verwendung `PutKeyPolicy` mit einem AWS SDK oder CLI.

Ein Beispiel für das Kopieren einer Schlüsselrichtlinie von einem KMS-Schlüssel in einen anderen finden Sie in der AWS CLI Befehlsreferenz. GetKeyPolicy

Berechtigungen für AWS Dienste in wichtigen Richtlinien

Viele AWS Dienste verwenden sie AWS KMS keys , um die von ihnen verwalteten Ressourcen zu schützen. Wenn ein Service AWS-eigene Schlüssel oder Von AWS verwaltete Schlüssel verwendet, legt der Service die Schlüsselrichtlinien für diese KMS-Schlüssel fest und verwaltet sie.

Wenn Sie jedoch einen vom Kunden verwalteten Schlüssel mit einem AWS -Service verwenden, legen Sie die Schlüsselrichtlinie fest und verwalten sie. Diese Schlüsselrichtlinie muss dem Service die Mindestberechtigungen gewähren, die er benötigt, um die Ressource in Ihrem Namen zu schützen. Wir empfehlen, dem Prinzip der geringsten Berechtigung zu folgen: Gewähren Sie dem Service nur die Berechtigungen, die er benötigt. Dazu sollten Sie ermitteln, welche Berechtigungen der Service benötigt, und die globalen Bedingungsschlüssel von AWS sowie die Bedingungsschlüssel von AWS KMS verwenden, um die Berechtigungen detaillierter festzulegen.

Um die Berechtigungen zu ermitteln, die der Service für einen vom Kunden verwalteten Schlüssel benötigt, lesen Sie in der Verschlüsselungsdokumentation für den Service nach. Informationen zu den Berechtigungen, die Amazon Elastic Block Store (Amazon EBS) benötigt, finden Sie beispielsweise unter Berechtigungen für IAM-Benutzer im EC2 Amazon-Benutzerhandbuch und im EC2 Amazon-Benutzerhandbuch. Die Berechtigungen, die Secrets Manager benötigt, finden Sie unter Autorisieren der Nutzung des KMS-Schlüssels im AWS Secrets Manager -Benutzerhandbuch.

Verwenden von IAM-Richtlinien mit AWS KMS

Sie können IAM-Richtlinien zusammen mit wichtigen Richtlinien, Zuschüssen und VPC-Endpunktrichtlinien verwenden, um den Zugriff auf Ihr AWS KMS keys IN zu kontrollieren. AWS KMS

> **Note**
>
> Um mit einer IAM-Richtlinie den Zugriff auf einen KMS-Schlüssel zu steuern, muss die Schlüsselrichtlinie für den KMS-Schlüssel dem Konto die Berechtigung erteilen, IAM-Richtlinien zu nutzen. Insbesondere muss die Schlüsselrichtlinie die Richtlinienanweisung enthalten, die IAM-Richtlinien aktiviert.
> In diesem Abschnitt wird erklärt, wie Sie mithilfe von IAM-Richtlinien den Zugriff auf Vorgänge steuern können. AWS KMS Weitere Informationen zu IAM finden Sie im IAM-Benutzerhandbuch.

Alle KMS-Schlüssel müssen eine Schlüsselrichtlinie haben. IAM-Richtlinien sind optional. Um mit einer IAM-Richtlinie den Zugriff auf einen KMS-Schlüssel zu steuern, muss die Schlüsselrichtlinie für den KMS-Schlüssel dem Konto die Berechtigung erteilen, IAM-Richtlinien zu nutzen. Insbesondere muss die Schlüsselrichtlinie die Richtlinienanweisung enthalten, die IAM-Richtlinien aktiviert.

IAM-Richtlinien können den Zugriff auf jeden AWS KMS Vorgang steuern. Im Gegensatz zu wichtigen Richtlinien können IAM-Richtlinien den Zugriff auf mehrere KMS-Schlüssel steuern und Berechtigungen für den Betrieb mehrerer verwandter AWS Dienste bereitstellen. IAM-Richtlinien sind jedoch besonders nützlich, um den Zugriff auf Vorgänge zu kontrollieren, z. B. solche CreateKey, die nicht durch eine Schlüsselrichtlinie gesteuert werden können, da sie keinen bestimmten KMS-Schlüssel beinhalten.

Wenn Sie AWS KMS über einen Amazon Virtual Private Cloud (Amazon VPC) -Endpunkt zugreifen, können Sie auch eine VPC-Endpunktrichtlinie verwenden, um den Zugriff auf Ihre AWS KMS Ressourcen zu beschränken, wenn Sie den Endpunkt verwenden. Wenn Sie beispielsweise den VPC-Endpunkt verwenden, gestatten Sie möglicherweise nur den Prinzipalen in Ihrem AWS-Konto Zugriff auf Ihre vom Kunden verwalteten Schlüssel. Einzelheiten finden Sie unter VPC-Endpunktrichtlinien.

Hilfe beim Schreiben und Formatieren eines JSON-Richtliniendokuments finden Sie in der IAM-JSON-Richtlinienreferenz im IAM-Benutzerhandbuch.

Sie können IAM-Richtlinien auf folgende Weisen verwenden:

- Fügen Sie einer Rolle eine Berechtigungsrichtlinie für Verbund- oder kontoübergreifende Berechtigungen hinzu — Sie können einer IAM-Rolle eine IAM-Richtlinie zuordnen, um einen Identitätsverbund zu aktivieren, kontoübergreifende Berechtigungen zuzulassen oder Anwendungen, die auf Instances ausgeführt werden, Berechtigungen zu erteilen. EC2 Weitere Informationen zu den verschiedenen Anwendungsfällen für IAM-Rollen finden Sie unter IAM-Rollen im IAM-Benutzerhandbuch.
- Anfügen einer Berechtigungsrichtlinie an einen Benutzer oder ein Gruppe – Sie können eine Richtlinie anfügen, die es einem Benutzer oder einer Gruppe von Benutzern erlaubt, AWS KMS -Operationen aufzurufen. In den bewährten Methoden von IAM wird jedoch empfohlen, nach Möglichkeit Identitäten mit temporären Anmeldeinformationen, z. B. IAM-Rollen, zu verwenden.

Das folgende Beispiel zeigt eine IAM-Richtlinie mit Berechtigungen. AWS KMS Diese Richtlinie erlaubt es den IAM-Identitäten, denen sie angefügt ist, eine Liste aller KMS-Schlüssel und -Aliasen abzurufen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
}
```

Wie alle IAM-Richtlinien enthält diese Richtlinie kein `Principal`-Element. Wenn Sie einer IAM-Identität eine IAM-Richtlinie anfügen, erhält diese Identität die in der Richtlinie angegebenen Berechtigungen.

Eine Tabelle mit allen AWS KMS API-Aktionen und den Ressourcen, für die sie gelten, finden Sie unterBerechtigungsreferenz.

Mehreren IAM-Prinzipalen Zugriff auf einen KMS-Schlüssel gewähren

IAM-Gruppen sind keine gültigen Prinzipale in einer Schlüsselrichtlinie. Um mehreren Benutzern und Rollen den Zugriff auf einen KMS-Schlüssel zu erlauben, führen Sie einen der folgenden Schritte aus:

- Verwenden Sie eine IAM-Rolle als Prinzipal in der Schlüsselrichtlinie. Je nach Bedarf können mehrere autorisierte Benutzer die Rolle übernehmen. Weitere Informationen finden Sie unter IAM-Rollen im IAM-Benutzerhandbuch.

  Sie können zwar mehrere IAM-Benutzer in einer Schlüsselrichtlinie auflisten, diese Vorgehensweise wird jedoch nicht empfohlen, da Sie die Schlüsselrichtlinie jedes Mal aktualisieren müssen, wenn sich die Liste der autorisierten Benutzer ändert. Bewährte IAM-Methoden raten außerdem von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
- Verwenden Sie eine IAM-Richtlinie, um einer IAM-Gruppe eine Berechtigung zu erteilen. Stellen Sie dazu sicher, dass die Schlüsselrichtlinie die Anweisung enthält, die es IAM-Richtlinien

ermöglicht, den Zugriff auf den KMS-Schlüssel zu erlauben, eine IAM-Richtlinie zu erstellen, die den Zugriff auf den KMS-Schlüssel erlaubt, diese Richtlinie dann einer IAM-Gruppe anzuhängen, die die autorisierten IAM-Benutzer enthält. Bei diesem Ansatz müssen Sie keine Richtlinien ändern, wenn sich die Liste der autorisierten Benutzer ändert. Stattdessen müssen Sie einfach nur die Benutzer zu der entsprechenden IAM-Gruppe hinzufügen oder daraus entfernen. Weitere Informationen finden Sie unter IAM-Benutzergruppen im IAM-Benutzerhandbuch.

Weitere Informationen darüber, wie AWS KMS wichtige Richtlinien und IAM-Richtlinien zusammenarbeiten, finden Sie unterProblembehandlung bei Berechtigungen AWS KMS.

Bewährte Methoden für IAM-Richtlinien

Die Sicherung des Zugriffs auf AWS KMS keys ist entscheidend für die Sicherheit all Ihrer AWS Ressourcen. KMS-Schlüssel werden verwendet, um viele der sensibelsten Ressourcen in Ihrem AWS-Konto zu schützen. Nehmen Sie sich Zeit, um die wichtigsten Richtlinien, IAM-Richtlinien, Zuschüsse und VPC-Endpunktrichtlinien zu entwerfen, die den Zugriff auf Ihre KMS-Schlüssel steuern.

Wenden Sie in IAM-Richtlinienanweisungen, die den Zugriff auf KMS-Schlüssel steuern, das Prinzip der geringsten Berechtigung an. Geben Sie IAM-Prinzipalen nur die Berechtigungen ein, die sie nur für die KMS-Schlüssel benötigen, die sie verwenden oder verwalten müssen.

Die folgenden bewährten Methoden gelten für IAM-Richtlinien, die den Zugriff auf AWS KMS Schlüssel und Aliase steuern. Eine allgemeine Anleitung zu bewährten IAM-Richtlinien finden Sie unter Bewährte Methoden zur Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden von Schlüsselrichtlinien

Stellen Sie nach Möglichkeit Berechtigungen, die sich auf einen KMS-Schlüssel auswirken, in einer Schlüsselrichtlinie bereit anstatt in einer IAM-Richtlinie, die auf viele KMS-Schlüssel angewendet werden kann, einschließlich derjenigen in anderen AWS-Konten. Dies ist besonders wichtig für vertrauliche Berechtigungen wie kms: PutKeyPolicy und kms:, ScheduleKeyDeletion aber auch für kryptografische Operationen, die bestimmen, wie Ihre Daten geschützt werden.

Beschränken Sie die Erlaubnis CreateKey

Erteilen Sie nur den Prinzipalen die Erlaubnis, Schlüssel (kms: CreateKey) zu erstellen, die sie benötigen. Prinzipale, die einen KMS-Schlüssel erstellen, legen auch seine Schlüsselrichtlinie fest, damit sie sich selbst und anderen die Berechtigung zur Verwendung und Verwaltung der von ihnen erstellten KMS-Schlüssel erteilen können. Wenn Sie diese Berechtigung erlauben, sollten

Sie sie vielleicht mithilfe von -Richtlinienbedingungen beschränken. Sie können beispielsweise die KeySpec Bedingung kms: verwenden, um die Erlaubnis auf KMS-Schlüssel mit symmetrischer Verschlüsselung zu beschränken.

Angeben von KMS-Schlüsseln in einer IAM-Richtlinie

Geben Sie als bewährte Methode den Schüssel-ARN jedes KMS-Schlüssels an, für den die Berechtigung im `Resource`-Element der Richtlinienanweisung gilt. Diese Methode beschränkt die Berechtigung auf die KMS-Schlüssel, die vom Prinzipal benötigt werden. Dieses `Resource`-Element listet beispielsweise nur den KMS-Schlüssel auf, den der Prinzipal verwenden muss.

```
"Resource": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
]
```

Wenn die Angabe von KMS-Schlüsseln nicht praktikabel ist, verwenden Sie einen `Resource` Wert, der den Zugriff auf KMS-Schlüssel in einer vertrauenswürdigen AWS-Konto Region einschränkt, z. B. `arn:aws:kms:`*`region`*`:`*`account`*`:key/*` Oder beschränken Sie den Zugriff auf KMS-Schlüssel in allen Regionen (*) einer vertrauenswürdigen Person AWS-Konto, z. B. `arn:aws:kms:*:`*`account`*`:key/*`

Sie können keine Schlüssel-ID, Aliasnamen oder Alias-ARN verwenden, um einen KMS-Schlüssel in dem `Resource`-Feld einer IAM-Richtlinie darzustellen. Wenn Sie einen Alias-ARN angeben, gilt die Richtlinie für den Alias und nicht für den KMS-Schlüssel. Weitere allgemeine Informationen zu IAM-Richtlinien für Aliase finden Sie unter Steuern des Zugriffs auf Aliasse

Vermeiden Sie "Resource": "*" in einer IAM-Richtlinie

Verwenden Sie Platzhalterzeichen (*) umsichtig. In einer Schlüsselrichtlinie stellt das Platzhalterzeichen in dem `Resource`-Element den KMS-Schlüssel dar, an den die Schlüsselrichtlinie angefügt ist. In einer IAM-Richtlinie wendet jedoch nur ein Platzhalterzeichen im `Resource` Element (`"Resource": "*"`) die Berechtigungen auf alle KMS-Schlüssel an, zu deren Verwendung AWS-Konten das Konto des Prinzipalbenutzers berechtigt ist. Dies kann KMS-Schlüssel in anderen Bereichen AWS-Konten sowie KMS-Schlüssel im Konto des Prinzipals umfassen.

Um beispielsweise einen KMS-Schlüssel in einem anderen zu verwenden AWS-Konto, benötigt ein Principal die Genehmigung durch die Schlüsselrichtlinie des KMS-Schlüssels im externen Konto und durch eine IAM-Richtlinie in seinem eigenen Konto. Angenommen, ein beliebiges Konto erteilte Ihrem AWS-Konto die kms:Decrypt-Berechtigung für seine KMS-Schlüssel.

In diesem Fall würde eine IAM-Richtlinie in Ihrem Konto, die einer Rolle die `kms:Decrypt`-Berechtigung für alle KMS-Schlüssel (`"Resource": "*"`) erteilt, den IAM-Teil der Anforderung erfüllen. Daher können Prinzipale, die diese Rolle übernehmen können, nun Chiffretexte mit dem KMS-Schlüssel im nicht-vertrauenswürdigen Konto entschlüsseln. Einträge für ihre Operationen werden in den CloudTrail Protokollen beider Konten angezeigt.

Vermeiden Sie insbesondere die Verwendung von `"Resource": "*"` in einer Richtlinienanweisung, die die folgenden API-Operationen zulässt. Diese Operationen können für KMS-Schlüssel in anderen Ländern aufgerufen werden AWS-Konten.

- DescribeKey
- GetKeyRotationStatus
- Kryptografische Operationen (Verschlüsseln, Entschlüsseln,,, GenerateDataKey,, GenerateDataKeyPair, GenerateDataKeyWithoutPlaintextGenerateDataKeyPairWithoutPlaintext, Signieren GetPublicKeyReEncrypt, Überprüfen)
- CreateGrant, ListGrants, ListRetirableGrants, RetireGrant, RevokeGrant

Wann "Resource": "*" verwendet werden sollte

Verwenden Sie in einer IAM-Richtlinie ein Platzhalterzeichen im `Resource`-Element nur für Berechtigungen, die es erfordern. Nur die folgenden Berechtigungen erfordern das `"Resource": "*"`-Element.

- km: CreateKey
- km: GenerateRandom
- km: ListAliases
- km: ListKeys
- Berechtigungen für benutzerdefinierte Schlüsselspeicher wie kms: CreateCustomKeyStore und kms: ConnectCustomKeyStore.

> **Note**
>
> Berechtigungen für Aliasoperationen (kms: CreateAlias, kms: UpdateAlias, kms: DeleteAlias) müssen an den Alias und den KMS-Schlüssel angehängt werden. Sie können die `"Resource": "*"` in einer IAM-Richtlinie verwenden, um die Aliase und die KMS-Schlüssel darzustellen, oder geben Sie die Aliase und KMS-Schlüssel im `Resource`-Element an. Beispiele finden Sie unter Steuern des Zugriffs auf Aliasse.

Die Beispiele in diesem Thema enthalten weitere Informationen und Anleitungen zum Entwerfen von IAM-Richtlinien für KMS-Schlüssel. Bewährte IAM-Methoden für alle AWS Ressourcen finden Sie unter Bewährte Sicherheitsmethoden in IAM im IAM-Benutzerhandbuch.

Angeben von KMS-Schlüsseln in IAM-Richtlinienanweisungen

Sie können eine IAM-Richtlinie verwenden, um einem Prinzipal die Verwendung oder Verwaltung von KMS-Schlüsseln zu erlauben. KMS-Schlüssel werden in dem `Resource`-Element der Richtlinienanweisung angegeben.

- Um einen KMS-Schlüssel in einer IAM-Richtlinienanweisung anzugeben, müssen Sie dessen Schüssel-ARN verwenden. Sie können keine Schlüssel-ID und keinen Aliasnamen oder Alias-ARN verwenden, um einen KMS-Schlüssel in einer IAM-Richtlinienanweisung zu identifizieren.

  Zum Beispiel: „`Resource": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`“

  Um den Zugriff auf einen KMS-Schlüssel anhand seiner Aliase zu steuern, verwenden Sie die Bedingungsschlüssel kms: RequestAlias oder kms:. ResourceAliases Details hierzu finden Sie unter ABAC für AWS KMS.

  Verwenden Sie einen Alias-ARN nur als Ressource in einer Richtlinienanweisung, die den Zugriff auf Aliasoperationen wie CreateAliasUpdateAlias, oder steuert DeleteAlias. Details hierzu finden Sie unter Steuern des Zugriffs auf Aliasse.
- Um mehrere KMS-Schlüssel im Konto und in der Region anzugeben, verwenden Sie Platzhalterzeichen (*) in den Regions- oder Ressourcen-ID-Positionen des Schlüssel-ARN.

  Um beispielsweise alle KMS-Schlüssel in der Region USA West (Oregon) eines Kontos anzugeben, verwenden Sie "`Resource": "arn:aws:kms:us-west-2:111122223333:key/*`". Um alle KMS-Schlüssel in allen Regionen des Kontos anzugeben, verwenden Sie "`Resource": "arn:aws:kms:*:111122223333:key/*`".
- Um alle KMS-Schlüssel darzustellen, verwenden Sie nur ein Platzhalterzeichen ("`*`"). Verwenden Sie dieses Format für Operationen, die keinen bestimmten KMS-Schlüssel verwenden, nämlich CreateKeyGenerateRandom, ListAliases, und ListKeys.

Wenn Sie Ihre Richtlinienanweisungen schreiben, ist es eine bewährte Methode, nur die KMS-Schlüssel anzugeben, die der Prinzipal verwenden muss, anstatt ihm Zugriff auf alle KMS-Schlüssel zu gewähren.

Die folgende IAM-Richtlinienanweisung ermöglicht es dem Principal beispielsweise DescribeKey, die GenerateDataKey, Decrypt-Operationen nur für die KMS-Schlüssel aufzurufen, die im `Resource` Element der Richtlinienanweisung aufgeführt sind. Durch die bewährte Methode, KMS-Schlüssel nach dem Schlüssel-ARN anzugeben, wird sichergestellt, dass die Berechtigungen nur auf die angegebenen KMS-Schlüssel beschränkt sind.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": [
     "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
     "arn:aws:kms:us-
west-2:111122223333:key/01234abcd-12ab-34cd-56ef-1234567890ab"
    ]
  }
}
```

Um die Berechtigung auf alle KMS-Schlüssel in einem bestimmten vertrauenswürdigen Bereich anzuwenden AWS-Konto, können Sie Platzhalterzeichen (*) an den Positionen Region und Schlüssel-ID verwenden. Beispielsweise erlaubt die folgende Richtlinienanweisung es dem Prinzipal, die angegebenen Operationen für alle KMS-Schlüssel in zwei vertrauenswürdigen Beispielkonten aufzurufen.

JSON

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
      "kms:GenerateDataKeyPair"
    ],
    "Resource": [
      "arn:aws:kms:*:111122223333:key/*",
      "arn:aws:kms:*:444455556666:key/*"
    ]
  }
}
```

Sie können auch nur ein Platzhalterzeichen (`"*"`) in dem `Resource`-Element verwenden. Da es den Zugriff auf alle KMS-Schlüssel erlaubt, die das Konto nutzen darf, wird es in erster Linie für Operationen ohne einen bestimmten KMS-Schlüssel und für `Deny`-Anweisungen empfohlen. Sie können es auch in Richtlinienanweisungen verwenden, die nur weniger vertrauliche schreibgeschützte Operationen erlauben. Um festzustellen, ob ein AWS KMS Vorgang einen bestimmten KMS-Schlüssel beinhaltet, suchen Sie in der Spalte Ressourcen der Tabelle unter nach dem KMS-Schlüsselwert. the section called “Berechtigungsreferenz”

Die folgende Richtlinienanweisung nutzt einen `Deny`-Effekt, um Prinzipalen die Nutzung der angegebenen Operationen auf irgendeinem KMS-Schlüssel zu verweigern. Sie verwendet ein Platzhalterzeichen in dem `Resource`-Element, um alle KMS-Schlüssel darzustellen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [
      "kms:CreateKey",
      "kms:PutKeyPolicy",
      "kms:CreateGrant",
      "kms:ScheduleKeyDeletion"
    ],
    "Resource": "*"
```

```
  }
}
```

Die folgende Richtlinienanweisung verwendet nur ein Platzhalterzeichen, um alle KMS-Schlüssel darzustellen. Sie erlaubt jedoch nur weniger sensible schreibgeschützte Operationen und Operationen, die nicht für einen bestimmten KMS-Schlüssel gelten.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:CreateKey",
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:ListResourceTags"
    ],
    "Resource": "*"
  }
}
```

Beispiele für IAM-Richtlinien

In diesem Abschnitt finden Sie Beispiele für IAM-Richtlinien, die Berechtigungen für diverse AWS KMS -Aktionen gewähren.

> **Important**
>
> Einige der Berechtigungen in den folgenden Richtlinien sind nur erlaubt, wenn sie auch in der Schlüsselrichtlinie des KMS-Schlüssels erlaubt werden. Weitere Informationen finden Sie unter Berechtigungsreferenz.

Hilfe beim Schreiben und Formatieren eines JSON-Richtliniendokuments finden Sie in der IAM-JSON-Richtlinienreferenz im IAM-Benutzerhandbuch.

Beispiele

- Erlauben Sie einem Benutzer, KMS-Schlüssel in der AWS KMS Konsole anzuzeigen
- Einem Benutzer das Erstellen von KMS-Schlüsseln erlauben
- Ermöglicht einem Benutzer das Verschlüsseln und Entschlüsseln mit einem beliebigen KMS-Schlüssel in einem bestimmten AWS-Konto
- Erlaubt einem Benutzer das Verschlüsseln und Entschlüsseln mit einem beliebigen KMS-Schlüssel in einer bestimmten Region AWS-Konto
- Einem Benutzer das Verschlüsseln und Entschlüsseln mit bestimmten KMS-Schlüssel erlauben
- Einen Benutzer am Deaktivieren oder Löschen von KMS-Schlüsseln hindern

Erlauben Sie einem Benutzer, KMS-Schlüssel in der AWS KMS Konsole anzuzeigen

Die folgende IAM-Richtlinie ermöglicht Benutzern nur Lesezugriff auf die Konsole. AWS KMS Benutzer mit diesen Berechtigungen können alle KMS-Schlüssel in ihrer Datenbank einsehen AWS-Konto, sie können jedoch keine KMS-Schlüssel erstellen oder ändern.

Um KMS-Schlüssel auf den Seiten Von AWS verwaltete Schlüsselund Vom Kunden verwaltete Schlüssel anzeigen zu können, benötigen Prinzipale die GetResources Berechtigungen kms: ListAliases, kms: und tag:, auch wenn die Schlüssel keine Tags oder Aliase haben. ListKeys Die verbleibenden Berechtigungen, insbesondere kms: DescribeKey, sind erforderlich, um optionale Spalten und Daten der KMS-Schlüsseltabelle auf den KMS-Schlüsseldetailseiten anzuzeigen. Die ListRoles Berechtigungen iam: ListUsers und iam: sind erforderlich, um die Schlüsselrichtlinie in der Standardansicht fehlerfrei anzuzeigen. Um Daten auf der Seite Benutzerdefinierte Schlüsselspeicher und Details zu KMS-Schlüsseln in benutzerdefinierten Schlüsselspeichern anzuzeigen, benötigen Prinzipale außerdem die kms: -Berechtigung. DescribeCustomKeyStores

Wenn Sie den Konsolenzugriff eines Benutzers auf bestimmte KMS-Schlüssel beschränken, zeigt die Konsole einen Fehler für jeden KMS-Schlüssel an, der nicht sichtbar ist.

Diese Richtlinie umfasst zwei Richtlinienanweisungen. Das `Resource`-Element in der ersten Richtlinienanweisung gewährt die angegebenen Berechtigungen für alle KMS-Schlüssel in allen Regionen des Beispiel- AWS-Konto. Konsolenbetrachter benötigen keinen zusätzlichen Zugriff, da die AWS KMS -Konsole nur KMS-Schlüssel im Konto des Prinzipals anzeigt. Dies gilt auch dann, wenn sie berechtigt sind, KMS-Schlüssel in anderen AWS-Konten Datenbanken einzusehen. Für die verbleibenden Berechtigungen AWS KMS und die IAM-Berechtigungen ist ein `"Resource": "*"` Element erforderlich, da sie für keinen bestimmten KMS-Schlüssel gelten.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnlyAccessForAllKMSKeysInAccount",
      "Effect": "Allow",
      "Action": [
        "kms:GetPublicKey",
        "kms:GetKeyRotationStatus",
        "kms:GetKeyPolicy",
        "kms:DescribeKey",
        "kms:ListKeyPolicies",
        "kms:ListResourceTags",
        "tag:GetResources"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "ReadOnlyAccessForOperationsWithNoKMSKey",
      "Effect": "Allow",
      "Action": [
        "kms:ListKeys",
        "kms:ListAliases",
        "iam:ListRoles",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Einem Benutzer das Erstellen von KMS-Schlüsseln erlauben

Die folgende IAM-Richtlinie erlaubt Benutzern die Erstellung aller Arten von KMS-Schlüsseln. Der Wert des `Resource` Elements ist darauf `*` zurückzuführen, dass der `CreateKey` Vorgang keine bestimmten AWS KMS Ressourcen (KMS-Schlüssel oder Aliase) verwendet.

Um den Benutzer auf bestimmte Typen von KMS-Schlüsseln zu beschränken, verwenden Sie die KeyOrigin Bedingungsschlüssel kms: KeyUsage, kms: und kms:. KeySpec

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "kms:CreateKey",
    "Resource": "*"
  }
}
```

Prinzipale, die Schlüssel erstellen, benötigen möglicherweise einige verwandte Berechtigungen.

- kms: PutKeyPolicy — Prinzipale, die über die entsprechenden `kms:CreateKey` Rechte verfügen, können die Richtlinie für den ersten Schlüssel für den KMS-Schlüssel festlegen. Der `CreateKey` Aufrufer muss jedoch über die PutKeyPolicykms-Berechtigung verfügen, mit der er die KMS-Schlüsselrichtlinie ändern kann, oder er muss den `BypassPolicyLockoutSafetyCheck` Parameter von angeben`CreateKey`, was nicht empfohlen wird. Der `CreateKey`-Anrufer kann die `kms:PutKeyPolicy`-Berechtigung für den KMS-Schlüssel aus einer IAM-Richtlinie erhalten, oder er kann diese Berechtigung in die Schlüsselrichtlinie des KMS-Schlüssels aufnehmen, den er erstellt.
- kms: TagResource — Um dem KMS-Schlüssel während des `CreateKey` Vorgangs Tags hinzuzufügen, muss der `CreateKey` Aufrufer in einer IAM-Richtlinie über die kms: TagResource -Berechtigung verfügen. Die Aufnahme dieser Berechtigung in die Schlüsselrichtlinie des neuen KMS-Schlüssels ist nicht ausreichend. Wenn der `CreateKey`-Anrufer jedoch `kms:TagResource` in die ursprünglichen Schlüsselrichtlinie aufnimmt, kann er Tags in einem separaten Aufruf hinzufügen, nachdem der KMS-Schlüssel erstellt wurde.
- kms: CreateAlias — Principals, die einen KMS-Schlüssel in der AWS KMS Konsole erstellen, müssen über die CreateAlias kms-Berechtigung für den KMS-Schlüssel und den Alias verfügen. (Die Konsole führt zwei Aufrufe aus; einen an `CreateKey` und einen an `CreateAlias`). Sie müssen die Aliasberechtigung in einer IAM-Richtlinie angeben. Sie können dem KMS-Schlüssel die Berechtigungen in einer Schlüsselrichtlinie oder einer IAM-Richtlinie bereitstellen. Details hierzu finden Sie unter Steuern des Zugriffs auf Aliasse.

Darüber hinaus gewährt die folgende IAM-Richtlinie `kms:TagResource` Berechtigungen für alle KMS-Schlüssel im Konto AWS-Konto und `kms:CreateAlias` Berechtigungen für alle Aliase des

Kontos. `kms:CreateKey` Sie enthält auch einige nützliche schreibgeschützte Berechtigungen, die nur in einer IAM-Richtlinie bereitgestellt werden können.

Diese IAM-Richtlinie enthält nicht `kms:PutKeyPolicy` oder andere Berechtigungen, die in einer Schlüsselrichtlinie festgelegt werden können. Dabei handelt es sich um eine bewährte Methode, diese Berechtigungen in der Schlüsselrichtlinie festzulegen, in der sie ausschließlich für einen KMS-Schlüssel gelten.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPermissionsForParticularKMSKeys",
      "Effect": "Allow",
      "Action": "kms:TagResource",
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMPermissionsForParticularAliases",
      "Effect": "Allow",
      "Action": "kms:CreateAlias",
      "Resource": "arn:aws:kms:*:111122223333:alias/*"
    },
    {
      "Sid": "IAMPermissionsForAllKMSKeys",
      "Effect": "Allow",
      "Action": [
        "kms:CreateKey",
        "kms:ListKeys",
        "kms:ListAliases"
      ],
      "Resource": "*"
    }
  ]
}
```

Ermöglicht einem Benutzer das Verschlüsseln und Entschlüsseln mit einem beliebigen KMS-Schlüssel in einem bestimmten AWS-Konto

Die folgende IAM-Richtlinie ermöglicht es einem Benutzer, Daten mit einem beliebigen KMS-Schlüssel in 111122223333 zu verschlüsseln und zu entschlüsseln. AWS-Konto

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:*:111122223333:key/*"
  }
}
```

Erlaubt einem Benutzer das Verschlüsseln und Entschlüsseln mit einem beliebigen KMS-Schlüssel in einer bestimmten Region AWS-Konto

Die folgende IAM-Richtlinie ermöglicht es einem Benutzer, Daten mit einem beliebigen KMS-Schlüssel AWS-Konto 111122223333 in der Region USA West (Oregon) zu verschlüsseln und zu entschlüsseln.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": [
      "arn:aws:kms:us-west-2:111122223333:key/*"
```

```
    ]
  }
}
```

Einem Benutzer das Verschlüsseln und Entschlüsseln mit bestimmten KMS-Schlüssel erlauben

Die folgende IAM-Richtlinie erlaubt es einem Benutzer, mit den beiden im `Resource`-Element angegebenen KMS-Schlüssel Daten zu verschlüsseln und zu entschlüsseln. Um einen KMS-Schlüssel in einer IAM-Richtlinienanweisung anzugeben, müssen Sie dessen Schüssel-ARN verwenden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": [
      "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "arn:aws:kms:us-west-2:111122223333:key/01234abc-d12a-b34c-d56e-
f1234567890a'"
    ]
  }
}
```

Einen Benutzer am Deaktivieren oder Löschen von KMS-Schlüsseln hindern

Die folgende IAM-Richtlinie verhindert die Deaktivierung oder Löschung von KMS-Schlüsseln durch Benutzer, auch wenn eine andere IAM-Richtlinie oder eine Schlüsselrichtlinie diese Berechtigungen gewährt. Eine Richtlinie, die Berechtigungen explizit verweigert, hat Vorrang vor allen anderen Richtlinien, auch wenn diese dieselben Berechtigungen explizit erteilen. Weitere Informationen finden Sie unter Problembehandlung bei Berechtigungen AWS KMS.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [
      "kms:DisableKey",
      "kms:ScheduleKeyDeletion"
    ],
    "Resource": "*"
  }
}
```

Richtlinien zur Ressourcenkontrolle in AWS KMS

Ressourcenkontrollrichtlinien (RCPs) sind eine Art von Unternehmensrichtlinie, mit der Sie präventive Kontrollen für AWS Ressourcen in Ihrer Organisation durchsetzen können. RCPs helfen Ihnen dabei, den externen Zugriff auf Ihre AWS Ressourcen zentral und in großem Umfang einzuschränken. RCPs ergänzen die Richtlinien zur Servicekontrolle (SCPs). SCPs Kann zwar verwendet werden, um die maximalen Berechtigungen für die IAM-Rollen und -Benutzer in Ihrer Organisation zentral festzulegen, RCPs kann aber auch verwendet werden, um die maximalen Berechtigungen für AWS Ressourcen in Ihrer Organisation zentral festzulegen.

Sie können es verwenden RCPs , um Berechtigungen für die vom Kunden verwalteten KMS-Schlüssel in Ihrer Organisation zu verwalten. RCPs allein reichen nicht aus, um Ihren vom Kunden verwalteten Schlüsseln Berechtigungen zu erteilen. Ein RCP gewährt keine Berechtigungen. Ein RCP definiert eine Berechtigungsschranke oder legt Grenzwerte für die Aktionen fest, die Identitäten mit Ressourcen in den betroffenen Konten ausführen können. Der Administrator muss IAM-Rollen oder -Benutzern weiterhin identitätsbasierte Richtlinien oder wichtige Richtlinien zuordnen, um tatsächlich Berechtigungen zu erteilen.

> **Note**
>
> Die Richtlinien zur Ressourcenkontrolle in Ihrer Organisation gelten nicht für. Von AWS verwaltete Schlüssel

Von AWS verwaltete Schlüssel werden in Ihrem Namen von einem AWS Dienst erstellt, verwaltet und verwendet, dessen Berechtigungen Sie nicht ändern oder verwalten können.

Weitere Informationen

- Weitere allgemeine Informationen dazu RCPs finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle.
- Einzelheiten zur Definition RCPs, einschließlich Beispiele, finden Sie unter RCP-Syntax im AWS Organizations Benutzerhandbuch.

Das folgende Beispiel zeigt, wie Sie mithilfe eines RCP verhindern können, dass externe Principals auf vom Kunden verwaltete Schlüssel in Ihrer Organisation zugreifen. Diese Richtlinie ist nur ein Beispiel, und Sie sollten sie an Ihre individuellen Geschäfts- und Sicherheitsanforderungen anpassen. Möglicherweise möchten Sie Ihre Richtlinie so anpassen, dass Ihre Geschäftspartner darauf zugreifen können. Weitere Informationen finden Sie im Repository mit Beispielen für Datenperimeter-Richtlinien.

> **Note**
>
> Die `kms:RetireGrant` Berechtigung ist in einem RCP nicht wirksam, auch wenn das `Action` Element ein Sternchen (*) als Platzhalter angibt.
> Weitere Hinweise dazu, wie die Berechtigung für bestimmt `kms:RetireGrant` wird, finden Sie unter. Außerbetriebnahme und Widerruf von Erteilungen

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RCPEnforceIdentityPerimeter",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "kms:*",
            "Resource": "*",
            "Condition": {
```

```
                "StringNotEqualsIfExists": {
                    "aws:PrincipalOrgID": "my-org-id"
                },
                "Bool": {
                    "aws:PrincipalIsAWSService": "false"
                }
            }
        }
    ]
}
```

Das folgende RCP-Beispiel erfordert, dass AWS Service Principals nur dann auf Ihre vom Kunden verwalteten KMS-Schlüssel zugreifen können, wenn die Anfrage von Ihrer Organisation stammt. Diese Richtlinie wendet die Kontrolle nur auf Anfragen an, die vorhanden sind. `aws:SourceAccount` Dadurch wird sichergestellt, dass Serviceintegrationen, für die keine Verwendung von erforderlich ist, `aws:SourceAccount` nicht beeinträchtigt werden. Wenn im Anforderungskontext vorhanden `aws:SourceAccount` ist, wird die `Null` Bedingung als 0 ausgewertet`true`, wodurch der `aws:SourceOrgID` Schlüssel erzwungen wird.

Weitere Informationen zum Problem mit dem verwirrten Stellvertreter finden Sie unter Das Problem mit dem verwirrten Stellvertreter im IAM-Benutzerhandbuch.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RCPEnforceConfusedDeputyProtection",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "kms:*",
            "Resource": "*",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "aws:SourceOrgID": "my-org-id"
                },
                "Bool": {
                    "aws:PrincipalIsAWSService": "true"
```

```
                },
                "Null": {
                    "aws:SourceAccount": "false"
                }
            }
        }
    ]
}
```

Zuschüsse in AWS KMS

Eine Erteilung ist ein Richtlinieninstrument, das es AWS -Prinzipalen erlaubt, KMS-Schlüssel in kryptografischen Operationen zu verwenden. Es kann ihnen auch erlauben, einen KMS-Schlüssel anzuzeigen (`DescribeKey`) und Erteilungen zu erstellen und zu verwalten. Bei der Autorisierung des Zugriffs auf einen KMS-Schlüssel werden Erteilungen zusammen mit Schlüsselrichtlinien und IAM-Richtlinien berücksichtigt. Erteilungen werden häufig für temporäre Berechtigungen verwendet, da Sie eine erstellen, deren Berechtigungen verwenden und sie dann wieder löschen können, ohne Ihre Schlüsselrichtlinien oder IAM-Richtlinien zu ändern.

Grants werden häufig von AWS Diensten verwendet, die sich integrieren, AWS KMS um Ihre Daten im Ruhezustand zu verschlüsseln. Der Service erstellt eine Erteilung im Namen eines Benutzers im Konto, verwendet seine Berechtigungen und hebt die Erteilung auf, sobald die Aufgabe abgeschlossen ist. Einzelheiten darüber, wie AWS Dienste Zuschüsse verwenden, finden Sie unter dem Thema Verschlüsselung im Ruhezustand im Benutzer- oder Entwicklerhandbuch des jeweiligen Dienstes.

Erteilungen sind ein sehr flexibler und nützlicher Zugriffsteuerungs-Mechanismus. Wenn Sie eine Erteilung für einen KMS-Schlüssel erstellen, erlaubt die Erteilung es dem Empfänger-Prinzipal, die angegebenen Erteilungs-Operationen für den KMS-Schlüssel aufzurufen, vorausgesetzt, dass alle in der Erteilung angegebenen Bedingungen erfüllt sind.

- Jede Erteilung erlaubt den Zugriff auf genau einen KMS-Schlüssel. Sie können eine Erteilung für einen KMS-Schlüssel in einem anderen AWS-Konto erstellen.
- Eine Erteilung kann den Zugriff auf einen KMS-Schlüssel erlauben, aber nicht den Zugriff verweigern.
- Jede Erteilung hat einen Empfänger-Prinzipal. Der Principal des Empfängers kann eine oder mehrere Identitäten im selben AWS-Konto KMS-Schlüssel oder in einem anderen Konto repräsentieren.

- Eine Erteilung kann nur Erteilungs-Operationen erlauben. Die Erteilungs-Operationen müssen durch den KMS-Schlüssel in der Erteilung unterstützt werden. Wenn Sie einen nicht unterstützten Vorgang angeben, schlägt die CreateGrantAnforderung mit einer Ausnahme fehl. `ValidationError`
- Der Empfänger-Prinzipal können die Berechtigungen verwenden, die ihnen durch die Erteilung gewährt werden, ohne die Erteilung anzugeben, genauso wie wenn die Berechtigungen aus einer Schlüsselrichtlinie oder IAM-Richtlinie stammen. Da die AWS KMS API jedoch einem Konsistenzmodell folgt, kann es bei der Erstellung, Einstellung oder Widerruf eines Zuschusses zu einer kurzen Verzögerung kommen, bis die Änderung vollständig verfügbar ist. AWS KMS Um die Berechtigungen in einer Erteilung sofort zu verwenden, verwenden Sie einen Erteilungs-Token.
- Ein autorisierter Prinzipal kann die Erteilung löschen, (aufheben oder widerrufen). Durch das Löschen einer Erteilung entfallen alle Berechtigungen, die durch die Erteilung erlaubt wurden. Sie müssen nicht herausfinden, welche Richtlinien hinzugefügt oder entfernt werden sollen, um die Erteilung rückgängig zu machen.
- AWS KMS begrenzt die Anzahl der Grants für jeden KMS-Schlüssel. Details hierzu finden Sie unter Erteilungen pro KMS-Schlüssel: 50 000.

Seien Sie vorsichtig, wenn Sie Erteilungen erstellen und anderen die Berechtigung zum Erstellen von Erteilungen erteilen. Die Erlaubnis zur Erstellung von Zuschüssen hat Auswirkungen auf die Sicherheit, ähnlich wie die PutKeyPolicy Erlaubnis, Richtlinien festzulegen, mit dem kms:.

- Benutzer mit der Berechtigung, Berechtigungen für einen KMS-Schlüssel (`kms:CreateGrant`) zu erstellen, können eine Erteilung verwenden, um Benutzern und Rollen, einschließlich AWS Diensten, die Verwendung des KMS-Schlüssels zu ermöglichen. Bei den Prinzipalen kann es sich um eigene Identitäten AWS-Konto oder um Identitäten in einem anderen Konto oder einer anderen Organisation handeln.
- Zuschüsse können nur eine Teilmenge von Vorgängen ermöglichen. AWS KMS Sie können Erteilungen verwenden, um es Prinzipalen zu erlauben, den KMS-Schlüssel anzuzeigen, ihn in kryptografischen Operationen zu verwenden und Erteilungen zu erstellen und außer Betrieb zu nehmen. Details dazu finden Sie unter Erteilungs-Operationen. Sie können auch Erteilungs-Einschränkungen verwenden, um die Berechtigungen in einer Erteilung für eine Schlüssel mit symmetrischer Verschlüsselung einzuschränken.
- Prinzipale können Berechtigung zum Erstellen von Erteilungen aus einer Schlüsselrichtlinie oder IAM-Richtlinie erhalten. Prinzipale, die `kms:CreateGrant`-Berechtigung aus einer Richtlinie bekommen, können Zuschüsse für alle Erteilungs-Operationen auf dem KMS-Schlüssel erstellen.

Diese Prinzipale müssen nicht über die Berechtigung verfügen, die sie für den Schlüssel erteilen. Wenn Sie die `kms:CreateGrant`-Berechtigung in einer Richtlinie erlauben, können Sie Richtlinienbedingungen verwenden, um diese Berechtigung einzuschränken.

- Prinzipale können auch die Berechtigung erhalten, Erteilungen aus einer Erteilung zu erstellen. Diese Prinzipale können nur die Berechtigungen delegieren, die ihnen erteilt wurden, auch wenn sie über andere Berechtigungen aus einer Richtlinie verfügen. Details hierzu finden Sie unter Genehmigung erteilen CreateGrant .

Konzepte für Erteilungen

Um Erteilungen effektiv nutzen zu können, müssen Sie die Begriffe und Konzepte verstehen, die AWS KMS verwendet.

Einschränkungen für Erteilungen

Eine Bedingung, die die Berechtigungen in der Erteilung einschränkt. AWS KMS Unterstützt derzeit Gewährungsbeschränkungen, die auf dem Verschlüsselungskontext in der Anforderung für einen kryptografischen Vorgang basieren. Details hierzu finden Sie unter Verwenden von Erteilungs-Einschränkungen.

Erteilungs-ID

Die eindeutige ID einer Erteilung für einen KMS-Schlüssel. Sie können eine Grant-ID zusammen mit einer Schlüssel-ID verwenden, um einen Zuschuss in einer RetireGrantRevokeGrantOR-Anfrage zu identifizieren.

Erteilungs-Operationen

Die AWS KMS Vorgänge, die Sie in einem Zuschuss zulassen können. Wenn Sie andere Operationen angeben, schlägt die CreateGrantAnfrage mit einer `ValidationError` Ausnahme fehl. Dies sind auch die Operationen, die einen Erteilungs-Token akzeptieren. Ausführliche Informationen über diese Berechtigungen finden Sie unter AWS KMS Berechtigungen.

Diese Erteilungs-Operationen stellen tatsächlich die Berechtigung zur Verwendung der Produktion dar. Daher gilt, dass Sie für die `ReEncrypt`-Produktion `ReEncryptFrom`, `ReEncryptTo`, oder beide `ReEncrypt*` angeben können.

Die Erteilungs-Operationen sind:

- Kryptografische Operationen
  - Decrypt

  - DeriveSharedSecret
  - Encrypt
  - GenerateDataKey
  - GenerateDataKeyPair
  - GenerateDataKeyPairWithoutPlaintext
  - GenerateDataKeyWithoutPlaintext
  - GenerateMac
  - ReEncryptFrom
  - ReEncryptTo
  - Sign
  - Verify
  - VerifyMac
- Andere Produktionen
  - CreateGrant
  - DescribeKey
  - GetPublicKey
  - RetireGrant

Die Erteilungs-Operationen, die Sie erlauben, müssen durch den KMS-Schlüssel in der Erteilung unterstützt werden. Wenn Sie einen nicht unterstützten Vorgang angeben, schlägt die CreateGrantAnforderung mit einer `ValidationError` Ausnahme fehl. Beispielsweise können Erteilungen für KMS-Schlüssel mit symmetrischer Verschlüsselung die Operationen Signieren, Überprüfen, `GenerateMac` oder `VerifyMac` nicht zulassen. Erteilungen für asymmetrische KMS-Schlüssel dürfen keine Operationen zulassen, die Datenschlüssel oder Datenschlüsselpaare generieren.

Erteilungs-Token

Die AWS KMS API folgt einem eventuellen Konsistenzmodell. Wenn Sie eine Erteilung erstellen, kann es zu einer kurzen Verzögerung kommen, bis die Änderung in AWS KMS verfügbar ist. In der Regel dauert es weniger als ein paar Sekunden, bis sich die Änderung im gesamten System verbreitet, in einigen Fällen kann es jedoch mehrere Minuten dauern. Wenn Sie versuchen, eine Erteilung zu verwenden, bevor sie sich vollständig im System verbreitet hat, erhalten Sie möglicherweise eine Zugriffsverweigerungsmeldung. Mit einem Erteilungs-Token können Sie auf die Erteilung verweisen und die Erteilungs-Berechtigungen sofort verwenden.

Ein Berechtigungserteilungs-Token ist eine eindeutige, nicht-geheime, base64-kodierte Zeichenfolge mit variabler Länge, die eine Erteilung darstellt. Sie können den Erteilungs-Token verwenden, um die Erteilung in jeder Erteilungs-Produktion zu identifizieren. Da der Token-Wert jedoch ein Hash-Digest ist, zeigt er keine Details über die Erteilung an.

Ein Erteilungs-Token darf nur so lange verwendet werden, bis sich die Erteilung vollständig in AWS KMS ausgebreitet hat. Danach kann der Empfänger-Prinzipal die Berechtigung in der Erteilung ohne Angabe eines Grant-Token oder eines anderen Beweises für die Erteilung verwenden. Sie können jederzeit ein Grant-Token verwenden, aber sobald der Grant letztendlich konsistent ist, AWS KMS verwendet es den Grant, um die Berechtigungen zu bestimmen, nicht das Grant-Token.

Mit dem folgenden Befehl wird die GenerateDataKeyOperation beispielsweise aufgerufen. Er verwendet ein Erteilung-Token, um die Erteilung darzustellen, die dem Aufrufer (dem erteilenden Prinzipal) die Berechtigung zum Aufrufen von `GenerateDataKey` für den angegebenen KMS-Schlüssel erteilt.

```
$ aws kms generate-data-key \
        --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
        --key-spec AES_256 \
        --grant-token $token
```

Sie können den Erteilungs-Token verwenden, um eine Erteilung in Operationen zu identifizieren, die Erteilungen verwalten. Beispielsweise kann der ausscheidende Schulleiter bei einem Aufruf der RetireGrantOperation ein Grant-Token verwenden.

```
$ aws kms retire-grant \
        --grant-token $token
```

`CreateGrant` ist die einzige Produktion, die ein Erteilungs-Token zurückgibt. Sie können kein Grant-Token aus einem anderen AWS KMS Vorgang oder aus dem CloudTrail Protokollereignis für den CreateGrant Vorgang abrufen. Die ListRetirableGrantsOperationen ListGrantsund geben die Grant-ID zurück, aber kein Grant-Token.

Details hierzu finden Sie unter Verwenden eines Erteilungs-Token.

Erteilender Prinzipal

Die Identitäten, die die in der Erteilung angegebenen Berechtigungen erhalten. Jede Erteilung hat einen Empfänger-Prinzipal, aber der Empfänger-Prinzipal kann mehrere Identitäten repräsentieren.

Der Prinzipal des Empfängers kann ein beliebiger AWS Prinzipal sein, einschließlich eines AWS-Konto (Root-), eines IAM-Benutzers, einer IAM-Rolle, einer Verbundrolle oder eines Verbundbenutzers oder eines Benutzers mit angenommener Rolle. Der Empfänger-Prinzipal kann sich im selben Konto wie der KMS-Schlüssel oder in einem anderen Konto befinden. Der Empfänger-Prinzipal kann jedoch kein Service-Prinzipal und keine IAM-Gruppe oder AWS - Organisation sein.

> **Note**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Außerbetriebnahme (einer Erteilung)

Beendet eine Erteilung. Sie können eine Erteilung aufheben, wenn Sie die Berechtigungen nicht mehr brauchen.

Wenn Sie eine Erteilung widerrufen oder außer Betrieb nehmen, wird die Erteilung gelöscht. Die Außerbetriebnahme erfolgt jedoch durch einen Prinzipal, der in der Erteilung angegeben ist. Das Widerrufen erfolgt in der Regel durch einen Schlüsseladministrator. Details hierzu finden Sie unter Außerbetriebnahme und Widerruf von Erteilungen.

Außerbetriebnahme eines Prinzipals

Ein Prinzipal, der eine Erteilung aufheben kann. Sie können einen ausscheidenden Prinzipal in einer Erteilung angeben, jedoch ist es nicht erforderlich. Bei dem ausscheidenden Prinzipal kann es sich um einen beliebigen AWS Prinzipal handeln, einschließlich IAM-Benutzer AWS-Konten, IAM-Rollen, Verbundbenutzer und Benutzer mit angenommener Rolle. Der ausscheidende Prinzipal kann sich im selben Konto wie der KMS-Schlüssel oder in einem anderen Konto befinden.

> **Note**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Neben dem im Zuschuss angegebenen ausscheidenden Hauptbenutzer kann ein Grant auch von demjenigen zurückgezogen werden, AWS-Konto in dem der Grant eingerichtet wurde. Wenn die Erteilung die `RetireGrant`-Produktion erlaubt, kann der Empfänger-Prinzipal die Erteilung aufheben. Außerdem kann der AWS-Konto oder ein Schulleiter AWS-Konto , der in den Ruhestand geht, die Genehmigung zur Einstellung eines Zuschusses an einen IAM-Schulleiter delegieren. AWS-Konto Details hierzu finden Sie unter Außerbetriebnahme und Widerruf von Erteilungen.

Widerrufen (einer Erteilung)

Beendet eine Erteilung. Sie widerrufen eine Erteilung, um aktiv die Berechtigung abzulehnen, die die Erteilung erlaubt.

Wenn Sie eine Erteilung widerrufen oder außer Betrieb nehmen, wird die Erteilung gelöscht. Die Außerbetriebnahme erfolgt jedoch durch einen Prinzipal, der in der Erteilung angegeben ist. Das Widerrufen erfolgt in der Regel durch einen Schlüsseladministrator. Details hierzu finden Sie unter Außerbetriebnahme und Widerruf von Erteilungen.

Letztendliche Konsistenz (für Erteilungen)

Die AWS KMS API folgt einem späteren Konsistenzmodell. Wenn Sie eine Erteilung erstellen, aufheben oder widerrufen, kann es zu einer kurzen Verzögerung kommen, bevor die Änderung in allen Bereichen von AWS KMS verfügbar ist. In der Regel dauert es weniger als ein paar Sekunden, bis sich die Änderung im gesamten System verbreitet, in einigen Fällen kann es jedoch mehrere Minuten dauern.

Diese kurze Verzögerung kann Ihnen auffallen, wenn Sie unerwartete Fehler erhalten. Wenn Sie beispielsweise versuchen, einen neuen Zuschuss zu verwalten oder die Berechtigungen in einem neuen Zuschuss zu verwenden, bevor der Zuschuss allgemein bekannt ist AWS KMS, erhalten Sie möglicherweise die Fehlermeldung Zugriff verweigert. Wenn Sie eine Erteilung aufheben oder widerrufen, kann der Empfänger-Prinzipal seine Berechtigungen möglicherweise für einen kurzen

Zeitraum verwenden, bis die Erteilung vollständig gelöscht wurde. Die typische Strategie besteht darin, die Anfrage erneut zu versuchen, und einige AWS SDKs beinhalten automatische Backoff- und Wiederholungslogik.

AWS KMS verfügt über Funktionen, um diese kurze Verzögerung zu minimieren.

- Um die Berechtigungen in einer neuen Erteilung sofort zu verwenden, verwenden Sie einen Erteilungs-Token. Sie können einen Erteilungs-Token verwenden, um die Erteilung in jeder Erteilungs-Produktion zu identifizieren. Detaillierte Anweisungen finden Sie unter Verwenden eines Erteilungs-Token.
- Der CreateGrantVorgang verfügt über einen `Name` Parameter, der verhindert, dass bei Wiederholungsvorgängen doppelte Zuschüsse erstellt werden.

> **Note**
>
> Erteilungs-Token ersetzen die Gültigkeit der Erteilung, bis alle Endpunkte im Service mit dem neuen Erteilungsstatus aktualisiert wurden. In den meisten Fällen wird die letztendliche Konsistenz innerhalb von fünf Minuten erreicht.

Weitere Informationen finden Sie unter AWS KMS Letztendliche Konsistenz.

Bewährte Verfahren für Zuschüsse AWS KMS

AWS KMS empfiehlt die folgenden bewährten Methoden für die Erstellung, Verwendung und Verwaltung von Zuschüssen.

- Beschränken Sie die Berechtigungen in der Erteilung auf diejenigen, die der Empfänger-Prinzipal benötigt. Erteilen Sie Zugriff nach dem Prinzip der geringsten Berechtigung.
- Verwenden Sie einen bestimmten Empfänger-Prinzipal, z. B. eine IAM-Rolle, und erteilen Sie dem Empfänger-Prinzipal nur die Berechtigung, die API-Operationen zu verwenden, die er benötigt.
- Verwenden Sie den Verschlüsselungskontext Erteilungs-Einschränkungen, um sicherzustellen, dass Anrufer den KMS-Schlüssel für den beabsichtigten Zweck verwenden. Einzelheiten zur Verwendung des Verschlüsselungskontextes in einer Anfrage zur Sicherung Ihrer Daten finden Sie unter So schützen Sie die Integrität Ihrer verschlüsselten Daten durch Verwendung AWS Key Management Service und EncryptionContext im AWS Sicherheitsblog.

> **Tip**
>
> Verwenden Sie wann immer möglich die EncryptionContextEqualZuschussbeschränkung. Die korrekte Verwendung der EncryptionContextSubsetZuschussbeschränkung ist schwieriger. Wenn Sie sie verwenden müssen, lesen Sie die Dokumentation sorgfältig durch und testen Sie die Erteilungs-Einschränkung, um sicherzustellen, dass sie wie beabsichtigt funktioniert.

- Löschen Sie doppelte Erteilungen. Doppelte Erteilungen haben dieselben API-Aktionen und denselben Schlüssel-ARN, Empfänger-Prinzipal, Verschlüsselungskontext und Namen. Wenn Sie die ursprüngliche Erteilung aufheben oder widerrufen, aber die doppelte Erteilungen belassen, stellen die verbleibenden doppelten Erteilungen unbeabsichtigte Eskalationen von Rechten dar. Um bei `CreateGrant`-Anforderungen zu vermeiden, dass doppelte Erteilungen erstellt werden, verwenden Sie den `Name`-Parameter. Verwenden Sie die ListGrantsOperation, um doppelte Zuschüsse zu erkennen. Wenn Sie versehentlich eine doppelte Erteilung erstellen, widerrufen Sie bzw. nehmen Sie diese sie so schnell wie möglich außer Betrieb.

> **Note**
>
> Erteilungen für AWS -verwaltete Schlüssel könnten wie Duplikate aussehen, haben aber unterschiedliche Empfänger-Prinzipale.
> Das `GranteePrincipal`-Feld in der `ListGrants`-Antwort enthält normalerweise den Berechtigungsprinzipal der Genehmigung. Wenn es sich bei der Hauptperson des Zuschusses jedoch um eine AWS Dienstleistung handelt, enthält das `GranteePrincipal` Feld den Dienstprinzipal, der für mehrere verschiedene Prinzipale des Zuschusses stehen kann.

- Denken Sie daran, dass Erteilungen nicht automatisch ablaufen. Außerbetriebnahme oder Widerruf der Erteilung sobald die Berechtigung nicht mehr benötigt wird. Erteilungen, die nicht gelöscht werden, können ein Sicherheitsrisiko für verschlüsselte Ressourcen verursachen.

Steuerung des Zugriffs auf Erteilungen

Sie können den Zugriff auf Operationen steuern, die Erteilungen in Schlüsselrichtlinien, IAM-Richtlinien und anderen Erteilungen erstellen und verwalten. Prinzipale, die die `CreateGrant`-Berechtigung von einer Erteilung erhalten, haben mehr eingeschränkte Erteilungs-Berechtigungen.

API-Produktion	Schlüsselrichtlinie oder IAM-Richtlinie	Gewährung
CreateGrant	✓	✓
ListGrants	✓	-
ListRetirableGrants	✓	-
Außerbetriebnahme einer Erteilung	(Begrenzt. Siehe Außerbetriebnahme und Widerruf von Erteilungen)	✓
RevokeGrant	✓	-

Wenn Sie eine Schlüsselrichtlinie oder IAM-Richtlinie verwenden, um den Zugriff auf Vorgänge zu steuern, die Zuschüsse erstellen und verwalten, können Sie eine oder mehrere der folgenden Richtlinienbedingungen verwenden, um die Berechtigung einzuschränken. AWS KMS unterstützt alle der folgenden Bedingungsschlüssel im Zusammenhang mit Zuschüssen. Ausführliche Informationen und Beispiele finden Sie unter AWS KMS Bedingungsschlüssel.

km: GrantConstraintType

Erlaubt es Prinzipalen, eine Erteilung nur zu erstellen, wenn die Erteilung die angegebene Erteilungs-Einschränkung enthält.

km: GrantIsFor AWSResource

Ermöglicht es Principals`CreateGrant`, anzurufen`ListGrants`, oder `RevokeGrant` nur, wenn ein AWS Service, der in integriert ist, die Anfrage im Namen des Prinzipals AWS KMS sendet.

km: GrantOperations

Erlaubt es Prinzipalen, eine Erteilung zu erstellen, beschränkt jedoch die Erteilung auf die angegebenen Operationen.

km: GranteePrincipal

Erlaubt es Prinzipalen, eine Erteilung nur für den angegebenen Empfänger-Prinzipal zu erstellen.

km: RetiringPrincipal

Erlaubt es Prinzipalen, eine Erteilung nur zu erstellen, wenn die Erteilung einen bestimmten ausscheidenden Prinzipal angibt.

Erstellen einer Erteilung

Informieren Sie sich vor dem Erstellen einer Erteilung über die Optionen für benutzerdefinierte Erteilungen. Sie können auch Erteilungs-Einschränkungen verwenden, um die Berechtigungen in einer Erteilung einzuschränken. Erfahren Sie auch mehr über die Erteilung der `CreateGrant`-Berechtigung. Prinzipale, die die Berechtigung zum Erstellen von Erteilungen aus einer Erteilung erhalten, sind in den Erteilungen beschränkt, die sie erstellen können.

Themen

- Erstellen einer Erteilung
- Genehmigung erteilen CreateGrant

Erstellen einer Erteilung

Rufen Sie die CreateGrantOperation auf, um einen Grant zu erstellen. Geben Sie einen KMS-Schlüssel an, einen Empfänger-Prinzipal und eine Liste der zulässigen Erteilungs-Operationen. Sie können auch einen optionalen ausscheidenden Prinzipal angeben. Um die Erteilung anzupassen, verwenden Sie optionale `Constraints`-Parameter, um Erteilungseinschränkungen zu definieren.

Wenn Sie eine Erteilung erstellen, aufheben oder widerrufen, kann es zu einer kurzen Verzögerung (in der Regel weniger als fünf Minuten) kommen, bevor die Änderung in allen Bereichen von AWS KMS verfügbar ist. Weitere Informationen finden Sie unter Letztendliche Konsistenz (für Erteilungen).

Der folgende `CreateGrant`-Befehl erstellt beispielsweise eine Erteilung, die Benutzer, die die `keyUserRole`-Rolle annehmen dürfen, den Aufruf der Decrypt-Operation für den angegebenen symmetrischen KMS-Schlüssel erlaubt. Die Erteilung legt mit dem Parameter `RetiringPrincipal` einen Prinzipal fest, der die erteilte Berechtigung aufheben kann. Es enthält auch eine Erteilungseinschränkung, die die Berechtigung nur zulässt, wenn der Verschlüsselungskontext in der Anforderung `"Department": "IT"` einschließt.

```
$  aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
```

```
    --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \
    --operations Decrypt \
    --retiring-principal arn:aws:iam::111122223333:role/adminRole \
    --constraints EncryptionContextSubset={Department=IT}
```

Wenn Ihr Code die `CreateGrant`-Produktion erneut versucht, oder ein AWS -SDK verwendet, das Anforderungen automatisch wiederholt, verwenden Sie den optionalen Name-Parameter, um das Erstellen von doppelten Erteilungen zu verhindern. Wenn AWS KMS eine `CreateGrant` Anfrage für einen Zuschuss mit den gleichen Eigenschaften wie ein vorhandener Zuschuss eingeht, einschließlich des Namens, wird die Anfrage als Wiederholungsversuch erkannt und kein neuer Zuschuss erstellt. Sie können nicht den `Name`-Wert verwenden, um die Erteilung in einer AWS KMS -Produktion zu identifizieren.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen im Namen der Erteilung an. Es kann als Klartext in CloudTrail Protokollen und anderen Ausgaben erscheinen.

```
$ aws kms create-grant \
    --name IT-1234abcd-keyUserRole-decrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \
    --operations Decrypt \
    --retiring-principal arn:aws:iam::111122223333:role/adminRole \
    --constraints EncryptionContextSubset={Department=IT}
```

Codebeispiele, die zeigen, wie Zuschüsse in verschiedenen Programmiersprachen erstellt werden, finden Sie unterVerwendung `CreateGrant` mit einem AWS SDK oder CLI.

Verwenden von Erteilungs-Einschränkungen

Erteilungs-Einschränkungen legen Bedingungen für die Berechtigungen fest, die der Empfänger-Prinzipal ausführen kann. Erteilungs-Einschränkungen treten an die Stelle von Bedingungsschlüssel in einer Schlüsselrichtlinie oder IAM-Richtlinie. Jeder Erteilungs-Einschränkungs-Wert kann bis zu 8 Verschlüsselungskontext-Paare enthalten. Der Verschlüsselungskontext-Wert in jeder Erteilungs-Einschränkung darf 384 Zeichen nicht überschreiten.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

AWS KMS unterstützt zwei Grant-Beschränkungen `EncryptionContextEquals` und`EncryptionContextSubset`, die beide Anforderungen an den Verschlüsselungskontext bei einer Anforderung für einen kryptografischen Vorgang festlegen.

Die Verschlüsselungskontext-Erteilungs-Einschränkungen sind für die Verwendung mit Erteilungs-Operationen entworfen, die einen Verschlüsselungskontext-Parameter haben.

- Beschränkungen für den Verschlüsselungskontext gelten nur in einer Erteilung für einen KMS-Schlüssel mit symmetrischer Verschlüsselung. Kryptografische Operationen mit asymmetrischen KMS-Schlüsseln unterstützen keinen Verschlüsselungskontext.
- Die Verschlüsselungskontext-Einschränkung wird für `DescribeKey`- und `RetireGrant`-Operationen ignoriert. `DescribeKey` und `RetireGrant` haben keinen Verschlüsselungskontext-Parameter, aber Sie können diese Operationen in eine Erteilung einschließen, die über eine Verschlüsselungskontext-Einschränkung verfügt.
- Sie können eine Verschlüsselungskontext-Einschränkung in einer Erteilung für die `CreateGrant`-Produktion verwenden. Die Verschlüsselungskontext-Einschränkung erfordert, dass alle Erteilungen, die mit der `CreateGrant`-Berechtigung erstellt wurden, eine ebenso strenge oder strengere Verschlüsselungskontext-Einschränkung haben.

AWS KMS unterstützt die folgenden Gewährungsbeschränkungen für den Verschlüsselungskontext.

EncryptionContextEquals

Verwenden Sie `EncryptionContextEquals`, um den exakten Verschlüsselungskontext für zulässige Anforderungen anzugeben.

`EncryptionContextEquals` erfordert, dass die Verschlüsselungskontext-Paare in der Anforderung genau mit den Verschlüsselungskontext-Paaren in der Erteilungs-Einschränkung übereinstimmen (inkl. Groß-/Kleinschreibung). Die Paare können in beliebiger Reihenfolge erscheinen. Die Schlüssel und Werte in den Paaren können nicht variieren.

Zum Beispiel, wenn die `EncryptionContextEquals`-Erteilungs-Einschränkung das `"Department": "IT"`-Verschlüsselungskontext-Paar erfordert, erlaubt die Erteilung Anforderungen des angegebenen Typs nur dann, wenn der Verschlüsselungskontext in der Anforderung genau `"Department": "IT"` ist.

EncryptionContextSubset

Verwenden Sie `EncryptionContextSubset`, um zu erfordern, dass Anforderungen bestimmte Verschlüsselungskontext-Paare enthalten.

`EncryptionContextSubset` erfordert, dass die Anforderung alle Verschlüsselungskontext-Paare in der Erteilungs-Einschränkung enthält (genaue Übereinstimmung, inkl. Groß-/Kleinschreibung), aber die Anforderung kann auch zusätzliche Verschlüsselungskontext-Paare enthalten. Die Paare können in beliebiger Reihenfolge erscheinen. Die Schlüssel und Werte in den Paaren können nicht variieren.

Zum Beispiel, wenn die `EncryptionContextSubset`-Erteilungs-Einschränkung das `Department=IT`-Verschlüsselungskontext-Paar erfordert, erlaubt die Erteilung Anforderungen des angegebenen Typs nur dann, wenn der Verschlüsselungskontext in der Anforderung `"Department": "IT"` ist oder `"Department": "IT"` enthält, zusammen mit anderen Verschlüsselungskontext-Paaren, z. B. `"Department": "IT","Purpose": "Test"`.

Verwenden Sie den `Constraints` Parameter im CreateGrantVorgang, um eine Einschränkung für den Verschlüsselungskontext in einer Gewährung für einen symmetrischen Verschlüsselungs-KMS-Schlüssel anzugeben. Die Erteilung, die dieser Befehl erstellt, gewährt Benutzern, die die `keyUserRole`-Rolle annehmen dürfen, die Erlaubnis, die Decrypt-Operation aufzurufen. Diese Berechtigung ist jedoch nur wirksam, wenn der Verschlüsselungskontext in der `Decrypt`-Anforderung ein `"Department": "IT"`-Verschlüsselungskontext-Paar ist.

```
$ aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \
    --operations Decrypt \
    --retiring-principal arn:aws:iam::111122223333:role/adminRole \
    --constraints EncryptionContextEquals={Department=IT}
```

Die resultierende Erteilung sieht wie die folgende aus. Beachten Sie, dass die erteilte Berechtigung für die `keyUserRole`-Rolle nur wirksam ist, wenn die `Decrypt`-Anforderung das

Verschlüsselungskontextpaar enthält, das in der Erteilungseinschränkung angegeben ist. Verwenden Sie den ListGrantsVorgang, um die Grants für einen KMS-Schlüssel zu finden.

```
$ aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Grants": [
        {
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Operations": [
                "Decrypt"
            ],
            "GranteePrincipal": "arn:aws:iam::111122223333:role/keyUserRole",
            "Constraints": {
                "EncryptionContextEquals": {
                    "Department": "IT"
                }
            },
            "CreationDate": 1568565290.0,
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole"
        }
    ]
}
```

Um die `EncryptionContextEquals`-Erteilungs-Einschränkung zu erfüllen, muss der Verschlüsselungskontext in der Anforderung für die `Decrypt`-Produktion ein `"Department": "IT"`-Paar sein. Eine Anforderung wie die folgende vom Empfänger-Prinzipal würde die `EncryptionContextEquals`-Erteilungs-Einschränkung erfüllen.

```
$ aws kms decrypt \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab\
    --ciphertext-blob fileb://encrypted_msg \
    --encryption-context Department=IT
```

Wenn die Erteilungs-Einschränkung `EncryptionContextSubset` ist, müssen die Verschlüsselungskontext-Paare in der Anforderung die Verschlüsselungskontext-Paare

in der Erteilungs-Einschränkung beinhalten, aber die Anforderung kann auch andere Verschlüsselungskontext-Paare enthalten. Die folgende Erteilungs-Einschränkung erfordert, dass eines der Verschlüsselungskontext-Paare in der Anforderung `"Deparment": "IT"` ist.

```
"Constraints": {
   "EncryptionContextSubset": {
       "Department": "IT"
   }
}
```

Eine Anforderung wie die folgende vom Empfänger-Prinzipal würde sowohl die `EncryptionContextEqual`- und `EncryptionContextSubset`-Erteilungs-Einschränkung in diesem Beispiel erfüllen.

```
$ aws kms decrypt \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --ciphertext-blob fileb://encrypted_msg \
    --encryption-context Department=IT
```

Eine Anforderung wie die folgende vom Empfänger-Prinzipal würde die `EncryptionContextSubset`-Erteilungs-Einschränkung erfüllen, aber nicht die `EncryptionContextEquals`-Erteilungs-Einschränkung.

```
$ aws kms decrypt \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --ciphertext-blob fileb://encrypted_msg \
    --encryption-context Department=IT,Purpose=Test
```

AWS Dienste verwenden häufig Einschränkungen des Verschlüsselungskontextes in den Zuweisungen, die ihnen die Erlaubnis geben, KMS-Schlüssel in Ihrem System zu verwenden AWS-Konto. Beispielsweise verwendet Amazon DynamoDB eine Erteilung wie die folgende, um die Berechtigung zu erhalten, den Von AWS verwalteter Schlüssel für DynamoDB in Ihrem Konto zu verwenden. Die `EncryptionContextSubset`-Erteilungseinschränkung in dieser Erteilung macht die Berechtigungen in der Erteilung nur wirksam, wenn der Verschlüsselungskontext in der Anforderung `"subscriberID": "111122223333"`- und `"tableName": "Services"`-Paare enthält. Diese Erteilungs-Einschränkung bedeutet, dass die Erteilung es DynamoDB erlaubt, den angegebenen KMS-Schlüssel nur für eine bestimmte Tabelle in Ihrem AWS-Konto zu verwenden.

Um diese Ausgabe zu erhalten, führen Sie den ListGrantsVorgang Von AWS verwalteter Schlüssel für DynamoDB in Ihrem Konto aus.

```
$ aws kms list-grants --key-id 0987dcba-09fe-87dc-65ba-ab0987654321

{
    "Grants": [
        {
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "ReEncryptFrom",
                "ReEncryptTo",
                "RetireGrant",
                "DescribeKey"
            ],
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "Constraints": {
                "EncryptionContextSubset": {
                    "aws:dynamodb:tableName": "Services",
                    "aws:dynamodb:subscriberId": "111122223333"
                }
            },
            "CreationDate": 1518567315.0,
            "KeyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
            "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",
            "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",
            "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",
            "GrantId":
 "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59"
        }
    ]
}
```

Genehmigung erteilen CreateGrant

Eine Erteilung kann die Berechtigung zum Aufrufen der `CreateGrant`-Produktion enthalten. Aber wenn ein Empfänger-Prinzipal die Berechtigung zum Aufrufen von `CreateGrant` von einer Erteilung erhält, und nicht von einer Richtlinie, ist diese Berechtigung eingeschränkt.

- Der Empfänger-Prinzipal kann nur Erteilungen erlauben, die einige oder alle Operationen in der übergeordneten Erteilung zulassen.
- Die Erteilungs-Einschränkungen in den Erteilungen, die sie erstellen, müssen mindestens so streng sein wie die in der übergeordneten Erteilung.

Diese Einschränkungen gelten nicht für Prinzipale, die die `CreateGrant`-Berechtigung aus einer Richtlinie erhalten, obwohl ihre Berechtigungen durch Richtlinienbedingungen eingeschränkt werden können.

Nehmen wir beispielsweise eine Erteilung, die es dem empfangenden Prinzipal ermöglicht, die `GenerateDataKey`-, `Decrypt`- und `CreateGrant`-Operationen aufzurufen. Wir nennen eine Erteilung, die die `CreateGrant`-Berechtigung erlaubt, eine übergeordnete Erteilung.

```
# The original grant in a ListGrants response.
{
    "Grants": [
        {
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1572216195.0,
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Operations": [
                "GenerateDataKey",
                "Decrypt",
                "CreateGrant
            ]
            "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole",
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GranteePrincipal": "arn:aws:iam::111122223333:role/keyUserRole",
            "Constraints": {
                "EncryptionContextSubset": {
                    "Department": "IT"
                }
            },
        }
    ]
}
```

Der Empfänger-Prinzipal exampleUser kann diese Berechtigung verwenden, um eine Erteilung zu erstellen, die eine Teilmenge der in der ursprünglichen Erteilung angegebenen Operationen enthält – z. B. `CreateGrant` und `Decrypt`. Die untergeordnete Erteilung kann keine anderen Operationen enthalten (z. B. `ScheduleKeyDeletion` oder `ReEncrypt`).

Außerdem müssen die Erteilungs-Beschränkungen in untergeordneten Erteilungen mindestens ebenso restriktiv sein, wie die in der übergeordneten Erteilungen. So kann die untergeordnete Erteilung beispielsweise Paare zu einer `EncryptionContextSubset`-Beschränkung in der übergeordneten Erteilung hinzufügen. Sie kann sie jedoch nicht entfernen. Die untergeordnete Erteilung kann eine `EncryptionContextSubset`-Beschränkung in eine `EncryptionContextEquals`-Beschränkung ändern – nicht jedoch umgekehrt.

Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Beispielsweise kann der Empfänger-Prinzipal die `CreateGrant`-Berechtigung verwenden, die er von der übergeordneten Erteilung erhalten hat, um die folgende untergeordnete Erteilung zu erstellen Die Operationen in der untergeordneten Erteilung sind eine Teilmenge der Operationen in der übergeordneten Erteilung, und die Erteilungs-Beschränkungen sind restriktiver.

```
# The child grant in a ListGrants response.
{
    "Grants": [
        {
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1572249600.0,
            "GrantId":
 "fedcba9999c1e2e9876abcde6e9d6c9b6a1987650000abcee009abcdef40183f",
            "Operations": [
                "CreateGrant"
                "Decrypt"
            ]
            "RetiringPrincipal": "arn:aws:iam::111122223333:user/exampleUser",
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GranteePrincipal": "arn:aws:iam::111122223333:user/anotherUser",
            "Constraints": {
                "EncryptionContextEquals": {
```

```
                    "Department": "IT"
                }
            },
        }
    ]
}
```

Der Empfänger-Prinzipal in der untergeordneten Erteilung, `anotherUser`, kann seine `CreateGrant`-Berechtigung zum Erstellen von Erteilungen verwenden. Allerdings müssen die Erteilungen, die `anotherUser` erstellt, die Operationen in der übergeordneten Erteilung oder in einer Teilmenge enthalten, und die Erteilungs-Einschränkungen müssen gleichwertig oder strenger sein.

Anzeigen einer Erteilung

Verwenden Sie den ListGrantsVorgang, um den Zuschuss einzusehen. Sie müssen den KMS-Schlüssel angeben, für den die Erteilungen gelten. Sie können die Liste der Erteilungen auch nach Erteilungs-ID oder Empfänger-Prinzipal filtern. Weitere Beispiele finden Sie unter Verwendung `ListGrants` mit einem AWS SDK oder CLI.

Um alle Zuschüsse in der Region AWS-Konto und mit einem bestimmten ausscheidenden Schulleiter einzusehen, verwenden Sie ListRetirableGrants. Die Antworten enthalten Details zu den einzelnen Erteilungen.

> **Note**
>
> Das `GranteePrincipal`-Feld in der `ListGrants`-Antwort enthält normalerweise den Berechtigungsprinzipal der Genehmigung. Handelt es sich bei der Hauptperson des Zuschusses jedoch um eine AWS Dienstleistung, enthält das `GranteePrincipal` Feld den Dienstprinzipal, der für mehrere verschiedene Schulleiter stehen kann.

Mit dem folgenden Befehl werden beispielsweise alle Erteilungen für einen KMS-Schlüssel aufgeführt.

```
$  aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Grants": [
        {
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
```

```
            "CreationDate": 1572216195.0,
            "GrantId":
 "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
            "Constraints": {
                "EncryptionContextSubset": {
                    "Department": "IT"
                }
            },
            "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole",
            "Name": "",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "GranteePrincipal": "arn:aws:iam::111122223333:user/exampleUser",
            "Operations": [
                "Decrypt"
            ]
        }
    ]
}
```

Verwenden eines Erteilungs-Token

Die AWS KMS API folgt einem späteren Konsistenzmodell. Beim Erstellen einer Erteilung ist die Erteilung möglicherweise nicht sofort gültig. Es kann zu einer kurzen Verzögerung kommen, bevor die Änderung in allen Bereichen von AWS KMS verfügbar ist. In der Regel dauert es weniger als ein paar Sekunden, bis sich die Änderung im gesamten System verbreitet, in einigen Fällen kann es jedoch mehrere Minuten dauern. Sobald sich die Änderung vollständig im System verbreitet hat, kann der Empfänger-Prinzipal die Berechtigungen in der Berechtigungserteilung verwenden, ohne das Berechtigungserteilungs-Token oder irgendwelche Beweise für die Berechtigungserteilung anzugeben. Wenn ein Zuschuss jedoch so neu ist, dass er noch nicht allen bekannt ist AWS KMS, schlägt die Anfrage möglicherweise mit einem `AccessDeniedException` Fehler fehl.

Um die Berechtigungen in einer neuen Erteilung sofort zu verwenden, verwenden Sie den Erteilungs-Token für die Erteilung. Speichern Sie das Grant-Token, das der CreateGrantVorgang zurückgibt. Reichen Sie dann das Grant-Token in der Anfrage für den AWS KMS Vorgang ein. Sie können ein Grant-Token für jede AWS KMS Grant-Operation einreichen und Sie können mehrere Grant-Token in derselben Anfrage einreichen.

Im folgenden Beispiel wird der `CreateGrant` Vorgang verwendet, um einen Zuschuss zu erstellen, der die Operationen GenerateDataKeyund Decrypt ermöglicht. Sie speichert das Erteilungs-Token,

das `CreateGrant` in der `token`-Variable zurückgibt. Dann verwendet sie in einem Aufruf an die `GenerateDataKey`-Produktion das Erteilungs-Token in der `token`-Variable.

```
# Create a grant; save the grant token
$ token=$(aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:user/appUser \
    --retiring-principal arn:aws:iam::111122223333:user/acctAdmin \
    --operations GenerateDataKey Decrypt \
    --query GrantToken \
    --output text)

# Use the grant token in a request
$ aws kms generate-data-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --key-spec AES_256 \
    --grant-tokens $token
```

Schulleiter mit entsprechender Genehmigung können ein Grant-Token auch verwenden, um einen neuen Grant zurückzuziehen, noch bevor der Grant vollständig verfügbar ist. AWS KMS(Die `RevokeGrant`-Produktion akzeptiert kein Erteilungs-Token.) Details hierzu finden Sie unter Außerbetriebnahme und Widerruf von Erteilungen.

```
# Retire the grant
$ aws kms retire-grant --grant-token $token
```

Außerbetriebnahme und Widerruf von Erteilungen

Um eine Erteilung zu löschen, müssen Sie sie aufheben oder widerrufen.

Die RevokeGrantOperationen RetireGrantund sind einander sehr ähnlich. Beide Operationen löschen eine Erteilung, wodurch die Berechtigungen, die die Erteilung erlaubt, eliminiert werden. Der Hauptunterschied zwischen diesen Operationen besteht darin, wie sie autorisiert werden.

RevokeGrant

Wie bei den meisten AWS KMS Vorgängen wird der Zugriff auf den `RevokeGrant` Vorgang durch wichtige Richtlinien und IAM-Richtlinien gesteuert. Die RevokeGrantAPI kann von jedem Principal mit entsprechender `kms:RevokeGrant` Genehmigung aufgerufen werden. Diese Berechtigung ist in den Standard-Berechtigungen enthalten, die Schlüsseladministratoren erteilt werden. In

der Regel widerrufen Administratoren eine Erteilung, um Berechtigungen zu verweigern, die die Erteilung erlaubt.

RetireGrant

Die Erteilung bestimmt, wer sie aufheben kann. Mit diesem Entwurf können Sie den Lebenszyklus einer Erteilung steuern, ohne Schlüsselrichtlinien oder IAM-Richtlinien zu ändern. In der Regel können Sie eine Erteilung aufheben, wenn Sie die Berechtigungen nicht mehr benötigen.

Eine Erteilung kann durch einen optionalen ausscheidenden Prinzipal, der in der Erteilung angegeben ist, aufgehoben werden. Der Empfänger-Prinzipal kann die Erteilung auch aufheben, aber nur, wenn er auch ein ausscheidender Prinzipal ist oder die Erteilung die `RetireGrant`-Produktion enthält. Als Backup kann der Grant, AWS-Konto in dem der Grant erstellt wurde, den Grant zurückziehen.

Es gibt eine `kms:RetireGrant`-Berechtigung, die in IAM-Richtlinien verwendet werden kann, aber sie verfügt über eingeschränkte Funktionalität. Prinzipale, die in der Erteilung angegeben sind, können eine Erteilung ohne die `kms:RetireGrant`-Berechtigung aufheben. Die `kms:RetireGrant`-Berechtigung allein erlaubt es Prinzipalen nicht, eine Erteilung aufzuheben. Die `kms:RetireGrant` Berechtigung ist in einer wichtigen Richtlinie oder Ressourcenkontrollrichtlinie nicht wirksam.

- Um die Genehmigung zur Einstellung eines Zuschusses zu verweigern, können Sie eine `Deny` Aktion mit der entsprechenden `kms:RetireGrant` Genehmigung in Ihren IAM-Richtlinien verwenden.
- Derjenige AWS-Konto , dem der KMS-Schlüssel gehört, kann die `kms:RetireGrant` Berechtigung an einen IAM-Prinzipal im Konto delegieren.
- Wenn es sich bei dem ausscheidenden Prinzipal um einen anderen handelt AWS-Konto, können die Administratoren des anderen Kontos die Erlaubnis `kms:RetireGrant` zum Ausscheiden des Zugriffs an einen IAM-Prinzipal in diesem Konto delegieren.

Die AWS KMS API folgt einem späteren Konsistenzmodell. Wenn Sie eine Erteilung erstellen, aufheben oder widerrufen, kann es zu einer kurzen Verzögerung kommen, bevor die Änderung in allen Bereichen von AWS KMS verfügbar ist. In der Regel dauert es weniger als ein paar Sekunden, bis sich die Änderung im gesamten System verbreitet, in einigen Fällen kann es jedoch mehrere Minuten dauern. Wenn Sie einen neuen Zuschuss sofort löschen müssen, bevor er vollständig verfügbar ist, verwenden Sie ein Grant-Token AWS KMS, um den Grant zurückzuziehen. Sie können kein Erteilungs-Token verwenden, um eine Erteilung zu widerrufen.

Zustandstasten für AWS KMS

Sie können Bedingungen in den Schlüsselrichtlinien und IAM-Richtlinien angeben, die den Zugriff auf AWS KMS Ressourcen steuern. Die Richtlinienanweisung ist nur wirksam, wenn diese Bedingungen erfüllt sind. Beispielsweise kann festgelegt werden, dass eine Richtlinienanweisung erst ab einem bestimmten Datum gilt. Sie können auch festlegen, dass eine Richtlinienanweisung den Zugriff nur steuert, wenn in einer API-Anforderung ein bestimmter Wert vorhanden ist.

Verwenden Sie Bedingungsschlüssel im `Condition`-Element einer Richtlinienanweisung mit IAM-Bedingungs-Operatoren, um Bedingungen festzulegen. Einige Bedingungsschlüssel gelten allgemein für AWS, andere sind spezifisch für. AWS KMS

Bedingungsschlüsselwerte müssen den Zeichen- und Kodierungsregeln für AWS KMS wichtige Richtlinien und IAM-Richtlinien entsprechen. Weitere Informationen zu wichtigen Dokumentenregeln für Schlüsselrichtlinien finden Sie unter Schlüsselrichtlinienformat. Weitere Informationen zu Regeln für IAM-Richtliniendokumente finden Sie unter Anforderungen für den IAM-Namen im IAM-Benutzerhandbuch.

Themen

- AWS globale Bedingungsschlüssel
- AWS KMS Bedingungsschlüssel
- AWS KMS Bedingungsschlüssel für AWS Nitro Enclaves

AWS globale Bedingungsschlüssel

AWS definiert globale Bedingungsschlüssel, eine Reihe von Schlüsseln für Richtlinienbedingungen für alle AWS Dienste, die IAM für die Zugriffskontrolle verwenden. AWS KMS unterstützt alle globalen Bedingungsschlüssel. Sie können sie in AWS KMS wichtigen Richtlinien und IAM-Richtlinien verwenden.

Sie können beispielsweise den PrincipalArn globalen Bedingungsschlüssel aws: verwenden, um den Zugriff auf einen AWS KMS key (KMS-Schlüssel) nur dann zuzulassen, wenn der Principal in der Anfrage durch den Amazon-Ressourcennamen (ARN) im Bedingungsschlüsselwert repräsentiert wird. Um die attributebasierte Zugriffskontrolle (ABAC) in zu unterstützen AWS KMS, können Sie den globalen Bedingungsschlüssel aws:ResourceTag/tag-key in einer IAM-Richtlinie verwenden, um den Zugriff auf KMS-Schlüssel mit einem bestimmten Tag zu ermöglichen.

Um zu verhindern, dass ein AWS Dienst in einer Richtlinie, in der der Principal ein AWS Dienstprinzipal ist, als verwirrter Stellvertreter verwendet wird, können Sie die globalen Bedingungsschlüssel oder verwenden. aws:SourceArnaws:SourceAccount Details hierzu finden Sie unter Verwenden der Bedingungsschlüssel `aws:SourceArn` oder `aws:SourceAccount`.

Informationen zu AWS globalen Bedingungsschlüsseln, einschließlich der Anforderungstypen, in denen sie verfügbar sind, finden Sie unter AWS Globale Bedingungskontextschlüssel im IAM-Benutzerhandbuch. Beispiele für die Verwendung globaler Bedingungsschlüssel in IAM-Richtlinien finden Sie unter Steuern des Zugriffs auf Anforderungen und Steuern von Tag-Schlüsseln im IAM-Benutzerhandbuch.

Die folgenden Themen bieten spezielle Anleitungen für die Verwendung von Bedingungsschlüsseln basierend auf IP-Adressen und VPC-Endpunkten.

Themen

- Verwenden der IP-Adressbedingung in Richtlinien mit AWS KMS -Berechtigungen
- Verwenden von VPC-Endpunkt-Bedingungen in Richtlinien mit AWS KMS -Berechtigungen
- Verwendung von IPv6 Adressen in IAM und in wichtigen Richtlinien AWS KMS

Verwenden der IP-Adressbedingung in Richtlinien mit AWS KMS -Berechtigungen

Sie können AWS KMS es verwenden, um Ihre Daten in einem integrierten AWS Service zu schützen. Seien Sie jedoch vorsichtig, wenn Sie die Operatoren für die `aws:SourceIp` IP-Adressbedingung oder den Bedingungsschlüssel in derselben Richtlinienerklärung angeben, die den Zugriff AWS KMS gewährt oder verweigert. Beispielsweise beschränkt die Richtlinie in AWS: Verweigert den Zugriff auf AWS Basierend auf der Quell-IP AWS Aktionen auf Anfragen aus dem angegebenen IP-Bereich.

Betrachten Sie folgendes Szenario:

1. Sie fügen einer IAM-Identität eine Richtlinie wie die unter AWS: Verweigert den Zugriff auf AWS Basierend auf der Quell-IP gezeigte Richtlinie an. Sie legen als Wert für den `aws:SourceIp`-Bedingungsschlüssel den IP-Adressbereich für das Unternehmen des Benutzers fest. An diese IAM-Identität sind weitere Richtlinien angehängt, die es ihr ermöglichen, Amazon EBS EC2, Amazon und zu verwenden. AWS KMS
2. Die Identität versucht, ein verschlüsseltes EBS-Volume an eine Instance anzuhängen. EC2 Diese Aktion schlägt aufgrund eines Autorisierungsfehlers fehl, obwohl der Benutzer zur Verwendung aller relevanten Services berechtigt ist.

Schritt 2 schlägt fehl, weil die Anfrage AWS KMS zur Entschlüsselung des verschlüsselten Datenschlüssels des Volumes von einer IP-Adresse stammt, die mit der EC2 Amazon-Infrastruktur verknüpft ist. Dieser Schritt wird nur erfolgreich durchgeführt, wenn die Anforderung von der IP-Adresse des ursprünglichen Benutzers stammt. Da die Richtlinie in Schritt 1 ausdrücklich alle Anfragen von anderen als den angegebenen IP-Adressen ablehnt, EC2 wird Amazon die Erlaubnis verweigert, den verschlüsselten Datenschlüssel des EBS-Volumes zu entschlüsseln.

Weiterhin ist der Bedingungsschlüssel `aws:SourceIP` nicht wirksam, wenn die Anforderung von einem Amazon-VPC-Endpunkt kommt. Um Anforderungen an einen VPC-Endpunkt, einschließlich eines AWS KMS -VPC-Endpunkts zu beschränken, verwenden Sie die `aws:SourceVpce`- oder `aws:SourceVpc`-Bedingungsschlüssel. Weitere Informationen finden Sie unter VPC-Endpunkte – Steuern der Nutzung von Endpunkten im Amazon VPC-Benutzerhandbuch.

Verwenden von VPC-Endpunkt-Bedingungen in Richtlinien mit AWS KMS -Berechtigungen

AWS KMS unterstützt Amazon Virtual Private Cloud (Amazon VPC) -Endpunkte, die von betrieben werden. AWS PrivateLink Sie können die folgenden globalen Bedingungsschlüssel in wichtigen Richtlinien und IAM-Richtlinien verwenden, um den Zugriff auf AWS KMS Ressourcen zu steuern, wenn die Anfrage von einer VPC kommt oder einen VPC-Endpunkt verwendet. Details hierzu finden Sie unter Verwenden Sie VPC-Endpunkte, um den Zugriff auf Ressourcen zu kontrollieren AWS KMS.

- `aws:SourceVpc` beschränkt den Zugriff auf Anforderungen von der angegebenen VPC.
- `aws:SourceVpce` beschränkt den Zugriff auf Anforderungen vom angegebenen VPC-Endpunkt.

Wenn Sie diese Bedingungsschlüssel verwenden, um den Zugriff auf KMS-Schlüssel zu steuern, verweigern Sie möglicherweise versehentlich den Zugriff auf AWS Dienste, die in Ihrem Namen verwendet werden. AWS KMS

Seien Sie sorgsam darum bemüht, eine Situation wie das IP-Adressen-Bedingungsschlüssel Beispiel zu vermeiden. Wenn Sie Anfragen für einen KMS-Schlüssel auf einen VPC- oder VPC-Endpunkt beschränken, schlagen Aufrufe AWS KMS von einem integrierten Service wie Amazon S3 oder Amazon EBS möglicherweise fehl. Dies kann auch dann vorkommen, wenn die Quell-Anforderung letztendlich von der VPC oder dem VPC-Endpunkt stammt.

Verwendung von IPv6 Adressen in IAM und in wichtigen Richtlinien AWS KMS

Stellen Sie vor dem Zugriff sicher IPv6, AWS KMS dass alle Schlüssel- und IAM-Richtlinien, die IP-Adressbeschränkungen enthalten, so aktualisiert wurden, dass sie auch IPv6 Adressbereiche enthalten. IP-basierte Richtlinien, die nicht für den Umgang mit IPv6 Adressen aktualisiert wurden, können dazu führen, dass Clients fälschlicherweise Zugriff verlieren oder erhalten, wenn sie sie verwenden IPv6. Allgemeine Hinweise zu KMS-Zugriffskontrollen finden Sie unterKMS-Schlüsselzugriff und Berechtigungen. Weitere Informationen zur KMS- und Dual-Stack-Unterstützung finden Sie unterUnterstützung für Dual-Stack-Endgeräte.

> **Important**
>
> Diese Anweisungen lassen keine Aktionen zu. Verwenden Sie diese Anweisungen in Kombination mit anderen Anweisungen, die bestimmte Aktionen zulassen.

Die folgende Anweisung verweigert ausdrücklich den Zugriff auf alle KMS-Berechtigungen für Anfragen, die aus dem `192.0.2.*` IPv4 Adressbereich stammen. Allen IP-Adressen außerhalb dieses Bereichs werden KMS-Berechtigungen nicht ausdrücklich verweigert. Da sich alle IPv6 Adressen außerhalb des verweigerten Bereichs befinden, verweigert diese Anweisung nicht ausdrücklich KMS-Berechtigungen für IPv6 Adressen.

```
{
     "Sid": "DenyKMSPermissions",
     "Effect": "Deny",
    "Action": [
        "kms:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24"
            ]
        }
    }
}
```

Sie können das `Condition` Element so ändern, dass sowohl IPv4 (`192.0.2.0/24`) als auch IPv6 (`2001:db8:1234::/32`) Adressbereiche verweigert werden, wie im folgenden Beispiel gezeigt.

```
{
    "Sid": "DenyKMSPermissions",
    "Effect": "Deny",
    "Action": [
        "kms:*"
    ],
    "Resource": "*",
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24",
                "2001:db8:1234::/32"
            ]
        }
    }
}
```

AWS KMS Bedingungsschlüssel

AWS KMS stellt eine Reihe von Bedingungsschlüsseln bereit, die Sie in wichtigen Richtlinien und IAM-Richtlinien verwenden können. Diese Bedingungsschlüssel sind spezifisch für. AWS KMS Sie können beispielsweise den Bedingungsschlüssel `kms:EncryptionContext:`*`context-key`* verwenden, damit ein bestimmter Verschlüsselungskontext bei der Steuerung des Zugriffs auf einen KMS-Schlüssel zur symmetrischen Verschlüsselung erforderlich ist.

Bedingungen für die Anforderung einer API-Produktion

Viele AWS KMS Bedingungsschlüssel steuern den Zugriff auf einen KMS-Schlüssel auf der Grundlage des Werts eines Parameters in der Anforderung für einen AWS KMS Vorgang. Sie können beispielsweise den KeySpec Bedingungsschlüssel kms: in einer IAM-Richtlinie verwenden, um die Verwendung des CreateKeyVorgangs nur dann zuzulassen, wenn der Wert des `KeySpec` Parameters in der `CreateKey` Anforderung lautet`RSA_4096`.

Dieser Bedingungstyp funktioniert sogar dann, wenn der Parameter nicht in der Anforderung angezeigt wird, z. B. wenn Sie den Standardwert des Parameters verwenden. Sie können beispielsweise den KeySpec Bedingungsschlüssel kms: verwenden, um Benutzern zu ermöglichen, die `CreateKey` Operation nur dann zu verwenden, wenn der Wert des `KeySpec` Parameters ist`SYMMETRIC_DEFAULT`, was der Standardwert ist. Diese Bedingung gewährt Anforderungen mit dem Parameter `KeySpec` und dem Wert `SYMMETRIC_DEFAULT` sowie Anforderungen ohne den Parameter `KeySpec`.

Bedingungen für KMS-Schlüssel, die in API-Operationen verwendet werden

Einige AWS KMS Bedingungsschlüssel können den Zugriff auf Operationen auf der Grundlage einer Eigenschaft des KMS-Schlüssels steuern, der für den Vorgang verwendet wird. Sie können beispielsweise die KeyOrigin Bedingung kms: verwenden, um es Prinzipalen zu ermöglichen, einen KMS-Schlüssel nur dann GenerateDataKeyaufzurufen, wenn `Origin` der KMS-Schlüssel aktiviert ist`AWS_KMS`. Um herauszufinden, ob ein Bedingungsschlüssel auf diese Weise verwendet werden kann, lesen Sie die Beschreibung des Bedingungsschlüssels.

Bei der Produktion muss es sich um eine KMS-Schlüsselressourcen-Produktion handeln, das heißt, eine Produktion, die für einen bestimmten KMS-Schlüssel autorisiert ist. Um die KMS-Schlüsselressourcen-Operationen zu identifizieren, suchen Sie in der Tabelle Actions and Resources (Aktionen und Ressourcen) Sie nach dem Wert von `KMS key` in der `Resources`-Spalte für die Produktion. Wenn Sie diese Art von Bedingungsschlüssel beispielsweise für einen Vorgang verwenden, der für eine bestimmte KMS-Schlüsselressource nicht autorisiert ist ListKeys, ist die Berechtigung nicht wirksam, da die Bedingung niemals erfüllt werden kann. Es ist keine KMS-Schlüssel-Ressource an der Autorisierung der Produktion `ListKeys` beteiligt und keine `KeySpec`-Eigenschaft.

In den folgenden Themen werden die einzelnen AWS KMS Bedingungsschlüssel beschrieben. Sie enthalten auch Beispiele für Richtlinienanweisungen, die die Richtliniensyntax veranschaulichen.

Verwenden von Satz-Operatoren mit Bedingungsschlüssel

Wenn eine Richtlinienbedingung zwei Gruppen von Werten vergleicht, z. B. die Gruppe von Tags in einer Anforderung und die Gruppe von Tags in einer Richtlinie, müssen Sie angeben, AWS wie die Gruppen miteinander verglichen werden sollen. IAM definiert für diesen Zweck zwei Satz-Operatoren, `ForAnyValue` und `ForAllValues`. Verwenden Sie Satz-Operatoren nur mit mehrwertigen Bedingungsschlüssel, die diese erfordern. Verwenden Sie keine Satz-Operatoren mit einzelwertigen Bedingungsschlüssel. Testen Sie Ihre Richtlinienanweisungen immer gründlich, bevor Sie sie in einer Operationsumgebung verwenden.

Bedingungsschlüssel sind einzelwertig oder mehrwertig. Informationen dazu, ob ein AWS KMS Bedingungsschlüssel ein- oder mehrwertig ist, finden Sie in der Spalte Wertetyp in der Beschreibung des Bedingungsschlüssels.

- Einzelwertige Bedingungsschlüssel haben höchstens einen Wert im Autorisierungskontext (die Anforderung oder Ressource). Da beispielsweise jeder API-Aufruf nur von einem API-Aufruf ausgehen kann AWS-Konto, CallerAccount ist kms: ein einwertiger Bedingungsschlüssel. Verwenden Sie keinen Satz-Operator mit einem einzelwertigen Bedingungsschlüssel.

- Mehrwertige Bedingungsschlüssel haben mehrfache Wert im Autorisierungskontext (die Anforderung oder Ressource). Da beispielsweise jeder KMS-Schlüssel mehrere Aliase haben kann, ResourceAliases kann kms: mehrere Werte haben. Mehrwertige Bedingungsschlüssel erfordern einen Satz-Operator.

Beachten Sie, dass der Unterschied zwischen einzelwertigen und mehrwertigen Bedingungsschlüsseln von der Anzahl der Werte im Autorisierungskontext abhängt, nicht von der Anzahl der Werte in der Richtlinienbedingung.

> **Warning**
>
> Wenn Sie einen Satz-Operator mit einem einzelwertigen Bedingungsschlüssel verwenden, können Sie eine Richtlinienanweisung erstellen, die übermäßig permissiv (oder zu restriktiv) ist. Verwenden Sie Satz-Operatoren nur mit mehrwertigen Bedingungsschlüssel, die diese erfordern.
>
> Wenn Sie eine Richtlinie erstellen oder aktualisieren, die einen `ForAllValues` Set-Operator mit dem Kontextschlüssel oder den `aws:RequestTag/`*`tag-key`* Bedingungsschlüsseln enthält, wird die folgende AWS KMS Fehlermeldung zurückgegeben kms:EncryptionContext:
>
> ```
> OverlyPermissiveCondition: Using the ForAllValues set operator with
> a single-valued condition key matches requests without the specified
> [encryption context or tag] or with an unspecified [encryption
> context or tag]. To fix, remove ForAllValues.
> ```

Ausführliche Informationen zu den `ForAnyValue`- und `ForAllValues`-Satz-Operatoren finden Sie unter Verwenden mehrerer Schlüssel und Werte im IAM-Benutzerhandbuch. Informationen zum Risiko der Verwendung des `ForAllValues` Set-Operators mit einer einwertigen Bedingung finden Sie unter Sicherheitswarnung — ForAllValues mit einwertigem Schlüssel im IAM-Benutzerhandbuch.

Themen

- km: BypassPolicyLockoutSafetyCheck
- km: CallerAccount
- kms: CustomerMasterKeySpec (veraltet)
- kms: CustomerMasterKeyUsage (veraltet)
- km: DataKeyPairSpec
- km: EncryptionAlgorithm

- kms:EncryptionContext: Kontextschlüssel
- km: EncryptionContextKeys
- km: ExpirationModel
- km: GrantConstraintType
- km: GrantIsFor AWSResource
- km: GrantOperations
- km: GranteePrincipal
- km: KeyAgreementAlgorithm
- km: KeyOrigin
- km: KeySpec
- km: KeyUsage
- km: MacAlgorithm
- km: MessageType
- km: MultiRegion
- km: MultiRegionKeyType
- km: PrimaryRegion
- km: ReEncryptOnSameKey
- km: RequestAlias
- km: ResourceAliases
- km: ReplicaRegion
- km: RetiringPrincipal
- km: RotationPeriodInDays
- km: ScheduleKeyDeletionPendingWindowInDays
- km: SigningAlgorithm
- km: ValidTo
- km: ViaService
- km: WrappingAlgorithm
- km: WrappingKeySpec

km: BypassPolicyLockoutSafetyCheck

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:BypassPolicyLockoutSafetyCheck`	Boolesch	Einzelwertig	`CreateKey` `PutKeyPolicy`	Nur IAM-Richtlinien Schlüsselrichtlinien und IAM-Richtlinien

Der `kms:BypassPolicyLockoutSafetyCheck` Bedingungsschlüssel steuert den Zugriff auf die PutKeyPolicyOperationen CreateKeyund auf der Grundlage des Werts des `BypassPolicyLockoutSafetyCheck` Parameters in der Anforderung.

Die folgende Beispiel-IAM-Richtlinienanweisung verhindert, dass Benutzer die Richtliniensperre-Sicherheitsprüfung umgehen, indem Sie die Berechtigung zum Erstellen von KMS-Schlüssel ablehnen, wenn der Wert des Parameters `BypassPolicyLockoutSafetyCheck` in der `CreateKey`-Anforderung `true.` lautet.

```
{
  "Effect": "Deny",
  "Action": [
    "kms:CreateKey",
    "kms:PutKeyPolicy"
  ],
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:BypassPolicyLockoutSafetyCheck": true
    }
  }
}
```

Außerdem können Sie den `kms:BypassPolicyLockoutSafetyCheck`-Bedingungsschlüssel in einer IAM- oder Schlüsselrichtlinie verwenden, um den Zugriff auf die `PutKeyPolicy`-Produktion zu steuern. Die folgende Beispiel-Richtlinienanweisung einer Schlüsselrichtlinie verhindert, dass

Benutzer beim Ändern der Richtlinie eines KMS-Schlüssel die Richtliniensperre-Sicherheitsprüfung umgehen.

Statt eine explizite `Deny`-Bedingung zu verwenden, nutzt diese Richtlinienanweisung `Allow` mit dem Null-bedingten Operator, um den Zugriff nur zu erlauben, wenn die Anforderung keinen `BypassPolicyLockoutSafetyCheck`-Parameter enthält. Wenn der Parameter nicht verwendet wird, lautet der Standardwert `false`. Diese etwas schwächere Richtlinienanweisung kann überschrieben werden, wenn ein Bypass erforderlich ist. Dies ist allerdings selten der Fall.

```
{
  "Effect": "Allow",
  "Action": "kms:PutKeyPolicy",
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:BypassPolicyLockoutSafetyCheck": true
    }
  }
}
```

Informationen finden Sie auch unter:

- km: KeySpec
- km: KeyOrigin
- km: KeyUsage

km: CallerAccount

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:CallerAccount`	String	Einzelwertig	KMS-Schlüsselressourcen-Operationen Benutzerdefinierter-Schlüss	Schlüsselrichtlinien und IAM-Richtlinien

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
			elspeicher-Operationen	

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf alle Identitäten (Benutzer und Rollen) in einem AWS-Konto zu gestatten oder verweigern. In Schlüsselrichtlinien verwenden Sie das Element `Principal`, um die Identitäten festzulegen, für die die Richtlinieanweisung gilt. Die Syntax für das `Principal`-Element bietet keine Möglichkeit, alle Identitäten in einem AWS-Konto festzulegen. Sie können diesen Effekt jedoch erzielen, indem Sie diesen Bedingungsschlüssel mit einem `Principal` Element kombinieren, das alle AWS Identitäten spezifiziert.

Sie können damit den Zugriff auf jeden KMS-Schlüsselressourcenvorgang steuern, d. h. auf jeden AWS KMS Vorgang, der einen bestimmten KMS-Schlüssel verwendet. Um die KMS-Schlüsselressourcen-Operationen zu identifizieren, suchen Sie in der Tabelle Actions and Resources (Aktionen und Ressourcen) Sie nach dem Wert von `KMS key` in der `Resources`-Spalte für die Operation. Er ist auch gültig für Operationen, die benutzerdefinierte Schlüsselspeicher verwalten.

Die folgende Schlüsselrichtlinienanweisung veranschaulicht beispielsweise die Verwendung des Bedingungsschlüssels `kms:CallerAccount`. Diese Grundsatzerklärung ist Teil der wichtigsten Richtlinie Von AWS verwalteter Schlüssel für Amazon EBS. Sie kombiniert ein `Principal` Element, das alle AWS Identitäten spezifiziert, mit dem `kms:CallerAccount` Bedingungsschlüssel, um effektiv den Zugriff auf alle Identitäten in 111122223333 zu ermöglichen. AWS-Konto Es enthält einen zusätzlichen AWS KMS Bedingungsschlüssel (`kms:ViaService`), um die Berechtigungen weiter einzuschränken, indem nur Anfragen zugelassen werden, die über Amazon EBS eingehen. Weitere Informationen finden Sie unter km: ViaService.

```
{
  "Sid": "Allow access through EBS for all principals in the account that are
 authorized to use EBS",
  "Effect": "Allow",
  "Principal": {"AWS": "*"},
  "Condition": {
    "StringEquals": {
      "kms:CallerAccount": "111122223333",
      "kms:ViaService": "ec2.us-west-2.amazonaws.com"
    }
```

```
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

kms: CustomerMasterKeySpec (veraltet)

Der Bedingungsschlüssel `kms:CustomerMasterKeySpec` ist veraltet. Verwenden Sie stattdessen den KeySpecBedingungsschlüssel kms:.

Die Bedingungsschlüssel `kms:CustomerMasterKeySpec` und `kms:KeySpec` funktionieren auf die gleiche Weise. Nur die Namen unterscheiden sich. Wir empfehlen Ihnen, `kms:KeySpec` zu verwenden. AWS KMS Unterstützt jedoch beide Bedingungsschlüssel, um fehlerhafte Änderungen zu vermeiden.

kms: CustomerMasterKeyUsage (veraltet)

Der Bedingungsschlüssel `kms:CustomerMasterKeyUsage` ist veraltet. Verwenden Sie stattdessen den KeyUsageBedingungsschlüssel kms:.

Die Bedingungsschlüssel `kms:CustomerMasterKeyUsage` und `kms:KeyUsage` funktionieren auf die gleiche Weise. Nur die Namen unterscheiden sich. Wir empfehlen Ihnen, `kms:KeyUsage` zu verwenden. AWS KMS Unterstützt jedoch beide Bedingungsschlüssel, um fehlerhafte Änderungen zu vermeiden.

km: DataKeyPairSpec

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:DataKeyPairSpec`	String	Einzelwertig	`GenerateDataKeyPair`	Schlüsselrichtlinien und IAM-Richtlinien

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
			`GenerateDataKeyPairWithoutPlaintext`	

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf die GenerateDataKeyPairWithoutPlaintextOperationen GenerateDataKeyPairund auf der Grundlage des `KeyPairSpec` Parameterwerts in der Anforderung zu steuern. Beispielsweise können Sie Benutzern erlauben, nur bestimmte Typen von Datenschlüsselpaaren zu generieren.

Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet den Bedingungsschlüssel `kms:DataKeyPairSpec`, um es Benutzern zu erlauben, mit dem KMS-Schlüssel nur RSA-Datenschlüsselpaare zu generieren.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:GenerateDataKeyPair",
    "kms:GenerateDataKeyPairWithoutPlaintext"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:DataKeyPairSpec": "RSA*"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: KeySpec
- the section called “km: EncryptionAlgorithm”

- the section called “kms:EncryptionContext: Kontextschlüssel”
- the section called “km: EncryptionContextKeys”

km: EncryptionAlgorithm

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp

`kms:EncryptionAlgorithm`	String	Einzelwertig	`Decrypt`
			`Encrypt`
			`GenerateDataKey`
			`GenerateDataKeyPair`
			`GenerateDataKeyPairWithoutPlaintext`
			`GenerateDataKeyWithoutPlaintext`
			`ReEncrypt`
				Schlüsselrichtlinien und IAM-Richtlinien

Mit dem Bedingungsschlüssel `kms:EncryptionAlgorithm` können Sie den Zugriff auf kryptografische Operationen basierend auf dem in der Produktion verwendeten Verschlüsselungsalgorithmus kontrollieren. Bei den ReEncryptVorgängen Verschlüsseln, Entschlüsseln und Entschlüsseln steuert es den Zugriff auf der Grundlage des Werts des EncryptionAlgorithmParameters in der Anforderung. Für Operationen, die Datenschlüssel

und Datenschlüsselpaare generieren, steuert er den Zugriff basierend auf dem Verschlüsselungsalgorithmus, der zum Verschlüsseln des Datenschlüssels verwendet wird.

Dieser Bedingungsschlüssel hat keine Auswirkung auf Operationen, die außerhalb von ausgeführt werden AWS KMS, wie z. B. die Verschlüsselung mit dem öffentlichen Schlüssel in einem asymmetrischen KMS-Schlüsselpaar außerhalb von. AWS KMS

EncryptionAlgorithm Parameter in einer Anfrage

Damit Benutzer nur einen bestimmten Verschlüsselungsalgorithmus mit einem KMS-Schlüssel verwenden können, verwenden Sie eine Richtlinienanweisung mit dem Effekt `Deny` und dem Bedingungsoperator `StringNotEquals`. Die folgende Schlüsselrichtlinienanweisung verhindert beispielsweise, dass Prinzipale, die die `ExampleRole`-Rolle annehmen können, diesen KMS-Schlüssel in den angegebenen kryptografischen Operationen verwenden, es sei denn, der Verschlüsselungsalgorithmus in der Anforderung ist `RSAES_OAEP_SHA_256` (ein asymmetrischer Verschlüsselungsalgorithmus, der mit RSA-KMS-Schlüsseln verwendet wird).

Im Gegensatz zu einer Richtlinienanweisung, die es einem Benutzer erlaubt, einen bestimmten Verschlüsselungsalgorithmus zu verwenden, hindert eine Richtlinienanweisung mit einem doppelten Negativwert wie diese andere Richtlinien und Erteilungen für diesen KMS-Schlüssel daran, dieser Rolle die Verwendung anderer Verschlüsselungsalgorithmen zu ermöglichen. Das `Deny` in dieser Schlüsselrichtlinienanweisung hat Vorrang vor allen Schlüsselrichtlinien oder IAM-Richtlinien mit `Allow`-Wirkung, und es hat vor allen Erteilungen für diesen KMS-Schlüssel und seine Prinzipale Vorrang.

```
{
  "Sid": "Allow only one encryption algorithm with this asymmetric KMS key",
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*"
  ],
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "kms:EncryptionAlgorithm": "RSAES_OAEP_SHA_256"
    }
```

```
  }
}
```

Der für die Produktion verwendete Verschlüsselungsalgorithmus

Sie können auch den `kms:EncryptionAlgorithm`-Bedingungsschlüssel verwenden, um den Zugriff auf Operationen basierend auf dem in der Produktion verwendeten Verschlüsselungsalgorithmus zu steuern, auch wenn der Algorithmus nicht in der Anforderung angegeben ist. Auf diese Weise können Sie den `SYMMETRIC_DEFAULT`-Algorithmus, der möglicherweise nicht in einer Anforderung angegeben wird, erfordern oder verbieten, da es sich um den Standardwert handelt.

Mit diesem Feature können Sie den `kms:EncryptionAlgorithm`-Bedingungsschlüssel verwenden, um den Zugriff auf die Operationen zu steuern, die Datenschlüssel und Datenschlüsselpaare generieren. Diese Operationen verwenden nur KMS-Schlüssel zur symmetrischen Verschlüsselung und den `SYMMETRIC_DEFAULT`-Algorithmus.

Beispielsweise beschränkt diese IAM-Richtlinie ihre Prinzipale auf symmetrische Verschlüsselung. Sie verweigert den Zugriff auf jeden KMS-Schlüssel im Beispielkonto für kryptografische Operationen, es sei denn, der in der Anforderung angegebene oder in der Produktion verwendete Verschlüsselungsalgorithmus ist SYMMETRIC_DEFAULT. `GenerateDataKey*`Einschließlich der GenerateDataKeyWithoutPlaintextErweiterungen GenerateDataKey, GenerateDataKeyPair, und GenerateDataKeyPairWithoutPlaintextzu den Berechtigungen. Die Bedingung hat keine Auswirkungen auf diese Vorgänge, da sie immer einen symmetrischen Verschlüsselungsalgorithmus verwenden.

```
{
  "Sid": "AllowOnlySymmetricAlgorithm",
  "Effect": "Deny",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringNotEquals": {
      "kms:EncryptionAlgorithm": "SYMMETRIC_DEFAULT"
    }
```

```
  }
}
```

Informationen finden Sie auch unter:

- the section called “km: MacAlgorithm”
- km: SigningAlgorithm

kms:EncryptionContext: Kontextschlüssel

AWS KMS Bedingungsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp

`kms:EncryptionContext:` *`context-key`*	String	Einzelwertig	`CreateGrant`
			`Encrypt`
			`Decrypt`
			`GenerateDataKey`
			`GenerateDataKeyPair`
			`GenerateDataKeyPairWithoutPlaintext`
			`GenerateDataKeyWithoutPlaintext`
			`ReEncrypt`
			`RetireGrant`
				Schlüsselrichtlinien und IAM-Richtlinien

Mit dem `kms:EncryptionContext:context-key`-Bedingungsschlüssel können Sie den Zugriff auf einen KMS-Schlüssel zu symmetrischen Verschlüsselung anhand des Verschlüsselungskontexts in einer Anforderung für eine kryptografische Produktion steuern. Verwenden Sie diesen Bedingungsschlüssel, um sowohl den Schlüssel als auch den Wert im Verschlüsselungskontext-Paar auszuwerten. Verwenden Sie den EncryptionContextKeys Bedingungsschlüssel kms:, um nur die Schlüssel für den Verschlüsselungskontext auszuwerten oder einen Verschlüsselungskontext unabhängig von Schlüsseln oder Werten zu benötigen.

> **Note**
>
> Bedingungsschlüsselwerte müssen die Zeichenregeln für Schlüsselrichtlinien und IAM-Richtlinien einhalten. Einige Zeichen, die in einem Verschlüsselungskontext gültig sind, sind in Richtlinien nicht gültig. Sie können diesen Bedingungsschlüssel möglicherweise nicht verwenden, um alle gültigen Verschlüsselungskontextwerte auszudrücken. Weitere Informationen zu wichtigen Dokumentenregeln für Schlüsselrichtlinien finden Sie unter Schlüsselrichtlinienformat. Weitere Informationen zu Regeln für IAM-Richtliniendokumente finden Sie unter Anforderungen für den IAM-Namen im IAM-Benutzerhandbuch.

Sie können mit einem asymmetrische KMS-Schlüssel oder einem HMAC-KMS-Schlüssel keinen Verschlüsselungskontext in einer kryptografischen Produktion angeben. Asymmetrische Algorithmen und MAC-Algorithmen unterstützen keinen Verschlüsselungskontext.

Um den Bedingungsschlüssel kms:EncryptionContext: context-key zu verwenden, ersetzen Sie den *`context-key`* Platzhalter durch den Verschlüsselungskontextschlüssel. Ersetzen Sie den *`context-value`*-Platzhalter durch den Kontextwert der Verschlüsselung.

```
"kms:EncryptionContext:context-key": "context-value"
```

Der folgende Bedingungsschlüssel gibt beispielsweise einen Verschlüsselungskontext an, in dem der Schlüssel `AppName` und der Wert `ExampleApp` (`AppName = ExampleApp`) ist.

```
"kms:EncryptionContext:AppName": "ExampleApp"
```

Dies ist ein einzelwertiger Bedingungsschlüssel. Der Schlüssel im Bedingungsschlüssel gibt einen bestimmten Verschlüsselungskontext-Schlüssel an (Kontext-Schlüssel). Obwohl Sie mehrere Verschlüsselungskontext-Paare in jede API-Anforderung einschließen können, kann

das Verschlüsselungskontext-Paar mit dem angegebenen Kontext-Schlüssel nur einen Wert haben. Der `kms:EncryptionContext:Department`-Bedingungsschlüssel gilt beispielsweise nur für Verschlüsselungskontext-Paare mit einem `Department`-Schlüssel, und jedes gegebene Verschlüsselungskontext-Paar mit dem `Department`-Schlüssel kann nur einen Wert haben.

Verwenden Sie keinen Satz-Operator mit dem `kms:EncryptionContext:`*`context-key`*-Bedingungsschlüssel. Wenn Sie eine Richtlinienanweisung mit einer `Allow`-Aktion, dem `kms:EncryptionContext:`*`context-key`*-Bedingungsschlüssel und dem `ForAllValues`-Satz-Operator erstellen, erlaubt die Bedingung Anforderungen ohne Verschlüsselungskontext und Anforderungen mit Verschlüsselungskontext-Paaren, die nicht in der Richtlinienbedingung angegeben sind.

> **Warning**
>
> Verwenden Sie keinen `ForAnyValue`- oder `ForAllValues`-Satz-Operator mit diesem einzelwertigen Bedingungsschlüssel. Diese Satz-Operatoren können eine Richtlinienbedingung erstellen, die keine der von Ihnen gewünschten Werte erfordert und Werte erlaubt, die Sie verbieten möchten.
> Wenn Sie eine Richtlinie erstellen oder aktualisieren, die einen `ForAllValues` Set-Operator mit dem Kontextschlüssel kms:EncryptionContext: enthält, wird die folgende Fehlermeldung AWS KMS zurückgegeben:
> `OverlyPermissiveCondition:EncryptionContext: Using the ForAllValues set operator with a single-valued condition key matches requests without the specified encryption context or with an unspecified encryption context. To fix, remove ForAllValues.`

Um ein bestimmtes Verschlüsselungskontext-Paar zu erfordern, verwenden Sie den `kms:EncryptionContext:`*`context-key`*-Bedingungsschlüssel mit dem `StringEquals`-Operator.

Die folgende Beispiel-Schlüsselrichtlinienanweisung erlaubt es Prinzipalen, die die Rolle übernehmen können, den KMS-Schlüssel in einer `GenerateDataKey`-Anforderung zu verwenden, nur dann, wenn der Verschlüsselungskontext in der Anforderung das `AppName:ExampleApp`-Paar beinhält. Andere Verschlüsselungskontext-Paare sind zulässig.

Für den Schlüsselnamen muss die Groß-/Kleinschreibung nicht berücksichtigt werden. Die Berücksichtigung der Groß-/Kleinschreibung des Wertes wird durch den Bedingungsoperator (z. B.

`StringEquals`) festgelegt. Details hierzu finden Sie unter Beachtung der Groß-/Kleinschreibung bei der Verschlüsselungskontextbedingung.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
}
```

Um ein Verschlüsselungskontextpaar vorzuschreiben und alle anderen Verschlüsselungskontextpaare zu verbieten, verwenden Sie sowohl kms:EncryptionContext: context-key als auch in der Richtlinienanweisung. `kms:EncryptionContextKeys` Die folgende Schlüsselrichtlinienanweisung verwendet die `kms:EncryptionContext:`AppName-Bedingung, um das AppName=`ExampleApp`-Verschlüsselungskontext-Paar in der Anforderung zu erfordern. Sie verwendet auch einen `kms:EncryptionContextKeys`-Bedingungsschlüssel mit dem `ForAllValues`-Satz-Operator, um nur den AppName-Verschlüsselungskontext-Schlüssel zu erlauben.

Der `ForAllValues`-Satz-Operator beschränkt Verschlüsselungskontext-Schlüssel in der Anforderung an AppName. Wenn die `kms:EncryptionContextKeys`-Bedingung mit dem `ForAllValues`-Satz-Operator alleine in einer Richtlinienanweisung verwendet würde, würde dieser Satz-Operator Anforderungen ohne Verschlüsselungskontext erlauben. Wenn die Anforderung jedoch keinen Verschlüsselungskontext hätte, würde die `kms:EncryptionContext:`AppName-Bedingung fehlschlagen. Ausführliche Informationen zu dem `ForAllValues`-Satz-Operator finden Sie unter Verwenden mehrerer Schlüssel und Werte im IAM-Benutzerhandbuch.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/KeyUsers"
  },
  "Action": "kms:GenerateDataKey",
```

```
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    },
    "ForAllValues:StringEquals": {
      "kms:EncryptionContextKeys": [
        "AppName"
      ]
    }
  }
}
```

Sie können diesen Bedingungsschlüssel auch verwenden, um den Zugriff auf einen KMS-Schlüssel für eine bestimmte Produktion zu verweigern. Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet eine `Deny`-Wirkung, um dem Prinzipal die Verwendung des KMS-Schlüssels zu verbieten, wenn der Verschlüsselungskontext in der Anforderung ein `Stage=Restricted`-Verschlüsselungskontext-Paar enthält. Diese Bedingung ermöglicht eine Anforderung mit anderen Verschlüsselungskontext-Paaren, einschließlich Verschlüsselungskontext-Paaren mit dem `Stage`-Schlüssel und andere Werte, wie `Stage=Test`.

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Stage": "Restricted"
    }
  }
}
```

Verwenden mehrerer Verschlüsselungskontext-Paare

Sie können mehrere Verschlüsselungskontext-Paare erfordern oder verbieten. Sie können auch eines von mehreren Verschlüsselungskontext-Paaren erfordern. Ausführliche Informationen zur Logik, die zum Interpretieren dieser Bedingungen verwendet wird, finden Sie unter Erstellen einer Bedingung mit mehreren Schlüsseln oder Werten im IAM-Benutzerhandbuch.

> **Note**
>
> In früheren Versionen dieses Themas wurden Richtlinienanweisungen angezeigt, in denen die Operatoren `ForAnyValue` und `ForAllValues` Set mit dem Bedingungsschlüssel kms:EncryptionContext: context-key verwendet wurden. Verwenden eines Set-Operators mit einem einzelwertigen Bedingungsschlüssel kann zu Richtlinien führen, die Anforderungen ohne Verschlüsselungskontext und mit nicht-spezifizierten Verschlüsselungskontext-Paaren erlauben.
> Eine Richtlinienbedingung mit dem `Allow`-Effekt, dem `ForAllValues`-Satz-Operator und dem `"kms:EncryptionContext:Department": "IT"`-Bedingungsschlüssel, beschränkt den Verschlüsselungskontext nicht auf das Paar „Department=IT“. Sie erlaubt Anforderungen ohne Verschlüsselungskontext und Anforderungen mit nicht spezifizierten Verschlüsselungskontext-Paaren, wie `Stage=Restricted`.
> Bitte überprüfen Sie Ihre Richtlinien und entfernen Sie den Operator set aus allen Bedingungen mit kms:EncryptionContext: context-key. Versuche, eine Richtlinie mit diesem Format zu erstellen oder zu aktualisieren, schlagen mit einer `OverlyPermissiveCondition`-Ausnahme fehl. Um den Fehler zu beheben, löschen Sie den Satz-Operator.

Um mehrere Verschlüsselungskontext-Paare zu erfordern, listen Sie die Paare in derselben Bedingung auf. Die folgende Beispiel-Schlüsselrichtlinienanweisung erfordert zwei Verschlüsselungskontext-Paare, `Department=IT` und `Project=Alpha`. Da die Bedingungen unterschiedliche Schlüssel haben (`kms:EncryptionContext:Department` und `kms:EncryptionContext:Project`), werden sie implizit durch einen AND-Operator verbunden. Andere Verschlüsselungskontext-Paare sind zulässig, aber nicht erforderlich.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Department": "IT",
      "kms:EncryptionContext:Project": "Alpha"
    }
```

```
  }
}
```

Um ein Verschlüsselungskontext-Paar OR ein anderes Paar zu erfordern, platzieren Sie jeden Bedingungsschlüssel in einer separaten Richtlinienanweisung. Das folgende Beispiel einer Schlüsselrichtlinie erfordert `Department=IT`- oder `Project=Alpha`-Paare oder beides. Andere Verschlüsselungskontext-Paare sind zulässig, aber nicht erforderlich.

```
{
 "Effect": "Allow",
 "Principal": {
  "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
 },
 "Action": "kms:GenerateDataKey",
 "Resource": "*",
 "Condition": {
  "StringEquals": {
   "kms:EncryptionContext:Department": "IT"
  }
 }
},
{
 "Effect": "Allow",
 "Principal": {
  "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
 },
 "Action": "kms:GenerateDataKey",
 "Resource": "*",
 "Condition": {
  "StringEquals": {
   "kms:EncryptionContext:Project": "Alpha"
  }
 }
}
```

Um bestimmte Verschlüsselungspaare vorzuschreiben und alle anderen Verschlüsselungskontextpaare auszuschließen, verwenden Sie sowohl den kms: EncryptionContext Kontextschlüssel als auch `kms:EncryptionContextKeys`in der Richtlinienerklärung. In der folgenden wichtigen Richtlinienanweisung wird die Bedingung kms:EncryptionContext: context-key verwendet, um einen Verschlüsselungskontext mit beiden Und-Paaren vorzuschreiben. *`Department=IT`* `Project=Alpha` Sie verwendet einen `kms:EncryptionContextKeys`-

Bedingungsschlüssel mit dem `ForAllValues`-Satz-Operator, um nur die `Department`- und `Project`-Verschlüsselungskontext-Schlüssel zu erlauben.

Der `ForAllValues`-Satz-Operator beschränkt Verschlüsselungskontext-Schlüssel in der Anforderung an `Department` und `Project`. Wenn er allein in einer Bedingung verwendet würde, würde dieser Set-Operator Anfragen ohne Verschlüsselungskontext zulassen, aber in dieser Konfiguration würde der Kontext-Schlüssel kms: EncryptionContext: in dieser Bedingung fehlschlagen.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Department": "IT",
      "kms:EncryptionContext:Project": "Alpha"
    },
    "ForAllValues:StringEquals": {
      "kms:EncryptionContextKeys": [
        "Department",
        "Project"
      ]
    }
  }
}
```

Sie können auch mehrere Verschlüsselungskontext-Paare verbieten. Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet eine `Deny`-Wirkung, um dem Prinzipal die Verwendung des KMS-Schlüssels zu verbieten, wenn der Verschlüsselungskontext in der Anforderung ein `Stage=Restricted`- oder `Stage=Production`-Paar enthält.

Mehrere Werte (`Restricted` und `Production`) für denselben Schlüssel (`kms:EncryptionContext:Stage`) sind implizit durch ein OR verbunden. Details dazu finden Sie unter Auswertungslogik für Bedingungen mit mehreren Schlüsseln oder Werten im IAM-Benutzerhandbuch.

```
{
```

```
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Stage": [
         "Restricted",
         "Production"
      ]
    }
  }
}
```

Beachtung der Groß-/Kleinschreibung bei der Verschlüsselungskontextbedingung

Der Verschlüsselungskontext, der in einem Entschlüsselungsvorgang angegeben wird, muss exakt mit dem Verschlüsselungskontext übereinstimmen, der in dem Verschlüsselungsvorgang angegeben wird (inklusive Groß-/Kleinschreibung). Nur die Reihenfolge, in der die Paare angegeben werden, spielt keine Rolle.

In den Richtlinienbedingungen wird die Groß-/Kleinschreibung für den Bedingungsschlüssel nicht berücksichtigt. Die Berücksichtigung der Groß-/Kleinschreibung des Bedingungswertes wird durch den von Ihnen verwendeten Richtlinienbedingungsoperator (z. B. `StringEquals` oder `StringEqualsIgnoreCase`) festgelegt.

Daher unterscheidet der Bedingungsschlüssel, der aus dem `kms:EncryptionContext:`-Präfix und dem *`context-key`*-Ersatz besteht, nicht zwischen Groß-/Kleinschreibung. Eine Richtlinie, die diese Bedingung verwendet, berücksichtigt die Groß-/Kleinschreibung bei beiden Elemente des Bedingungsschlüssels nicht. Die Berücksichtigung der Groß-/Kleinschreibung des Wertes (d. h. der *`context-value`*-Ersatz) wird durch den Richtlinienbedingungsoperator festgelegt.

Die folgende Richtlinie erlaubt den Vorgang, wenn der Verschlüsselungskontext einen `Appname`-Schlüssel enthält – unabhängig von der Groß-/Kleinschreibung. Die `StringEquals`-Bedingung erfordert, dass `ExampleApp` wie ursprünglich angegeben geschrieben wird.

```
{
  "Effect": "Allow",
  "Principal": {
```

```
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Appname": "ExampleApp"
    }
  }
}
```

Um einen Verschlüsselungskontextschlüssel zu verlangen, bei dem die Groß- und Kleinschreibung beachtet werden muss, verwenden Sie die EncryptionContextKeys Richtlinienbedingung kms: mit einem Bedingungsoperator, bei dem Groß- und Kleinschreibung beachtet werden muss, z. B. `StringEquals` Da der Verschlüsselungskontext-Schlüssel in dieser Richtlinienbedingung der Wert der Richtlinienbedingung ist, wird die Groß-/Kleinschreibung durch den Bedingungsoperator berücksichtigt.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    }
  }
}
```

Um eine Bewertung sowohl des Schlüssels als auch des Werts für den Verschlüsselungskontext unter Berücksichtigung der Groß- und Kleinschreibung zu verlangen, verwenden Sie die Richtlinienbedingungen `kms:EncryptionContextKeys` und kms:EncryptionContext: für den Kontextschlüssel zusammen in derselben Richtlinienanweisung. Der Bedingungsoperator (z. B.`StringEquals`) gilt immer für den Wert der Bedingung. Der Verschlüsselungskontext-Schlüssel (z. B.AppName) ist der Wert der `kms:EncryptionContextKeys`-Bedingung. Der Wert für den Verschlüsselungskontext (z. B.*`ExampleApp`*) ist der Wert der Bedingung kms:EncryptionContext: context-key.

In der folgenden Beispiel-Schlüsselrichtlinienanweisung wird, nachdem der `StringEquals`-Operator die Groß-/Kleinschreibung berücksichtigt, sowohl im Schlüssel als auch im Wert des Verschlüsselungskontextes die Groß-/Kleinschreibung berücksichtigt.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    },
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
}
```

Verwenden von Variablen in einer Verschlüsselungskontextbedingung

Der Schlüssel und der Wert in einem Verschlüsselungskontextpaar müssen einfache Literalzeichenfolgen sein. Sie dürfen keine Ganzzahlen oder Objekte oder Typen, die nicht vollständig aufgelöst sind, sein. Wenn Sie einen anderen Typ verwenden, z. B. eine Ganzzahl oder eine Fließkommazahl, AWS KMS interpretiert dies als Literalzeichenfolge.

```
"encryptionContext": {
    "department": "10103.0"
}
```

Der Wert im `kms:EncryptionContext:`*`context-key`*-Bedingungsschlüsselpaar kann jedoch eine IAM-Richtlinien-Variable sein. Diese Richtlinien-Variablen werden zur Laufzeit basierend auf Werten in der Anforderung aufgelöst. Beispielsweise wird `aws:CurrentTime` zum Zeitpunkt der Anforderung und `aws:username` zum Anzeigenamen des Aufrufers aufgelöst.

Sie können diese Richtlinien-Variablen verwenden, um eine Richtlinienanweisung mit einer Bedingung zu erstellen, die sehr spezifische Informationen in einem Verschlüsselungskontext erfordert, z. B. den Benutzernamen des Aufrufers. Da sie eine Variable enthält, können Sie dieselbe

Richtlinienanweisung für alle Benutzer verwenden, die die Rolle übernehmen können. Sie müssen nicht für jeden Benutzer eine separate Richtlinienanweisung schreiben.

Angenommen, Sie möchten, dass alle Benutzer, die eine Rolle annehmen können, denselben KMS-Schlüssel verwenden, um ihre Daten zu verschlüsseln und zu entschlüsseln. Sie möchten ihnen jedoch nur erlauben, die Daten zu entschlüsseln, die sie verschlüsselt haben. Stellen Sie zunächst fest, dass jede Anfrage einen Verschlüsselungskontext AWS KMS enthalten muss, in dem sich der Schlüssel befindet `user` und der Wert dem AWS Benutzernamen des Aufrufers entspricht, z. B. der folgende.

```
"encryptionContext": {
    "user": "bob"
}
```

Um diese Anforderung zu erzwingen, können Sie dann eine Richtlinienanweisung wie die im folgenden Beispiel verwenden. Diese Richtlinienanweisung erteilt der `TestTeam`-Rolle die Berechtigung zum Verschlüsseln und Entschlüsseln von Daten mit dem KMS-Schlüssel. Die Berechtigung ist jedoch nur gültig, wenn der Verschlüsselungskontext in der Anforderung ein `"user": "<username>"`-Paar enthält. Um den Benutzernamen darzustellen, verwendet die Bedingung die `aws:username`-Richtlinien-Variable.

Wenn die Anforderung ausgewertet wird, ersetzt der Benutzername des Aufrufers die Variable in der Bedingung. So erfordert die Bedingung einen Verschlüsselungskontext von `"user": "bob"` für "bob" und `"user": "alice"` für "alice".

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/TestTeam"
  },
  "Action": [
    "kms:Decrypt",
    "kms:Encrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:user": "${aws:username}"
    }
  }
```

```
}
```

Sie können eine IAM-Richtlinien-Variable nur im Wert des Bedingungsschlüssels `kms:EncryptionContext:context-key` verwenden. Sie können keine Variable im Schlüssel verwenden.

Sie können auch anbieterspezifische Kontextschlüssel in Variablen verwenden. Diese Kontextschlüssel identifizieren eindeutig Benutzer, die sich AWS mithilfe des Web Identity Federation angemeldet haben.

Wie alle Variablen können diese Variablen nur in der `kms:EncryptionContext:context-key`-Richtlinienbedingung und nicht im eigentlichen Verschlüsselungskontext verwendet werden. Und sie können nur im Wert der Bedingung verwendet werden, nicht im Schlüssel.

Die folgende Schlüsselrichtlinienanweisung ähnelt beispielsweise der vorherigen. Die Bedingung erfordert jedoch einen Verschlüsselungskontext, in dem der Schlüssel `sub` ist und der Wert einen Benutzer eindeutig identifiziert, der in einem Amazon-Cognito-Benutzerpool angemeldet ist. Weitere Informationen zum Identifizieren von Benutzern und Rollen in Amazon Cognito finden Sie unter IAM-Rollen im Amazon-Cognito-Entwicklerhandbuch.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/TestTeam"
  },
  "Action": [
    "kms:Decrypt",
    "kms:Encrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
       "kms:EncryptionContext:sub": "${cognito-identity.amazonaws.com:sub}"
    }
  }
}
```

Informationen finden Sie auch unter:

- the section called “km: EncryptionContextKeys”
- the section called “km: GrantConstraintType”

km: EncryptionContextKeys

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp

`kms:EncryptionContextKeys`	Zeichenfolge (Liste)	Mehrwertig	`CreateGrant`
			`Decrypt`
			`Encrypt`
			`GenerateDataKey`
			`GenerateDataKeyPair`
			`GenerateDataKeyPairWithoutPlaintext`
			`GenerateDataKeyWithoutPlaintext`
			`ReEncrypt`
			`RetireGrant`
				Schlüsselrichtlinien und IAM-Richtlinien

Mit dem Bedingungsschlüssel `kms:EncryptionContextKeys` können Sie den Zugriff auf einen KMS-Schlüssel zu symmetrischen Verschlüsselung anhand des Verschlüsselungskontexts in einer Anforderung für eine kryptografische Produktion steuern. Verwenden Sie diesen Bedingungsschlüssel, um nur den Wert in den einzelnen Verschlüsselungskontext-Paaren auszuwerten. Verwenden Sie den Bedingungsschlüssel `kms:EncryptionContext:`*`context-key`*, um sowohl den Schlüssel als auch den Wert im Verschlüsselungskontext auszuwerten.

Sie können mit einem asymmetrische KMS-Schlüssel oder einem HMAC-KMS-Schlüssel keinen Verschlüsselungskontext in einer kryptografischen Produktion angeben. Asymmetrische Algorithmen und MAC-Algorithmen unterstützen keinen Verschlüsselungskontext.

> **Note**
>
> Bedingungsschlüsselwerte, einschließlich eines Verschlüsselungskontextschlüssels, müssen den Zeichen- und Kodierungsregeln für AWS KMS Schlüsselrichtlinien entsprechen. Sie können diesen Bedingungsschlüssel möglicherweise nicht verwenden, um alle gültigen Verschlüsselungskontextschlüssel auszudrücken. Weitere Informationen zu wichtigen Dokumentenregeln für Schlüsselrichtlinien finden Sie unter Schlüsselrichtlinienformat. Weitere Informationen zu Regeln für IAM-Richtliniendokumente finden Sie unter Anforderungen für den IAM-Namen im IAM-Benutzerhandbuch.

Dies ist ein mehrwertiger Bedingungsschlüssel. Sie können mehrere Verschlüsselungskontext-Paare in jeder API-Anforderung angeben. `kms:EncryptionContextKeys` vergleicht die Verschlüsselungskontext-Schlüssel in der Anforderung mit dem Satz von Verschlüsselungskontext-Schlüsseln in der Richtlinie. Um zu bestimmen, wie diese Sätze verglichen werden, müssen Sie einen `ForAnyValue` oder `ForAllValues`-Satz-Operator in der Richtlinienbedingung angeben. Ausführliche Informationen zu den Satz-Operatoren finden Sie unter Verwenden mehrerer Schlüssel und Werte im IAM-Benutzerhandbuch.

- `ForAnyValue`: Mindestens ein Verschlüsselungskontext-Schlüssel in der Anforderung muss mit einem Verschlüsselungskontext-Schlüssel in der Richtlinienbedingung übereinstimmen. Andere Verschlüsselungskontext-Schlüssel sind zulässig. Wenn die Anforderung keinen Verschlüsselungskontext aufweist, ist die Bedingung nicht erfüllt.
- `ForAllValues`: Mindestens ein Verschlüsselungskontext-Schlüssel in der Anforderung muss mit einem Verschlüsselungskontext-Schlüssel in der Richtlinienbedingung übereinstimmen. Dieser Satz-Operator beschränkt die Verschlüsselungskontext-Schlüssel auf diejenigen in der Richtlinienbedingung. Er erfordert keine Verschlüsselungskontext-Schlüssel, aber er verbietet nicht spezifizierte Verschlüsselungskontext-Schlüssel.

Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet den `kms:EncryptionContextKeys`-Bedingungsschlüssel mit dem `ForAnyValue`-Operator. Diese Richtlinienanweisung erlaubt die Verwendung eines KMS-Schlüssels für die angegebenen

Operationen nur, wenn mindestens eines der Verschlüsselungskontext-Paare in der Anforderung den `AppName`-Schlüssel enthält, unabhängig von dessen Wert.

Diese Schlüsselrichtlinienanweisung erlaubt beispielsweise eine `GenerateDataKey`-Anforderung mit zwei Verschlüsselungskontext-Paaren, `AppName=Helper` und `Project=Alpha`, da das erste Verschlüsselungskontext-Paar die Bedingung erfüllt. Eine Anforderung mit nur `Project=Alpha` oder ohne Verschlüsselungskontext würde fehlschlagen.

Da bei der StringEqualsBedingungsoperation Groß- und Kleinschreibung beachtet wird, erfordert diese Richtlinienanweisung die Schreibweise und Groß- und Kleinschreibung des Verschlüsselungskontextschlüssels. Sie können aber einen Bedingungsoperator verwenden, der die Groß-/Kleinschreibung des Schlüssels ignoriert – z. B. `StringEqualsIgnoreCase`.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": [
    "kms:Encrypt",
    "kms:GenerateDataKey*"
  ],
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    }
  }
}
```

Sie können den `kms:EncryptionContextKeys`-Bedingungsschlüssel auch verwenden, um einen (beliebigen) Verschlüsselungskontext in kryptografischen Operationen, die den KMS-Schlüssel verwenden, zu erfordern.

Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet den Bedingungsschlüssel `kms:EncryptionContextKeys` mit dem Null-bedingten Operator, um den Zugriff auf einen KMS-Schlüssel nur zu erlauben, wenn der Verschlüsselungskontext in der API-Anforderung nicht Null ist. Diese Bedingung überprüft nicht die Schlüssel oder Werte des der Verschlüsselungskontexts. Sie überprüft nur, ob der Verschlüsselungskontext vorhanden ist.

```
{
```

```
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": [
    "kms:Encrypt",
    "kms:GenerateDataKey*"
  ],
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:EncryptionContextKeys": false
    }
  }
}
```

Informationen finden Sie auch unter:

- kms:EncryptionContext: Kontextschlüssel
- km: GrantConstraintType

km: ExpirationModel

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:ExpirationModel`	String	Einzelwertig	`ImportKeyMaterial`	Schlüsselrichtlinien und IAM-Richtlinien

Der `kms:ExpirationModel` Bedingungsschlüssel steuert den Zugriff auf die ImportKeyMaterialOperation auf der Grundlage des Werts des ExpirationModelParameters in der Anforderung.

`ExpirationModel` ist ein optionaler Parameter, der festlegt, ob das importierte Schlüsselmaterial abgelaufen ist. Gültige Werte sind `KEY_MATERIAL_EXPIRES` und `KEY_MATERIAL_DOES_NOT_EXPIRE`. Der Standardwert ist `KEY_MATERIAL_EXPIRES`.

Das Ablaufdatum und die Uhrzeit werden durch den Wert des ValidToParameters bestimmt. Der `ValidTo`-Parameter ist erforderlich, es sei denn der Wert des `ExpirationModel`-Parameters lautet `KEY_MATERIAL_DOES_NOT_EXPIRE`. Sie können auch den ValidTo Bedingungsschlüssel kms: verwenden, um ein bestimmtes Ablaufdatum als Bedingung für den Zugriff festzulegen.

Das folgende Richtlinienanweisungsbeispiel verwendet den `kms:ExpirationModel`-Bedingungsschlüssel, um Benutzern das Importieren von Schlüsselmaterial in einen KMS-Schlüssel nur dann zu erlauben, wenn die Anforderung den `ExpirationModel`-Parameter enthält und der entsprechende Wert `KEY_MATERIAL_DOES_NOT_EXPIRE` lautet.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:ImportKeyMaterial",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ExpirationModel": "KEY_MATERIAL_DOES_NOT_EXPIRE"
    }
  }
}
```

Sie können auch den `kms:ExpirationModel`-Bedingungsschlüssel verwenden, um Benutzern das Importieren von Schlüsselmaterial nur dann zu erlauben, wenn das Schlüsselmaterial abgelaufen ist, ohne ein Ablaufdatum in der Bedingung anzugeben. Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet den `kms:ExpirationModel`-Bedingungsschlüssel mit dem Null-bedingten Operator, um Benutzern das Importieren von Schlüsselmaterial nur dann zu erlauben, wenn die Anforderung keinen `ExpirationModel`-Parameter enthält. Der Standardwert für ExpirationModel ist`KEY_MATERIAL_EXPIRES`.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:ImportKeyMaterial",
  "Resource": "*",
  "Condition": {
    "Null": {
```

```
      "kms:ExpirationModel": true
    }
  }
}
```

Informationen finden Sie auch unter:

- km: ValidTo
- km: WrappingAlgorithm
- km: WrappingKeySpec

km: GrantConstraintType

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:GrantConstraintType`	String	Einzelwertig	`CreateGrant` `RetireGrant`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf den CreateGrantVorgang auf der Grundlage der Art der Gewährungsbeschränkung in der Anfrage zu steuern.

Wenn Sie eine Erteilung erstellen, können Sie optional eine Erteilungseinschränkung festlegen, damit die Operationen nur dann Zugriff gewähren, wenn ein bestimmter Verschlüsselungskontext vorhanden ist. Die Erteilungseinschränkung kann einem der folgenden beiden Typen vorliegen: `EncryptionContextEquals` oder `EncryptionContextSubset`. Sie können diesen Bedingungsschlüssel verwenden, um zu überprüfen, ob die Anforderung den einen oder den anderen Typen enthält.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet den Bedingungsschlüssel `kms:GrantConstraintType`, um Benutzern nur dann das Erstellen von Erteilungen zu gestatten, wenn die Anforderung die Erteilungseinschränkung `EncryptionContextEquals` enthält. Das Beispiel zeigt eine Richtlinienanweisung in einer Schlüsselrichtlinie.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:GrantConstraintType": "EncryptionContextEquals"
    }
  }
}
```

Informationen finden Sie auch unter:

- kms:EncryptionContext: Kontextschlüssel
- km: EncryptionContextKeys
- km: GrantIsFor AWSResource
- km: GrantOperations
- km: GranteePrincipal
- km: RetiringPrincipal

km: GrantIsFor AWSResource

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:GrantIsForAWSResource`	Boolesch	Einzelwertig	`CreateGrant` `ListGrants`	Schlüsselrichtlinien und IAM-Richtlinien

AWS KMS Zustandss chlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
			`RevokeGrant`	

Erlaubt oder verweigert die Erlaubnis für die RevokeGrantOperationen CreateGrantListGrants, oder nur, wenn ein in integrierter AWS Dienst den Vorgang im Namen des Benutzers AWS KMS aufruft. Diese Richtlinienbedingung erlaubt es dem Benutzer nicht, diese Erteilungs-Operationen direkt aufzurufen.

Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet den Bedingungsschlüssel `kms:GrantIsForAWSResource`. Es ermöglicht integrierten AWS Diensten wie Amazon EBS AWS KMS, im Namen des angegebenen Prinzipals Grants für diesen KMS-Schlüssel zu gewähren.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:GrantIsForAWSResource": true
    }
  }
}
```

Informationen finden Sie auch unter:

- km: GrantConstraintType
- km: GrantOperations
- km: GranteePrincipal
- km: RetiringPrincipal

km: GrantOperations

AWS KMS Zustandss chlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:GrantOperations`	String	Mehrwertig	`CreateGrant`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf den CreateGrantVorgang auf der Grundlage der in der Anfrage enthaltenen Zuschussoperationen zu steuern. Sie können beispielsweise Benutzer berechtigen, Erteilungen zu erstellen, mit denen die Berechtigung zum Verschlüsseln, aber nicht zum Entschlüsseln gewährt wird. Weitere Informationen zu Erteilungen finden Sie unter Verwenden von Erteilungen.

Dies ist ein mehrwertiger Bedingungsschlüssel. `kms:GrantOperations` vergleicht den Satz von Erteilungs-Operationen in der `CreateGrant`-Anforderung an den Satz der Erteilungs-Operationen in der Richtlinie. Um zu bestimmen, wie diese Sätze verglichen werden, müssen Sie einen `ForAnyValue` oder `ForAllValues`-Satz-Operator in der Richtlinienbedingung angeben. Ausführliche Informationen zu den Satz-Operatoren finden Sie unter Verwenden mehrerer Schlüssel und Werte im IAM-Benutzerhandbuch.

- `ForAnyValue`: Mindestens eine Erteilungs-Operation in der Anforderung muss mit einem der Erteilungs-Operationen in der Richtlinienbedingung übereinstimmen. Andere Erteilungs-Operationen sind zulässig.
- ForAllValues: Jeder Grant-Vorgang in der Anfrage muss mit einem Grant-Vorgang in der Richtlinienbedingung übereinstimmen. Dieser Satz-Operator beschränkt die Erteilungs-Operationen auf die in der Richtlinienbedingung angegebenen Operationen. Er erfordert keine Erteilungs-Operationen, aber er verbietet nicht-spezifizierte Erteilungs-Operationen.

  ForAllValues gibt auch „true“ zurück, wenn die Anfrage keine Zuschussvorgänge enthält, lässt sie aber `CreateGrant` nicht zu. Wenn das `Operations`-Symbol fehlt oder einen Nullwert hat, schlägt die `CreateGrant`-Anforderung fehl.

Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet den Bedingungsschlüssel `kms:GrantOperations`, um nur dann das Erstellen von Erteilungen zu erlauben, wenn die Erteilungs-Operationen `Encrypt`, `ReEncryptTo` oder beides sind. Wenn die Erteilung andere Operationen umfasst, schlägt die `CreateGrant`-Anforderung fehl.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "ForAllValues:StringEquals": {
      "kms:GrantOperations": [
        "Encrypt",
        "ReEncryptTo"
      ]
    }
  }
}
```

Wenn Sie den Satz-Operator in der Richtlinienbedingung auf `ForAnyValue` ändern, würde die Richtlinienanweisung erfordern, dass mindestens eine der Erteilungs-Operationen in der Erteilung `Encrypt` oder `ReEncryptTo` ist, aber es würde andere Erteilungs-Operationen wie `Decrypt` oder `ReEncryptFrom` erlauben.

Informationen finden Sie auch unter:

- km: GrantConstraintType
- km: GrantIsFor AWSResource
- km: GranteePrincipal
- km: RetiringPrincipal

km: GranteePrincipal

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:GranteePrincipal`	String	Einzelwertig	`CreateGrant`	IAM- und Schlüsselrichtlinien

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf die CreateGrantOperation anhand des GranteePrincipalParameterwerts in der Anforderung zu steuern. Sie können beispielsweise das Erstellen von Erteilungen zur Verwendung eines KMS-Schlüssels nur erlauben, wenn der erteilungsempfangende Prinzipal in der `CreateGrant`-Anforderung dem in der Bedingungsanweisung angegebenen Prinzipal entspricht.

Um den Principal des Empfängers anzugeben, verwenden Sie den Amazon-Ressourcennamen (ARN) eines AWS Prinzipals. Zu den gültigen Prinzipalen gehören AWS-Konten IAM-Benutzer, IAM-Rollen, Verbundbenutzer und Benutzer mit angenommenen Rollen. Hilfe zur ARN-Syntax für einen Prinzipal finden Sie unter IAM ARNs im IAM-Benutzerhandbuch.

Die folgende Beispiel-Schlüsselrichtlinienanweisung verwendet den Bedingungsschlüssel `kms:GranteePrincipal`, um nur dann das Erstellen von Erteilungen für einen KMS-Schlüssel zu erlauben, wenn der erteilungsempfangende Prinzipal in der Erteilung die `LimitedAdminRole` ist.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:GranteePrincipal": "arn:aws:iam::111122223333:role/LimitedAdminRole"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: GrantConstraintType
- km: GrantIsFor AWSResource
- km: GrantOperations
- km: RetiringPrincipal

km: KeyAgreementAlgorithm

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:KeyAgreementAlgorithm`	String	Einzelwertig	`DeriveSharedSecret`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können den `kms:KeyAgreementAlgorithm` Bedingungsschlüssel verwenden, um den Zugriff auf die DeriveSharedSecretOperation basierend auf dem Wert des `KeyAgreementAlgorithm` Parameters in der Anforderung zu steuern. Der einzig gültige Wert für `KeyAgreementAlgorithm` ist`ECDH`.

In der folgenden wichtigen Richtlinienanweisung wird beispielsweise der `kms:KeyAgreementAlgorithm` Bedingungsschlüssel verwendet, um jeglichen Zugriff zu verweigern, DeriveSharedSecret sofern dies nicht der `KeyAgreementAlgorithm` Fall ist`ECDH`.

```
{
       "Effect": "Deny",
       "Principal": {
         "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
       },
       "Action": "kms:DeriveSharedSecret",
       "Resource": "*",
       "Condition": {
            "StringNotEquals": {
               "kms:KeyAgreementAlgorithm": "ECDH"
```

```
            }
        }
}
```

Informationen finden Sie auch unter:

- the section called “km: KeyUsage”

km: KeyOrigin

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:KeyOrigin`	String	Einzelwertig	`CreateKey` KMS-Schlüsselressourcen-Operationen	IAM-Richtlinien Schlüsselrichtlinien und IAM-Richtlinien

Der Bedingungsschlüssel `kms:KeyOrigin` steuert den Zugriff auf Vorgänge anhand des Wertes der `Origin`-Eigenschaft des KMS-Schlüssels, der von der Produktion erstellt oder verwendet wird. Er funktioniert als Ressourcenbedingung oder als Anforderungsbedingung.

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf die CreateKeyOperation auf der Grundlage des Werts des Origin-Parameters in der Anfrage zu steuern. Gültige Werte für `Origin` sind `AWS_KMS`, `AWS_CLOUDHSM` und `EXTERNAL`.

Sie können beispielsweise einen KMS-Schlüssel nur erstellen, wenn das Schlüsselmaterial in AWS KMS (`AWS_KMS`) generiert wird, nur wenn das Schlüsselmaterial in einem AWS CloudHSM Cluster generiert wird, der einem benutzerdefinierten Schlüsselspeicher (`AWS_CLOUDHSM`) zugeordnet ist, oder nur, wenn das Schlüsselmaterial aus einer externen Quelle importiert wird (`EXTERNAL`).

In der folgenden Beispielanweisung für eine Schlüsselrichtlinie wird der `kms:KeyOrigin` Bedingungsschlüssel nur dann verwendet, um einen KMS-Schlüssel zu erstellen, wenn das Schlüsselmaterial AWS KMS erstellt wird.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
      },
      "Action": "kms:CreateKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:KeyOrigin": "AWS_KMS"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:GenerateDataKeyPair",
        "kms:GenerateDataKeyPairWithoutPlaintext",
        "kms:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "kms:KeyOrigin": "AWS_CLOUDHSM"
        }
      }
    }
  ]
}
```

Sie können mit dem Bedingungsschlüssel `kms:KeyOrigin` auch den Zugriff auf Operationen steuern, die einen KMS-Schlüssel verwenden oder verwalten, basierend auf der `Origin`-Eigenschaft des KMS-Schlüssels, der für die Produktion verwendet wird. Bei der Operation muss es sich um eine KMS-Schlüsselressourcen-Operation handeln, das heißt, eine Operation, die für einen bestimmten KMS-Schlüssel autorisiert ist. Um die KMS-Schlüsselressourcen-Operationen zu identifizieren, suchen Sie in der Tabelle Actions and Resources (Aktionen und Ressourcen) Sie nach dem Wert von `KMS key` in der `Resources`-Spalte für die Operation.

Die folgende IAM-Richtlinie erlaubt es beispielsweise Prinzipalen, die angegebenen KMS-Schlüsselressourcen-Operationen auszuführen, jedoch nur mit KMS-Schlüssel in dem Konto, die in einem benutzerdefinierten Schlüsselspeicher erstellt wurde.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:GenerateDataKey",
    "kms:GenerateDataKeyWithoutPlaintext",
    "kms:GenerateDataKeyPair",
    "kms:GenerateDataKeyPairWithoutPlaintext",
    "kms:ReEncrypt*"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "AWS_CLOUDHSM"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: BypassPolicyLockoutSafetyCheck
- km: KeySpec
- km: KeyUsage

km: KeySpec

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:KeySpec`	String	Einzelwertig	`CreateKey` KMS-Schlüsselressourcen-Operationen	IAM-Richtlinien Schlüsselrichtlinien und IAM-Richtlinien

Der Bedingungsschlüssel `kms:KeySpec` steuert den Zugriff auf Operationen anhand des Wertes der `KeySpec`-Eigenschaft des KMS-Schlüssels, der von der Produktion erstellt oder verwendet wird.

Sie können diesen Bedingungsschlüssel in einer IAM-Richtlinie verwenden, um den Zugriff auf den CreateKeyVorgang anhand des KeySpecParameterwerts in einer `CreateKey` Anforderung zu steuern. Beispielsweise können Sie es mit dieser Bedingung Benutzern erlauben, nur KMS-Schlüssel für symmetrische Verschlüsselung oder nur HMAC-KMS-Schlüssel zu erstellen.

Die folgende Beispiel-IAM-Richtlinienanweisung verwendet den Bedingungsschlüssel `kms:KeySpec`, um den Prinzipalen nur dann das Erstellen eines asymmetrischen RSA-KMS-Schlüssels zu erlauben. Die Berechtigung ist nur gültig, wenn der `KeySpec` in der Anfrage mit `RSA_` beginnt.

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:KeySpec": "RSA_*"
    }
  }
}
```

Sie können mit dem Bedingungsschlüssel `kms:KeySpec` auch den Zugriff auf Operationen steuern, die einen KMS-Schlüssel verwenden oder verwalten, basierend auf der `KeySpec`-Eigenschaft des KMS-Schlüssels, der für die Produktion verwendet wird. Bei der Operation muss es sich um eine

KMS-Schlüsselressourcen-Operation handeln, das heißt, eine Operation, die für einen bestimmten KMS-Schlüssel autorisiert ist. Um die KMS-Schlüsselressourcen-Operationen zu identifizieren, suchen Sie in der Tabelle Actions and Resources (Aktionen und Ressourcen) Sie nach dem Wert von `KMS key` in der `Resources`-Spalte für die Produktion.

Die folgende IAM-Richtlinie erlaubt es beispielsweise Prinzipalen, die angegebenen KMS-Schlüsselressourcen-Operationen auszuführen, jedoch nur mit KMS-Schlüsseln zur symmetrischen Verschlüsselung im Konto.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeySpec": "SYMMETRIC_DEFAULT"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: BypassPolicyLockoutSafetyCheck
- kms: CustomerMasterKeySpec (veraltet)
- km: DataKeyPairSpec
- km: KeyOrigin
- km: KeyUsage

km: KeyUsage

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:KeyUsage`	String	Einzelwertig	`CreateKey` KMS-Schlüsselressourcen-Operationen	IAM-Richtlinien Schlüsselrichtlinien und IAM-Richtlinien

Der Bedingungsschlüssel `kms:KeyUsage` steuert den Zugriff auf Operationen anhand des Wertes der `KeyUsage`-Eigenschaft des KMS-Schlüssels, der von der Produktion erstellt oder verwendet wird.

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf die CreateKeyOperation anhand des KeyUsageParameterwerts in der Anforderung zu steuern. Gültige Werte für `KeyUsage` sind `ENCRYPT_DECRYPTSIGN_VERIFY`,`GENERATE_VERIFY_MAC`, und`KEY_AGREEMENT`.

Beispielsweise können Sie es erlauben, einen KMS-Schlüssel nur dann zu erstellen, wenn `KeyUsage` auf `ENCRYPT_DECRYPT` eingestellt ist, oder einem Benutzer die Berechtigung verweigert, wenn `KeyUsage` auf `SIGN_VERIFY` eingestellt ist.

Die folgende Beispiel-IAM-Richtlinienanweisung verwendet den Bedingungsschlüssel `kms:KeyUsage`, damit nur dann ein KMS-Schlüssel erstellt werden kann, wenn `KeyUsage` auf `ENCRYPT_DECRYPT` eingestellt ist.

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:KeyUsage": "ENCRYPT_DECRYPT"
    }
  }
}
```

Sie können mit dem Bedingungsschlüssel `kms:KeyUsage` auch den Zugriff auf Operationen steuern, die einen KMS-Schlüssel verwenden oder verwalten, basierend auf der `KeyUsage`-Eigenschaft des KMS-Schlüssels, der für die Produktion verwendet wird. Bei der Operation muss es sich um eine KMS-Schlüsselressourcen-Operation handeln, das heißt, eine Operation, die für einen bestimmten KMS-Schlüssel autorisiert ist. Um die KMS-Schlüsselressourcen-Operationen zu identifizieren, suchen Sie in der Tabelle Actions and Resources (Aktionen und Ressourcen) Sie nach dem Wert von `KMS key` in der `Resources`-Spalte für die Operation.

Die folgende IAM-Richtlinie erlaubt es beispielsweise Prinzipalen, die angegebenen KMS-Schlüsselressourcen-Operationen auszuführen, jedoch nur mit KMS-Schlüsseln im Konto, die für Signatur und Verifizierung verwendet werden.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:CreateGrant",
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:ScheduleKeyDeletion"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyUsage": "SIGN_VERIFY"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: BypassPolicyLockoutSafetyCheck
- kms: CustomerMasterKeyUsage (veraltet)
- km: KeyOrigin
- km: KeySpec

km: MacAlgorithm

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:MacAlgorithm`	String	Einzelwertig	`GenerateMac` `VerifyMac`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können den `kms:MacAlgorithm` Bedingungsschlüssel verwenden, um den Zugriff auf die VerifyMacOperationen GenerateMacund basierend auf dem Wert des `MacAlgorithm` Parameters in der Anforderung zu steuern.

Die folgende Beispiel-Schlüsselrichtlinie ermöglicht Benutzern, die zur Verwendung des HMAC-KMS-Schlüssels zum Generieren und Verifizieren von HMAC-Tags nur dann, wenn der MAC-Algorithmus in der Anforderung `HMAC_SHA_384` oder `HMAC_SHA_512` ist, die Rolle `testers` annehmen können. Diese Richtlinie verwendet zwei separate Richtlinienaussagen mit jeweils einer eigenen Bedingung. Wenn Sie mehr als einen MAC-Algorithmus in einer einzigen Bedingungsanweisung angeben, erfordert die Bedingung beide Algorithmen anstelle des einen oder anderen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/testers"
      },
      "Action": [
        "kms:GenerateMac",
        "kms:VerifyMac"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:MacAlgorithm": "HMAC_SHA_384"
```

```
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/testers"
      },
      "Action": [
        "kms:GenerateMac",
        "kms:VerifyMac"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:MacAlgorithm": "HMAC_SHA_512"
        }
      }
    }
  ]
}
```

Informationen finden Sie auch unter:

- the section called “km: EncryptionAlgorithm”
- km: SigningAlgorithm

km: MessageType

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:MessageType`	String	Einzelwertig	`Sign` `Verify`	Schlüsselrichtlinien und IAM-Richtlinien

Der Bedingungsschlüssel `kms:MessageType` steuert den Zugriff auf die Operationen Sign und Verify basierend auf dem Wert des `MessageType`-Parameters in der Anforderung. Gültige Werte für `MessageType` sind `RAW` und `DIGEST`.

Die folgende Schlüsselrichtlinienanweisung verwendet beispielsweise den Bedingungsschlüssel `kms:MessageType`, um zu erlauben, einen asymmetrischen KMS-Schlüssel zum Signieren einer Nachricht zu verwenden, jedoch keinen Nachrichten-Digest.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:Sign",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:MessageType": "RAW"
    }
  }
}
```

Informationen finden Sie auch unter:

- the section called “km: SigningAlgorithm”

km: MultiRegion

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:MultiRegion`	Boolesch	Einzelwertig	`CreateKey` KMS-Schlüsselressourcen-Operationen	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um Operationen entweder nur für einzelregionale Schlüssel oder nur für multiregionale Schlüssel zu erlauben. Der `kms:MultiRegion` Bedingungsschlüssel steuert den Zugriff auf AWS KMS Operationen mit KMS-Schlüsseln und auf den CreateKeyVorgang, der auf dem Wert der `MultiRegion` Eigenschaft des KMS-Schlüssels basiert. Gültige Werte sind `true` (multiregionaler Schlüssel) oder `false` (einzelregionaler Schlüssel). Alle KMS-Schlüssel verfügen über eine `MultiRegion`-Eigenschaft.

Das folgende Beispiel einer IAM-Richtlinienanweisung verwendet den `kms:MultiRegion`-Bedingungsschlüssel, um den Prinzipalen nur dann das Erstellen von einzelregionalen Schlüsseln zu gestatten.

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:MultiRegion": false
    }
  }
}
```

km: MultiRegionKeyType

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:MultiRegionKeyType`	String	Einzelwertig	`CreateKey` KMS-Schlüsselressourcen-Operationen	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um Operationen entweder nur für multiregionale Primärschlüssel oder nur für multiregionale Replikatschlüssel zu erlauben. Der `kms:MultiRegionKeyType` Bedingungsschlüssel steuert den Zugriff auf AWS KMS Operationen mit KMS-Schlüsseln und den CreateKeyVorgang, der auf der `MultiRegionKeyType` Eigenschaft

des KMS-Schlüssels basiert. Die gültigen Werte sind `PRIMARY` und `REPLICA`. Nur multiregionale Schlüssel verfügen über eine `MultiRegionKeyType`-Eigenschaft.

In der Regel verwenden Sie den `kms:MultiRegionKeyType`-Bedingungsschlüssel in einer IAM-Richtlinie, um den Zugriff auf mehrere KMS-Schlüssel zu kontrollieren. Da ein bestimmter multiregionaler Schlüssel jedoch zu Primär- oder Replikat wechseln kann, sollten Sie diese Bedingung in einer Schlüsselrichtlinie verwenden, um eine Produktion nur dann zu erlauben, wenn der bestimmte multiregionale Schlüssel ein Primär- oder Replikatschlüssel ist.

In diesem Beispiel verwendet die IAM-Richtlinienanweisung den Bedingungsschlüssel `kms:MultiRegionKeyType`, um Prinzipalen das Planen und Abbrechen einer Schlüssellöschung nur für multiregionale Replikatschlüssel im angegebenen AWS-Konto zu erlauben.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "arn:aws:kms:*:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:MultiRegionKeyType": "REPLICA"
    }
  }
}
```

Um den Zugriff auf alle multiregionale Schlüssel zu erlauben oder zu verweigern, können Sie beide Werte oder einen Nullwert mit `kms:MultiRegionKeyType` verwenden. Zu diesem Zweck wird jedoch der MultiRegion Bedingungsschlüssel kms: empfohlen.

km: PrimaryRegion

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:PrimaryRegion`	Zeichenfolge (Liste)	Einzelwertig	`UpdatePrimaryRegion`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um die Zielregionen in einem UpdatePrimaryRegionVorgang einzuschränken. Diese AWS-Regionen können Ihre Primärschlüssel für mehrere Regionen hosten.

Der `kms:PrimaryRegion` Bedingungsschlüssel steuert den Zugriff auf die UpdatePrimaryRegionOperation auf der Grundlage des `PrimaryRegion` Parameterwerts. Der `PrimaryRegion` Parameter gibt den AWS-Region Replikatschlüssel für mehrere Regionen an, der zum Primärschlüssel heraufgestuft wird. Der Wert der Bedingung besteht aus einem oder mehreren AWS-Region Namen, z. B. `us-east-1` oder`ap-southeast-2`, oder Regionsnamenmustern, wie `eu-*`

Die folgende Schlüsselrichtlinienanweisung verwendet beispielsweise den Bedingungsschlüssel `kms:PrimaryRegion`, um es Prinzipalen zu erlauben, die primäre Region eines multiregionalen Schlüssels auf eine der vier angegebenen Regionen zu aktualisieren.

```
{
  "Effect": "Allow",
  "Action": "kms:UpdatePrimaryRegion",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Developer"
  },
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:PrimaryRegion": [
         "us-east-1",
         "us-west-2",
         "eu-west-3",
         "ap-southeast-2"
      ]
    }
  }
}
```

km: ReEncryptOnSameKey

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:ReEncryptOnSameKey`	Boolesch	Einzelwertig	`ReEncrypt`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf den ReEncryptVorgang zu steuern, je nachdem, ob in der Anforderung ein KMS-Zielschlüssel angegeben ist, der derselbe ist, der für die ursprüngliche Verschlüsselung verwendet wurde.

Die folgende Richtlinienanweisung verwendet beispielsweise den Bedingungsschlüssel `kms:ReEncryptOnSameKey`, um zu erlauben, nur dann eine erneute Verschlüsselung vorzunehmen, wenn der verwendete Ziel-KMS-Schlüssel mit dem der ursprünglichen Verschlüsselung übereinstimmt.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:ReEncrypt*",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:ReEncryptOnSameKey": true
    }
  }
}
```

km: RequestAlias

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:RequestAlias`	Zeichenfolge (Liste)	Einzelwertig	Kryptografische Operationen DescribeKey GetPublicKey	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um eine Produktion nur dann zu erlauben, wenn die Anforderung einen bestimmten Alias zum Identifizieren des KMS-Schlüssels verwendet. Der Bedingungsschlüssel `kms:RequestAlias` steuert den Zugriff auf einen KMS-Schlüssel, der in einer kryptografischen Produktion verwendet wird, `GetPublicKey`, oder `DescribeKey`, basierend auf dem Alias, der diesen KMS-Schlüssel in der Anforderung identifiziert. (Diese Richtlinienbedingung hat keine Auswirkung auf den GenerateRandomVorgang, da der Vorgang keinen KMS-Schlüssel oder Alias verwendet.)

Diese Bedingung unterstützt die attributebasierte Zugriffskontrolle (ABAC) AWS KMS, mit der Sie den Zugriff auf KMS-Schlüssel anhand der Tags und Aliase eines KMS-Schlüssels steuern können. Sie können Tags und Aliase verwenden, um den Zugriff auf einen KMS-Schlüssel zu erlauben oder zu verweigern, ohne Richtlinien oder Erteilungen zu ändern. Details hierzu finden Sie unter ABAC für AWS KMS.

Um den Alias in dieser Richtlinienbedingung anzugeben, verwenden Sie einen Aliasnamen, wie `alias/project-alpha`, oder ein Alias-Namensmuster, wie `alias/*test*`. Sie können keinen Alias-ARN im Wert dieses Bedingungsschlüssels angeben.

Um diese Bedingung zu erfüllen, muss der Wert des `KeyId`-Parameters in der Anforderung ein übereinstimmender Aliasname oder Alias-ARN sein. Wenn die Anforderung einen anderen Schlüsselbezeichner verwendet, erfüllt er die Bedingung nicht, selbst wenn er denselben KMS-Schlüssel identifiziert.

Die folgende wichtige Richtlinienanweisung ermöglicht es dem Principal beispielsweise, den GenerateDataKeyVorgang mit dem KMS-Schlüssel aufzurufen. Dies ist jedoch nur zulässig, wenn der

Wert des `KeyId`-Parameters in der Anforderung `alias/finance-key` ist oder ein Alias-ARN mit diesem Aliasnamen, z. B. `arn:aws:kms:us-west-2:111122223333:alias/finance-key`.

```
{
  "Sid": "Key policy using a request alias condition",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/developer"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:RequestAlias": "alias/finance-key"
    }
  }
}
```

Sie können diesen Bedingungsschlüssel nicht verwenden, um den Zugriff auf Aliasoperationen wie CreateAliasoder zu steuern DeleteAlias. Weitere Hinweise zum Steuern des Zugriffs auf Alias-Operationen finden Sie unter Steuern des Zugriffs auf Aliasse.

km: ResourceAliases

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:ResourceAliases`	Zeichenfolge (Liste)	Mehrwertig	KMS-Schlüsselressourcen-Operationen	Nur IAM-Richtlinien

Verwenden Sie diesen Bedingungsschlüssel, um den Zugriff auf einen KMS-Schlüssel basierend auf den Aliasen, die dem KMS-Schlüssel zugeordnet sind. Bei der Operation muss es sich um eine KMS-Schlüsselressourcen-Operation handeln, das heißt, eine Operation, die für einen bestimmten KMS-Schlüssel autorisiert ist. Um die KMS-Schlüsselressourcen-Operationen zu identifizieren, suchen Sie in der Tabelle Actions and Resources (Aktionen und Ressourcen) Sie nach dem Wert von `KMS key` in der `Resources`-Spalte für die Produktion.

Diese Bedingung unterstützt attributbasierte Zugriffssteuerung (ABAC) in AWS KMS. Mit ABAC können Sie den Zugriff auf KMS-Schlüssel anhand der Tags steuern, die einem KMS-Schlüssel zugewiesen sind, und den Aliasen, die einem KMS-Schlüssel zugeordnet sind. Sie können Tags und Aliasse verwenden, um den Zugriff auf einen KMS-Schlüssel zu erlauben oder zu verweigern, ohne Richtlinien oder Erteilungen zu ändern. Details hierzu finden Sie unter ABAC für AWS KMS.

Ein Alias muss in einer AWS-Konto UND-Region eindeutig sein. Mit dieser Bedingung können Sie jedoch den Zugriff auf mehrere KMS-Schlüssel in derselben Region (mithilfe des `StringLike` Vergleichsoperators) oder auf mehrere KMS-Schlüssel in verschiedenen AWS-Regionen Konten steuern.

> **Note**
>
> Die kms: ResourceAliases -Bedingung ist nur wirksam, wenn der KMS-Schlüssel den Aliasnamen pro KMS-Schlüsselkontingent entspricht. Wenn ein KMS-Schlüssel dieses Kontingent überschreitet, wird auch Prinzipalen, die berechtigt sind, den KMS-Schlüssel zu nutzen, durch die Bedingung `kms:ResourceAliases` der Zugriff auf den KMS-Schlüssel verweigert.

Um den Alias in dieser Richtlinienbedingung anzugeben, verwenden Sie einen Aliasnamen, wie `alias/project-alpha`, oder ein Alias-Namensmuster, wie `alias/*test*`. Sie können keinen Alias-ARN im Wert dieses Bedingungsschlüssels angeben. Um die Bedingung zu erfüllen, muss der in der Produktion verwendete KMS-Schlüssel über den angegebenen Alias verfügen. Es spielt keine Rolle, ob oder wie der KMS-Schlüssel in der Anforderung für die Produktion identifiziert wird.

Dies ist ein mehrwertiger Bedingungsschlüssel, der den Satz von Aliasen, die einem KMS-Schlüssel zugeordnet sind, mit dem Satz von Aliasen in der Richtlinie vergleicht. Um zu bestimmen, wie diese Sätze verglichen werden, müssen Sie einen `ForAnyValue` oder `ForAllValues`-Satz-Operator in der Richtlinienbedingung angeben. Ausführliche Informationen zu den Satz-Operatoren finden Sie unter Verwenden mehrerer Schlüssel und Werte im IAM-Benutzerhandbuch.

- ForAnyValue: Mindestens ein dem KMS-Schlüssel zugeordneter Alias muss mit einem Alias in der Richtlinienbedingung übereinstimmen. Andere Aliase sind zulässig. Wenn der KMS-Schlüssel keine Aliase aufweist, ist die Bedingung nicht erfüllt.
- ForAllValues: Jeder Alias, der dem KMS-Schlüssel zugeordnet ist, muss mit einem Alias in der Richtlinie übereinstimmen. Dieser Satz-Operator beschränkt die Aliase, die dem KMS-Schlüssel

zugeordnet sind, auf diejenigen in der Richtlinienbedingung. Er erfordert keine Aliase, aber er verbietet nicht-spezifizierte Aliase.

Die folgende IAM-Richtlinienanweisung ermöglicht es dem Principal beispielsweise, den GenerateDataKeyVorgang für jeden KMS-Schlüssel in der angegebenen Datei aufzurufen AWS-Konto , der dem `finance-key` Alias zugeordnet ist. (Die Schlüsselrichtlinien der betroffenen KMS-Schlüssel müssen es auch dem Konto des Prinzipals erlauben, sie für diese Produktion zu verwenden.) Um anzuzeigen, dass die Bedingung erfüllt ist, wenn einer der vielen Aliase, die dem KMS-Schlüssel zugeordnet werden könnten, `alias/finance-key` ist, verwendet die Bedingung den `ForAnyValue`-Satz-Operator.

Da die `kms:ResourceAliases`-Bedingung auf der Ressource und nicht auf der Anforderung basiert, ist ein Aufruf an `GenerateDataKey` für jeden KMS-Schlüssel erfolgreich, der dem `finance-key`-Alias zugeordnet ist, auch wenn die Anforderung eine Schlüssel-ID oder einen Schüssel-ARN verwendet, um den KMS-Schlüssel zu identifizieren.

```
{
  "Sid": "AliasBasedIAMPolicy",
  "Effect": "Allow",
  "Action": "kms:GenerateDataKey",
  "Resource": [
    "arn:aws:kms:*:111122223333:key/*",
    "arn:aws:kms:*:444455556666:key/*"
  ],
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:ResourceAliases": "alias/finance-key"
    }
  }
}
```

Die folgende Beispiel-IAM-Richtlinienanweisung erlaubt es dem Prinzipal, KMS-Schlüssel zu aktivieren und zu deaktivieren, aber nur, wenn alle Aliase den KMS-Schlüssel "`Test`" enthalten. Diese Richtlinienanweisung verwendet zwei Bedingungen. Die Bedingung mit dem `ForAllValues`-Satz-Operator erfordert, dass alle Aliase, die dem KMS-Schlüssel zugeordnet sind, "Test" enthalten. Die Bedingung mit dem `ForAnyValue`-Satz-Operator erfordert, dass der KMS-Schlüssel mindestens einen Alias mit "Test" enthält. Ohne die `ForAnyValue`-Bedingung, hätte diese Richtlinienanweisung es dem Prinzipal erlaubt, KMS-Schlüssel zu verwenden, die keine Aliase hatten.

```
{
  "Sid": "AliasBasedIAMPolicy",
  "Effect": "Allow",
  "Action": [
    "kms:EnableKey",
    "kms:DisableKey"
  ],
  "Resource": "arn:aws:kms:*:111122223333:key/*",
  "Condition": {
    "ForAllValues:StringLike": {
      "kms:ResourceAliases": [
        "alias/*Test*"
      ]
    },
    "ForAnyValue:StringLike": {
      "kms:ResourceAliases": [
        "alias/*Test*"
      ]
    }
  }
}
```

km: ReplicaRegion

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:ReplicaRegion`	Zeichenfolge (Liste)	Einzelwertig	`ReplicateKey`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um einzuschränken, AWS-Regionen in dem ein Prinzipal einen Schlüssel für mehrere Regionen replizieren kann. Der `kms:ReplicaRegion` Bedingungsschlüssel steuert den Zugriff auf den ReplicateKeyVorgang auf der Grundlage des Werts des ReplicaRegionParameters in der Anforderung. Dieser Parameter gibt die AWS-Region für den neuen Replikatschlüssel an.

Der Wert der Bedingung besteht aus einem oder mehreren AWS-Region Namen, z. B. `us-east-1` oder`ap-southeast-2`, oder aus Namensmustern wie`eu-*`. Eine Liste der Namen, die AWS-Regionen das AWS KMS unterstützt, finden Sie unter AWS Key Management Service Endpunkte und Kontingente in der Allgemeine AWS-Referenz.

In der folgenden wichtigen Richtlinienanweisung wird beispielsweise der `kms:ReplicaRegion` Bedingungsschlüssel verwendet, damit Prinzipale den ReplicateKeyVorgang nur aufrufen können, wenn der Wert des `ReplicaRegion` Parameters einer der angegebenen Regionen entspricht.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Administrator"
  },
  "Action": "kms:ReplicateKey"
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ReplicaRegion": [
         "us-east-1",
         "eu-west-3",
         "ap-southeast-2"
      ]
    }
  }
}
```

Dieser Bedingungsschlüssel steuert nur den Zugriff auf den ReplicateKeyVorgang. Um den Zugriff auf den UpdatePrimaryRegionVorgang zu steuern, verwenden Sie den PrimaryRegion Bedingungsschlüssel kms:.

km: RetiringPrincipal

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:RetiringPrincipal`	Zeichenfolge (Liste)	Einzelwertig	`CreateGrant`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um den Zugriff auf die CreateGrantOperation anhand des RetiringPrincipalParameterwerts in der Anforderung zu steuern. Sie können beispielsweise das Erstellen von Erteilungen zur Verwendung eines KMS-Schlüssels nur erlauben, wenn der `RetiringPrincipal` in der `CreateGrant`-Anforderung dem in der Bedingungsanweisung angegebenen `RetiringPrincipal` entspricht.

Um den ausscheidenden Prinzipal anzugeben, verwenden Sie den Amazon-Ressourcennamen (ARN) eines AWS Prinzipals. Zu den gültigen Prinzipalen gehören AWS-Konten IAM-Benutzer, IAM-Rollen, Verbundbenutzer und Benutzer mit angenommenen Rollen. Hilfe zur ARN-Syntax für einen Prinzipal finden Sie unter IAM ARNs im IAM-Benutzerhandbuch.

Das folgende Beispiel für eine Schlüsselrichtlinienanweisung ermöglicht es einem Benutzer, Berechtigungen für den KMS-Schlüssel zu erstellen. Der `kms:RetiringPrincipal` Bedingungsschlüssel beschränkt die Berechtigung auf `CreateGrant` Anfragen, bei denen der ausscheidende Schulleiter im Zuschuss der ist. `LimitedAdminRole`

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:RetiringPrincipal": "arn:aws:iam::111122223333:role/LimitedAdminRole"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: GrantConstraintType
- km: GrantIsFor AWSResource
- km: GrantOperations
- km: GranteePrincipal

km: RotationPeriodInDays

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:RotationPeriodInDays`	Numerischer Wert	Einzelwertig	`EnableKeyRotation`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um die Werte einzuschränken, die Prinzipale im `RotationPeriodInDays` Parameter einer EnableKeyRotationAnfrage angeben können.

Der `RotationPeriodInDays` gibt die Anzahl der Tage zwischen den einzelnen automatischen Schlüsselrotationsdaten an. AWS KMS ermöglicht es Ihnen, einen Rotationszeitraum zwischen 90 und 2560 Tagen anzugeben, aber Sie können die `kms:RotationPeriodInDays` Bedingungstaste verwenden, um den Rotationszeitraum weiter einzuschränken, indem Sie beispielsweise eine Mindestrotationsperiode innerhalb des gültigen Bereichs erzwingen.

In der folgenden wichtigen Richtlinienanweisung wird beispielsweise der `kms:RotationPeriodInDays` Bedingungsschlüssel verwendet, um zu verhindern, dass Prinzipale die Schlüsselrotation aktivieren, wenn der Rotationszeitraum 180 Tage oder weniger beträgt.

```
{
  "Effect": "Deny",
  "Action": "kms:EnableKeyRotation",
  "Principal": "*",
  "Resource": "*",
  "Condition" : {
      "NumericLessThanEquals" : {
        "kms:RotationPeriodInDays" : "180"
      }
  }
}
```

km: ScheduleKeyDeletionPendingWindowInDays

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:ScheduleKeyDeletionPendingWindowInDays`	Numerischer Wert	Einzelwertig	`ScheduleKeyDeletion`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können diesen Bedingungsschlüssel verwenden, um die Werte einzuschränken, die Prinzipale im `PendingWindowInDays` Parameter einer ScheduleKeyDeletionAnfrage angeben können.

Der `PendingWindowInDays` gibt die Anzahl der Tage an, die warten, bis ein Schlüssel gelöscht AWS KMS wird. AWS KMS ermöglicht es Ihnen, eine Wartezeit zwischen 7 und 30 Tagen anzugeben, aber Sie können den `kms:ScheduleKeyDeletionPendingWindowInDays` Bedingungsschlüssel verwenden, um die Wartezeit weiter einzuschränken, indem Sie beispielsweise eine Mindestwartezeit innerhalb des gültigen Bereichs erzwingen.

Die folgende Schlüsselrichtlinienanweisung verwendet beispielsweise den `kms:ScheduleKeyDeletionPendingWindowInDays`-Bedingungsschlüssel, um zu verhindern, dass Prinzipale das Löschen von Schlüsseln planen, wenn die Wartezeit weniger als oder gleich 21 Tage beträgt.

```
{
  "Effect": "Deny",
  "Action": "kms:ScheduleKeyDeletion",
  "Principal": "*",
  "Resource": "*",
  "Condition" : {
      "NumericLessThanEquals" : {
        "kms:ScheduleKeyDeletionPendingWindowInDays" : "21"
      }
  }
}
```

km: SigningAlgorithm

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:SigningAlgorithm`	String	Einzelwertig	`Sign` `Verify`	Schlüsselrichtlinien und IAM-Richtlinien

Sie können den `kms:SigningAlgorithm` Bedingungsschlüssel verwenden, um den Zugriff auf die Vorgänge Signieren und Verifizieren auf der Grundlage des SigningAlgorithmParameterwerts in der Anforderung zu steuern. Dieser Bedingungsschlüssel hat keine Auswirkung auf Operationen, die außerhalb von ausgeführt werden AWS KMS, wie z. B. die Überprüfung von Signaturen mit dem öffentlichen Schlüssel in einem asymmetrischen KMS-Schlüsselpaar außerhalb von. AWS KMS

Die folgende Beispiel-Schlüsselrichtlinie erlaubt es Benutzern, die die `testers`-Rolle annehmen können, den KMS-Schlüssel zum Signieren von Nachrichten nur dann zu verwenden, wenn der für die Anforderung verwendete Signaturalgorithmus ein RSASSA_PSS-Algorithmus ist, z. B `RSASSA_PSS_SHA512`.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/testers"
  },
  "Action": "kms:Sign",
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:SigningAlgorithm": "RSASSA_PSS*"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: EncryptionAlgorithm
- the section called “km: MacAlgorithm”

- the section called “km: MessageType”

km: ValidTo

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:ValidTo`	Zeitstempel	Einzelwertig	`ImportKeyMaterial`	Schlüsselrichtlinien und IAM-Richtlinien

Der `kms:ValidTo` Bedingungsschlüssel steuert den Zugriff auf die ImportKeyMaterialOperation auf der Grundlage des Werts des ValidToParameters in der Anforderung, der bestimmt, wann das importierte Schlüsselmaterial abläuft. Der Wert wird im Unix-Zeitformat angegeben.

Standardmäßig ist der `ValidTo`-Parameter in einer `ImportKeyMaterial`-Anforderung erforderlich. Wenn der Wert des ExpirationModelParameters jedoch ist`KEY_MATERIAL_DOES_NOT_EXPIRE`, ist der `ValidTo` Parameter ungültig. Sie können auch den ExpirationModel Bedingungsschlüssel kms: verwenden, um den `ExpirationModel` Parameter oder einen bestimmten Parameterwert anzufordern.

Die folgende Beispiel-Richtlinienanweisung erlaubt es einem Benutzer, Schlüsselmaterial in einen KMS-Schlüssel zu importieren. Der `kms:ValidTo`-Bedingungsschlüssel beschränkt die Berechtigung für `ImportKeyMaterial`-Anforderungen, bei denen der `ValidTo`-Wert kleiner oder gleich `1546257599.0` ist (31. Dezember 2018 23:59:59).

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:ImportKeyMaterial",
  "Resource": "*",
  "Condition": {
    "NumericLessThanEquals": {
      "kms:ValidTo": "1546257599.0"
    }
```

```
  }
}
```

Informationen finden Sie auch unter:

- km: ExpirationModel
- km: WrappingAlgorithm
- km: WrappingKeySpec

km: ViaService

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:ViaService`	String	Einzelwertig	KMS-Schlüsselressourcen-Operationen	Schlüsselrichtlinien und IAM-Richtlinien

Der `kms:ViaService` Bedingungsschlüssel beschränkt die Verwendung eines KMS-Schlüssels auf Anfragen von bestimmten Personen AWS-Services. Dieser Bedingungsschlüssel gilt nur für Forward-Access-Sitzungen. In jedem `kms:ViaService`-Bedingungsschlüssel können Sie einen oder mehrere Services angeben. Bei der Operation muss es sich um eine KMS-Schlüsselressourcen-Operation handeln, das heißt, eine Operation, die für einen bestimmten KMS-Schlüssel autorisiert ist. Um die KMS-Schlüsselressourcen-Operationen zu identifizieren, suchen Sie in der Tabelle Actions and Resources (Aktionen und Ressourcen) Sie nach dem Wert von `KMS key` in der `Resources`-Spalte für die Produktion.

In der folgenden Grundsatzerklärung wird beispielsweise der `kms:ViaService` Bedingungsschlüssel verwendet, um zu ermöglichen, dass ein vom Kunden verwalteter Schlüssel nur dann für die angegebenen Aktionen verwendet wird, wenn die Anfrage von Amazon EC2 oder Amazon RDS in der Region USA West (Oregon) im Namen von kommt`ExampleRole`.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
```

```
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:DescribeKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "ec2.us-west-2.amazonaws.com",
        "rds.us-west-2.amazonaws.com"
      ]
    }
  }
}
```

Außerdem können Sie mit dem Bedingungsschlüssel `kms:ViaService` die Berechtigung zur Verwendung eines KMS-Schlüssels verweigern, wenn die Anforderung von bestimmten Services stammt. Die folgende Richtlinienanweisung einer Schlüsselrichtlinie verwendet beispielsweise einen `kms:ViaService`-Bedingungsschlüssel, um zu verhindern, dass ein kundenverwalteter KMS-Schlüssel für `Encrypt`-Operationen verwendet wird, wenn die Anforderung im Namen von `ExampleRole` von AWS Lambda ausgegeben wird.

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
          "lambda.us-west-2.amazonaws.com"
      ]
```

```
    }
  }
}
```

> **Important**
>
> Wenn Sie den Bedingungsschlüssel `kms:ViaService` verwenden, gibt der Service die Anforderung im Namen eines Prinzipals im AWS-Konto aus. Diese Prinzipale müssen über die folgenden Berechtigungen verfügen:
>
> - Berechtigung zur Verwendung des KMS-Schlüssels Der Prinzipal muss dem integrierten Service diese Berechtigungen erteilen, sodass der Service den kundenverwalteten Schlüssel im Auftrag des Prinzipals verwenden kann. Weitere Informationen finden Sie unter AWS KMS Verschlüsselung mit AWS Diensten verwenden.
> - Berechtigung zur Verwendung des integrierten Service. Einzelheiten darüber, wie Sie Benutzern Zugriff auf einen AWS Dienst gewähren, der sich in integrieren lässt AWS KMS, finden Sie in der Dokumentation für den integrierten Dienst.

Alle Von AWS verwaltete Schlüssel verwenden einen `kms:ViaService`-Bedingungsschlüssel in ihrem Schlüsselrichtliniendokument. Diese Bedingung erlaubt es dem KMS-Schlüssel, nur für Anforderungen verwendet zu werden, die von dem Service stammen, der den KMS-Schlüssel erstellt hat. Um die wichtigsten Richtlinien für einen zu sehen Von AWS verwalteter Schlüssel, verwenden Sie den GetKeyPolicyVorgang.

Der Bedingungsschlüssel `kms:ViaService` ist in IAM- und Schlüsselrichtlinienanweisungen gültig. Die von Ihnen angegebenen Services müssen inin AWS KMS integriert sein und den Bedingungsschlüssel `kms:ViaService` unterstützen.

Services, die den Bedingungsschlüssel **`kms:ViaService`** unterstützen

In der folgenden Tabelle sind AWS Dienste aufgeführt, die in den `kms:ViaService` Bedingungsschlüssel integriert sind AWS KMS und dessen Verwendung in vom Kunden verwalteten Schlüsseln unterstützen. Die Dienste in dieser Tabelle sind möglicherweise nicht in allen Regionen verfügbar. Verwenden Sie das `.amazonaws.com` Suffix des AWS KMS ViaService Namens in allen AWS Partitionen.

> **Note**
>
> Möglicherweise müssen Sie horizontal oder vertikal scrollen, um alle Daten in dieser Tabelle anzuzeigen.

Service-Name	AWS KMS ViaService Name
Amazon-KI-Operationen	`aiops.`*`AWS_region`*` .amazonaws.com`
AWS App Runner	`apprunner.`*`AWS_region`*` .amazonaws.com`
AWS AppFabric	`appfabric.`*`AWS_region`*` .amazonaws.com`
Amazon AppFlow	`appflow.`*`AWS_region`*` .amazonaws.com`
AWS Application Migration Service	`mgn.`*`AWS_region`*` .amazonaws.com`
Amazon Athena	`athena.`*`AWS_region`*` .amazonaws.com`
AWS Audit Manager	`auditmanager.`*`AWS_region`*` .amazonaws.com`
Amazon Aurora	`rds.`*`AWS_region`*` .amazonaws.com`
AWS Backup	`backup.`*`AWS_region`*` .amazonaws.com`
AWS Backup Gateway	`backup-gateway.`*`AWS_region`*` .amazonaws.com`
Amazon Bedrock Modellkopie	`bedrock.`*`AWS_region`*` .amazonaws.com`
Amazon Chime SDK	`chimevoiceconnector.`*`AWS_region`*` .amazonaws.com`
AWS Clean Rooms ML	`cleanrooms-ml.`*`AWS_region`*` .amazonaws.com`

Service-Name	AWS KMS ViaService Name
AWS CodeArtifact	`codeartifact.` *`AWS_region`* `.amazonaws.com`
CodeGuru Amazon-Rezensent	`codeguru-reviewer.` *`AWS_region`* `.amazonaws.com`
Amazon Comprehend	`comprehend.` *`AWS_region`* `.amazonaws.com`
Amazon Connect	`connect.`*`AWS_region`* `.amazonaws.com`
Amazon Connect Customer Profiles	`profile.`*`AWS_region`* `.amazonaws.com`
Amazon Q in Connect	`wisdom.`*`AWS_region`* `.amazonaws.com`
AWS Database Migration Service (AWS DMS)	`dms.`*`AWS_region`* `.amazonaws.com`
AWS DeepRacer	`deepracer.` *`AWS_region`* `.amazonaws.com`
AWS Directory Service	`directoryservice.` *`AWS_region`* `.amazonaws.com`
Amazon DocumentDB	`docdb-elastic.` *`AWS_region`* `.amazonaws.com`
Amazon-DynamoDB	`dynamodb.` *`AWS_region`* `.amazonaws.com`
EC2 Amazon-Systemmanager (SSM)	`ssm.`*`AWS_region`* `.amazonaws.com`
Amazon Elastic Block Store (Amazon EBS)	`ec2.`*`AWS_region`* `.amazonaws.com` (Nur EBS)
Amazon Elastic Container Registry (Amazon ECR)	`ecr.`*`AWS_region`* `.amazonaws.com`

Service-Name	AWS KMS ViaService Name
Amazon Elastic File System (Amazon EFS)	`elasticfilesystem.` *`AWS_region`* `.amazonaws.com`
Amazon ElastiCache	Nehmen Sie beide ViaService Namen in den Wert des Bedingungsschlüssels auf: • `elasticache.` *`AWS_region`* `.amazonaws.com` • `dax.`*`AWS_region`* `.amazonaws.com`
AWS Elemental MediaTailor	`mediatailor.` *`AWS_region`* `.amazonaws.com`
AWS Auflösung der Entität	`entityresolution.` *`AWS_region`* `.amazonaws.com`
Amazon EventBridge	`events.`*`AWS_region`* `.amazonaws.com`
Amazon FinSpace	`finspace.` *`AWS_region`* `.amazonaws.com`
Amazon Forecast	`forecast.` *`AWS_region`* `.amazonaws.com`
Amazon FSx	`fsx.`*`AWS_region`* `.amazonaws.com`
AWS Glue	`glue.`*`AWS_region`* `.amazonaws.com`
AWS Ground Station	`groundstation.` *`AWS_region`* `.amazonaws.com`
Amazon GuardDuty	`malware-protection.` *`AWS_region`* `.amazonaws.com`
AWS HealthLake	`healthlake.` *`AWS_region`* `.amazonaws.com`

Service-Name	AWS KMS ViaService Name
AWS IoT SiteWise	`iotsitewise.` *`AWS_region`* `.amazonaws.com`
Amazon Kendra	`kendra.`*`AWS_region`* `.amazonaws.com`
Amazon Keyspaces (für Apache Cassandra)	`cassandra.` *`AWS_region`* `.amazonaws.com`
Amazon Kinesis	`kinesis.`*`AWS_region`* `.amazonaws.com`
Amazon Data Firehose	`firehose.` *`AWS_region`* `.amazonaws.com`
Amazon Kinesis Video Streams	`kinesisvideo.` *`AWS_region`* `.amazonaws.com`
AWS Lambda	`lambda.`*`AWS_region`* `.amazonaws.com`
Amazon Lex	`lex.`*`AWS_region`* `.amazonaws.com`
AWS License Manager	`license-manager.` *`AWS_region`* `.amazonaws.com`
Amazon Location Service	`geo.`*`AWS_region`* `.amazonaws.com`
Amazon Lookout für Equipment	`lookoutequipment.` *`AWS_region`* `.amazonaws.com`
Amazon Lookout für Metrics	`lookoutmetrics.` *`AWS_region`* `.amazonaws.com`
Amazon Lookout für Vision	`lookoutvision.` *`AWS_region`* `.amazonaws.com`
Amazon Macie	`macie.`*`AWS_region`* `.amazonaws.com`
AWS Mainframe Modernization	`m2.`*`AWS_region`* `.amazonaws.com`

Service-Name	AWS KMS ViaService Name
AWS Mainframe Modernization Testen von Anwendungen	`apptest.`*`AWS_region`*` .amazonaws.com`
Amazon Managed Blockchain	`managedblockchain.`*`AWS_region`*` .amazonaws.com`
Amazon Managed Streaming for Apache Kafka (Amazon MSK)	`kafka.`*`AWS_region`*` .amazonaws.com`
Amazon Managed Workflows for Apache Airflow (MWAA)	`airflow.`*`AWS_region`*` .amazonaws.com`
Amazon MemoryDB	`memorydb.`*`AWS_region`*` .amazonaws.com`
Amazon Monitron	`monitron.`*`AWS_region`*` .amazonaws.com`
Amazon MQ	`mq.`*`AWS_region`*` .amazonaws.com`
Amazon Neptune	`rds.`*`AWS_region`*` .amazonaws.com`
Amazon Nimble Studio	`nimble.`*`AWS_region`*` .amazonaws.com`
AWS HealthOmics	`omics.`*`AWS_region`*` .amazonaws.com`
OpenSearch Amazon-Dienst	`es.`*`AWS_region`*` .amazonaws.com` , `aoss.`*`AWS_region`*` .amazonaws.com`
OpenSearch Maßgeschneiderte Amazon-Pakete	`custom-packages.`*`AWS_region`*` .amazonaws.com`
AWS Proton	`proton.`*`AWS_region`*` .amazonaws.com`
Amazon Quantum Ledger Database (Amazon QLDB)	`qldb.`*`AWS_region`*` .amazonaws.com`
Erkenntnisse zur Amazon-RDS-Leistung	`rds.`*`AWS_region`*` .amazonaws.com`

Service-Name	AWS KMS ViaService Name
Amazon Redshift	`redshift.` *`AWS_region`* `.amazonaws.com`
Amazon Redshift Query Editor V2	`sqlworkbench.` *`AWS_region`* `.amazonaws.com`
Amazon Redshift Serverless	`redshift-serverless.` *`AWS_region`* `.amazonaws.com`
Amazon Rekognition	`rekognition.` *`AWS_region`* `.amazonaws.com`
Amazon Relational Database Service (Amazon RDS)	`rds.`*`AWS_region`* `.amazonaws.com`
Amazon Replicated Data Store	`ards.`*`AWS_region`* `.amazonaws.com`
Amazon SageMaker KI	`sagemaker.` *`AWS_region`* `.amazonaws.com`
AWS Secrets Manager	`secretsmanager.` *`AWS_region`* `.amazonaws.com`
Amazon Security Lake	`securitylake.` *`AWS_region`* `.amazonaws.com`
Amazon Simple Email Service (Amazon SES)	`ses.`*`AWS_region`* `.amazonaws.com`
Amazon Simple Notification Service (Amazon SNS)	`sns.`*`AWS_region`* `.amazonaws.com`
Amazon Simple Queue Service (Amazon SQS)	`sqs.`*`AWS_region`* `.amazonaws.com`
Amazon Simple Storage Service (Amazon S3)	`s3.`*`AWS_region`* `.amazonaws.com`
Amazon S3 Tables	`s3tables.` *`AWS_region`* `.amazonaws.com`

Service-Name	AWS KMS ViaService Name
AWS Snowball Edge	`importexport.` *`AWS_region`* `.amazonaws.com`
AWS Step Functions	`states.`*`AWS_region`* `.amazonaws.com`
AWS Storage Gateway	`storagegateway.` *`AWS_region`* `.amazonaws.com`
AWS Systems Manager Incident Manager	`ssm-incidents.` *`AWS_region`* `.amazonaws.com`
AWS Systems Manager Incident Manager Kontakte	`ssm-contacts.` *`AWS_region`* `.amazonaws.com`
Amazon Timestream	`timestream.` *`AWS_region`* `.amazonaws.com`
Amazon Translate	`translate.` *`AWS_region`* `.amazonaws.com`
AWS Verified Access	`verified-access.` *`AWS_region`* `.amazonaws.com`
Amazon WorkMail	`workmail.` *`AWS_region`* `.amazonaws.com`
Amazon WorkSpaces	`workspaces.` *`AWS_region`* `.amazonaws.com`
Amazon WorkSpaces Thin Client	`thinclient.` *`AWS_region`* `.amazonaws.com`
Amazon WorkSpaces Web	`workspaces-web.` *`AWS_region`* `.amazonaws.com`
AWS X-Ray	`xray.`*`AWS_region`* `.amazonaws.com`

km: WrappingAlgorithm

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:WrappingAlgorithm`	String	Einzelwertig	`GetParametersForImport`	Schlüsselrichtlinien und IAM-Richtlinien

Dieser Bedingungsschlüssel steuert den Zugriff auf die GetParametersForImportOperation auf der Grundlage des Werts des WrappingAlgorithmParameters in der Anforderung. Sie können diese Bedingung verwenden, damit die Prinzipale während des Importvorgangs einen bestimmten Algorithmus zur Verschlüsselung von Schlüsselmaterial verwenden müssen. Anforderungen für den erforderlichen öffentlichen Schlüssel und Import-Token schlagen fehl, wenn sie einen anderen Wrapping-Algorithmus angeben.

Die folgende Beispiel-Richtlinienanweisung verwendet den Bedingungsschlüssel `kms:WrappingAlgorithm`, um dem Beispielbenutzer die Berechtigung zum Aufrufen der Produktion `GetParametersForImport` zu geben, wobei jedoch die Verwendung des Verpackungsalgorithmus `RSAES_OAEP_SHA_1` verhindert wird. Wenn der `WrappingAlgorithm` in der `GetParametersForImport`-Anforderung `RSAES_OAEP_SHA_1` lautet, schlägt die Produktion fehl.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:GetParametersForImport",
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "kms:WrappingAlgorithm": "RSAES_OAEP_SHA_1"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: ExpirationModel
- km: ValidTo
- km: WrappingKeySpec

km: WrappingKeySpec

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:WrappingKeySpec`	String	Einzelwertig	`GetParametersForImport`	Schlüsselrichtlinien und IAM-Richtlinien

Dieser Bedingungsschlüssel steuert den Zugriff auf die GetParametersForImportOperation auf der Grundlage des Werts des WrappingKeySpecParameters in der Anforderung. Sie können diese Bedingung verwenden, um von den Prinzipalen zu verlangen, dass sie während des Importvorgangs einen bestimmten Typ eines öffentlichen Schlüssels verwenden. Wenn die Anforderung einen anderen Schlüsseltyp angibt, schlägt sie fehl.

Da der einzige gültige Wert für den `WrappingKeySpec`-Parameter `RSA_2048` lautet, werden die Benutzer durch Verweigern dieses Werts effektiv an der Nutzung der `GetParametersForImport`-Produktion gehindert.

Das folgende Richtlinienanweisungsbeispiel verwendet den `kms:WrappingAlgorithm`-Bedingungsschlüssel, um zu erfordern, dass der Wert für `WrappingKeySpec` in der Anforderung `RSA_4096` lautet.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:GetParametersForImport",
  "Resource": "*",
```

```
  "Condition": {
    "StringEquals": {
      "kms:WrappingKeySpec": "RSA_4096"
    }
  }
}
```

Informationen finden Sie auch unter:

- km: ExpirationModel
- km: ValidTo
- km: WrappingAlgorithm

AWS KMS Bedingungsschlüssel für AWS Nitro Enclaves

AWS Nitro Enclaves ist eine EC2 Amazon-Funktion, mit der Sie isolierte Computerumgebungen, sogenannte Enklaven, erstellen können, um hochsensible Daten zu schützen und zu verarbeiten. AWS KMS bietet Bedingungsschlüssel zur Unterstützung von Nitro Enclaves. AWS Diese Bedingungsschlüssel gelten nur für Anfragen nach einer AWS KMS Nitro-Enklave.

Wenn Sie die Decrypt -,, DeriveSharedSecretGenerateDataKeyGenerateDataKeyPair, oder GenerateRandomAPI-Operationen mit dem signierten Beglaubigungsdokument aus einer Enklave aufrufen, APIs verschlüsseln diese den Klartext in der Antwort unter dem öffentlichen Schlüssel aus dem Beglaubigungsdokument und geben Chiffretext statt Klartext zurück. Dieser Geheimtext kann nur mit dem privaten Schlüssel in der Enklave entschlüsselt werden. Weitere Informationen finden Sie unter Kryptografische Bescheinigung für Nitro-Enklaven AWS.

> **Note**
>
> Wenn Sie bei der Erstellung eines Schlüssels keine Schlüsselrichtlinie angeben, erstellt er eine für Sie. AWS KMS AWS Diese Standard-Schlüsselrichtlinie gewährt denjenigen AWS-Konten , denen der KMS-Schlüssel gehört, vollen Zugriff auf den Schlüssel und ermöglicht es dem Konto, IAM-Richtlinien zu verwenden, um den Zugriff auf den Schlüssel zu ermöglichen. Diese Richtlinie ermöglicht alle Aktionen wie Decrypt. AWS empfiehlt, Principal of Berechtigungen mit den geringsten Rechten auf Ihre KMS-Schlüsselrichtlinien anzuwenden. Sie können den Zugriff auch einschränken, indem Sie die Aktion der KMS-Schlüsselrichtlinie für `kms:*` to ändernNotAction:`kms:Decrypt`.

Mit den folgenden Bedingungsschlüsseln können Sie die Berechtigungen für diese Operationen anhand des Inhalts des signierten Bescheinigungsdokuments einschränken. Bevor Sie einen Vorgang zulassen, wird das Bescheinigungsdokument aus der Enklave mit den Werten in diesen AWS KMS Bedingungsschlüsseln AWS KMS verglichen.

km: 384 RecipientAttestation ImageSha

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp

`kms:RecipientAttestation:ImageSha384`	String	Einzelwertig	`Decrypt`
			`DeriveSharedSecret`
			`GenerateDataKey`
			`GenerateDataKeyPair`
			`GenerateRandom`
				Schlüsselrichtlinien und IAM-Richtlinien

Der `kms:RecipientAttestation:ImageSha384` Bedingungsschlüssel steuert den Zugriff auf `DecryptDeriveSharedSecret`,`GenerateDataKey`,`GenerateDataKeyPair`, und `GenerateRandom` mit einem KMS-Schlüssel, wenn der Bildauszug aus dem signierten Bestätigungsdokument in der Anforderung mit dem Wert im Bedingungsschlüssel übereinstimmt. Der `ImageSha384`-Wert entspricht PCR0 im Bescheinigungsdokument. Dieser Bedingungsschlüssel ist nur wirksam, wenn der `Recipient` Parameter in der Anforderung ein signiertes Bestätigungsdokument für eine Nitro-Enklave angibt. AWS

Dieser Wert ist auch in CloudTrailEreignissen für Anfragen an Nitro-Enklaven enthalten. AWS KMS

Die folgende wichtige Richtlinienanweisung ermöglicht es der `data-processing` Rolle beispielsweise, den KMS-Schlüssel für Decrypt -,, DeriveSharedSecretGenerateDataKey, GenerateDataKeyPair- und -Operationen zu verwenden. GenerateRandom Der Bedingungsschlüssel

`kms:RecipientAttestation:ImageSha384` erlaubt die Operationen nur, wenn der Bild-Digest-Wert (PCR0) des Bescheinigungsdokuments in der Anforderung mit dem Bild-Digest-Wert in der Bedingung übereinstimmt. Dieser Bedingungsschlüssel ist nur wirksam, wenn der `Recipient` Parameter in der Anforderung ein signiertes Bestätigungsdokument für eine AWS Nitro-Enklave angibt.

Wenn die Anfrage kein gültiges Bescheinigungsdokument aus einer AWS Nitro-Enklave enthält, wird die Genehmigung verweigert, da diese Bedingung nicht erfüllt ist.

```
{
  "Sid" : "Enable enclave data processing",
  "Effect" : "Allow",
  "Principal" : {
    "AWS" : "arn:aws:iam::111122223333:role/data-processing"
  },
  "Action": [
    "kms:Decrypt",
    "kms:DeriveSharedSecret",
    "kms:GenerateDataKey",
    "kms:GenerateDataKeyPair",
    "kms:GenerateRandom"
  ],
  "Resource" : "*",
  "Condition": {
    "StringEqualsIgnoreCase": {
      "kms:RecipientAttestation:ImageSha384":
 "9fedcba8abcdef7abcdef6abcdef5abcdef4abcdef3abcdef2abcdef1abcdef1abcdef0abcdef1abcdef2abcdef3a
    }
  }
}
```

RecipientAttestationkm: :PCR <PCR_ID>

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp
`kms:RecipientAttes`	String	Einzelwertig	`Decrypt` `DeriveSharedSecret`	Schlüsselrichtlinien und IAM-Richtlinien

AWS KMS Zustandsschlüssel	Bedingungstyp	Werttyp	API-Operationen	Richtlinientyp

`tation:PCR<PCR_ID>`			`GenerateDataKey`
			`GenerateDataKeyPair`
			`GenerateRandom`	

Der `kms:RecipientAttestation:PCR<PCR_ID>` Bedingungsschlüssel steuert den Zugriff auf `DecryptDeriveSharedSecret`,`GenerateDataKey`,`GenerateDataKeyPair`, und `GenerateRandom` mit einem KMS-Schlüssel nur, wenn die Registrierung (PCRs) der Plattformkonfiguration aus dem signierten Bestätigungsdokument in der Anfrage mit denen PCRs im Bedingungsschlüssel übereinstimmt. Dieser Bedingungsschlüssel ist nur wirksam, wenn der `Recipient` Parameter in der Anfrage ein signiertes Bestätigungsdokument aus einer Nitro-Enklave angibt. AWS

Dieser Wert ist auch in CloudTrailEreignissen enthalten, die Anfragen an Nitro-Enklaven darstellen. AWS KMS

Verwenden Sie das folgende Format, um einen PCR-Wert anzugeben. Verketten Sie die PCR-ID mit dem Bedingungsschlüssel-Namen. Sie können eine PCR-ID angeben, die eine der sechs Enklavenmessungen identifiziert, oder eine benutzerdefinierte PCR-ID, die Sie für einen bestimmten Anwendungsfall definiert haben. Der PCR-Wert muss eine Hexadezimalzeichenfolge in Kleinbuchstaben von bis zu 96 Bytes sein.

```
"kms:RecipientAttestation:PCRPCR_ID": "PCR_value"
```

Der folgende Bedingungsschlüssel gibt beispielsweise einen bestimmten Wert für an PCR1, der dem Hash des Kernels entspricht, der für die Enklave und den Bootstrap-Prozess verwendet wird.

```
kms:RecipientAttestation:PCR1:
 "0x1abcdef2abcdef3abcdef4abcdef5abcdef6abcdef7abcdef8abcdef9abcdef8abcdef7abcdef6abcdef5abcdef
```

Die folgende Beispiel-Schlüsselrichtlinienanweisung erlaubt es der`data-processing` -Rolle, den KMS-Schlüssel für die Operation Decrypt zu verwenden.

Der `kms:RecipientAttestation:PCR` Bedingungsschlüssel in dieser Anweisung lässt den Vorgang nur zu, wenn der PCR1 Wert im signierten Bestätigungsdokument in der Anforderung mit dem `kms:RecipientAttestation:PCR1` Wert in der Bedingung übereinstimmt. Verwenden des `StringEqualsIgnoreCase`-Richtlinienoperators, um einen Vergleich der PCR-Werte ohne Berücksichtigung der Groß-/Kleinschreibung zu erfordern.

Wenn die Anforderung kein Bescheinigungsdokument enthält, wird die Berechtigung verweigert, da diese Bedingung nicht erfüllt ist.

```
{
  "Sid" : "Enable enclave data processing",
  "Effect" : "Allow",
  "Principal" : {
    "AWS" : "arn:aws:iam::111122223333:role/data-processing"
  },
  "Action": "kms:Decrypt",
  "Resource" : "*",
  "Condition": {
    "StringEqualsIgnoreCase": {
      "kms:RecipientAttestation:PCR1":
 "0x1de4f2dcf774f6e3b679f62e5f120065b2e408dcea327bd1c9dddaea6664e7af79355814748447674530 82c6f15
    }
  }
}
```

Berechtigungen mit den geringsten Rechten

Da Ihre KMS-Schlüssel vertrauliche Informationen schützen, empfehlen wir, dem Prinzip des Zugriffs mit den geringsten Rechten zu folgen. Delegieren Sie die für die Ausführung einer Aufgabe erforderlichen Mindestberechtigungen, wenn Sie Ihre wichtigsten Richtlinien definieren. Lassen Sie alle Aktionen (`kms:*`) für eine KMS-Schlüsselrichtlinie nur zu, wenn Sie beabsichtigen, die Berechtigungen mit zusätzlichen IAM-Richtlinien weiter einzuschränken. Wenn Sie beabsichtigen, Berechtigungen mit IAM-Richtlinien zu verwalten, schränken Sie ein, wer IAM-Richtlinien erstellen und an IAM-Prinzipale anhängen darf, und achten Sie auf Richtlinienänderungen.

Wenn Sie alle Aktionen (`kms:*`) sowohl in der Schlüsselrichtlinie als auch in der IAM-Richtlinie zulassen, verfügt der Prinzipal sowohl über Administratorrechte als auch über

Nutzungsberechtigungen für den KMS-Schlüssel. Aus Sicherheitsgründen empfehlen wir, diese Berechtigungen nur an bestimmte Prinzipale zu delegieren. Sie können dies tun, indem Sie den Prinzipal in der Schlüsselrichtlinie explizit benennen oder indem Sie einschränken, an welche Prinzipale die IAM-Richtlinie angehängt ist. Sie können auch Bedingungsschlüssel verwenden, um Berechtigungen einzuschränken. Beispielsweise können Sie die verwenden, `aws:PrincipalTag`um alle Aktionen zuzulassen, wenn der Principal, der den API-Aufruf durchführt, das in der Bedingungsregel angegebene Tag hat.

Weitere Informationen darüber, wie Richtlinienaussagen bewertet werden AWS, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch. Wir empfehlen, dieses Thema zu lesen, bevor Sie Richtlinien verfassen, um die Wahrscheinlichkeit zu verringern, dass Ihre Richtlinie unbeabsichtigte Auswirkungen hat, wie z. B. die Gewährung von Zugriff für Prinzipale, die keinen Zugriff haben sollten.

> **Tip**
>
> Wenn Sie eine Anwendung in einer Umgebung außerhalb der Produktionsumgebung testen, verwenden Sie IAM Access Analyzer, um Ihren IAM-Richtlinien die geringsten Rechte zuzuweisen.

Wenn Sie IAM-Benutzer anstelle von IAM-Rollen verwenden, empfehlen wir dringend, die AWS Multi-Faktor-Authentifizierung (MFA) zu aktivieren, um die Sicherheitsanfälligkeit langfristiger Anmeldeinformationen zu verringern. Über MFA können Sie die folgenden Aktionen ausführen:

- Erfordern Sie, dass Benutzer ihre Anmeldeinformationen mit MFA validieren, bevor sie privilegierte Aktionen ausführen, z. B. das Löschen von Schlüsseln planen.
- Teilen Sie den Besitz eines Administratorkontokennworts und eines MFA-Geräts auf einzelne Personen auf, um die geteilte Autorisierung zu implementieren.

Weitere Informationen

- AWS verwaltete Richtlinien für Jobfunktionen
- Techniken zum Erstellen von IAM-Richtlinien mit geringsten Berechtigungen

Implementieren der geringsten Berechtigungen

Wenn Sie einem AWS Dienst die Erlaubnis zur Verwendung eines KMS-Schlüssels erteilen, stellen Sie sicher, dass die Berechtigung nur für die Ressourcen gilt, auf die der Dienst in Ihrem Namen zugreifen muss. Diese Strategie der geringsten Rechte trägt dazu bei, die unbefugte Verwendung eines KMS-Schlüssels zu verhindern, wenn Anfragen zwischen AWS Diensten weitergeleitet werden.

Um eine Strategie mit den geringsten Rechten zu implementieren, empfehlen wir die Verwendung von Bedingungsschlüsseln für den AWS KMS Verschlüsselungskontext und den globalen Quell-ARN- oder Quellkonto-Bedingungsschlüssel.

Verwenden von Verschlüsselungskontext-Bedingungsschlüsseln

Die effektivste Methode zur Implementierung von Berechtigungen mit den geringsten Rechten bei der Nutzung von AWS KMS Ressourcen besteht darin, die kms:EncryptionContextKeysBedingungsschlüssel kms:EncryptionContext:context-keyoder in die Richtlinie aufzunehmen, die es den Prinzipalen ermöglicht, AWS KMS kryptografische Operationen aufzurufen. Diese Bedingungsschlüssel sind besonders effektiv, da sie die Berechtigung mit dem Verschlüsselungskontext verknüpfen, der an den Chiffretext gebunden ist, wenn die Ressource verschlüsselt wird.

Verwenden Sie Schlüssel für Bedingungen für den Verschlüsselungskontext nur, wenn es sich bei der Aktion in der Richtlinienanweisung um eine AWS KMS symmetrische kryptografische Operation handelt, die einen `EncryptionContext` Parameter benötigt, z. B. Operationen wie CreateGrantoder Decrypt. GenerateDataKey (Eine Liste der unterstützten Operationen finden Sie unter kms:EncryptionContext:context-key oder kms:EncryptionContextKeys.) Wenn Sie diese Bedingungsschlüssel verwenden, um z. B. andere Operationen zuzulassen, wird der DescribeKeyZugriff verweigert.

Legen Sie den Wert auf den Verschlüsselungskontext fest, den der Service beim Verschlüsseln der Ressource verwendet. Diese Informationen finden Sie normalerweise im Kapitel zur Sicherheit in der Service-Dokumentation. Beispielsweise identifiziert der Verschlüsselungskontext für AWS Proton die AWS Proton-Ressource und die zugehörige Vorlage. Der Verschlüsselungskontext für AWS Secrets Manager identifiziert das Geheimnis und seine Version. Der Verschlüsselungskontext für Amazon Location identifiziert den Tracker oder die Erfassung.

Die folgende Schlüsselrichtlinien-Beispielanweisung ermöglicht es Amazon Location Service, Erteilungen im Namen autorisierter Benutzer zu erstellen. Diese Richtlinienanweisung

schränkt die Berechtigung ein, indem sie die Schlüssel kms: ViaService, kms: und `kms:EncryptionContext:context-key` condition verwendetCallerAccount, um die Berechtigung an eine bestimmte Tracker-Ressource zu binden.

```
{
  "Sid": "Allow Amazon Location to create grants on behalf of authorized users",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/LocationTeam"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "geo.us-west-2.amazonaws.com",
      "kms:CallerAccount": "111122223333",
      "kms:EncryptionContext:aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:tracker/
SAMPLE-Tracker"
    }
  }
}
```

Verwenden der Bedingungsschlüssel **`aws:SourceArn`** oder **`aws:SourceAccount`**

Wenn der Prinzipal in einer Schlüsselrichtlinien-Anweisung ein AWS -Service-Prinzipal ist, empfehlen wir dringend, zusätzlich zum Bedingungsschlüssel `kms:EncryptionContext:context-key` die globalen Bedingungsschlüssel aws:SourceArn oder aws:SourceAccount zu verwenden. Die ARN- und Kontowerte sind nur dann im Autorisierungskontext enthalten, wenn eine Anfrage AWS KMS von einem anderen AWS Dienst eingeht. Diese Kombination von Bedingungen implementiert die geringsten Berechtigungen und verhindert ein potenzielles Szenario des verwirrten Stellvertreters. Dienstprinzipale werden normalerweise nicht als Prinzipale in einer wichtigen Richtlinie verwendet, aber für einige AWS Dienste, z. B. AWS CloudTrail, ist dies erforderlich.

Um die globalen Bedingungsschlüssel `aws:SourceArn` oder `aws:SourceAccount` zu verwenden, legen Sie den Wert auf den Amazon-Ressourcennamen (ARN) oder das Konto der Ressource fest, die verschlüsselt wird. In einer Schlüsselrichtlinien-Anweisung, die AWS CloudTrail die Berechtigung zum Verschlüsseln eines Trail gibt, legen Sie den Wert von `aws:SourceArn` auf den ARN des Trail fest. Nutzen Sie, wann immer möglich, den spezifischeren Wert `aws:SourceArn`. Legen Sie den Wert auf den ARN oder ein ARN-Muster mit Platzhalterzeichen fest. Wenn Sie den ARN der Ressource nicht kennen, verwenden Sie stattdessen `aws:SourceAccount`.

> **Note**
>
> Wenn ein Ressourcen-ARN Zeichen enthält, die in einer AWS KMS Schlüsselrichtlinie nicht zulässig sind, können Sie diesen Ressourcen-ARN nicht im Wert des `aws:SourceArn` Bedingungsschlüssels verwenden. Verwenden Sie stattdessen den Bedingungsschlüssel `aws:SourceAccount`. Weitere Informationen zu Dokumentenregeln für Schlüsselrichtlinien finden Sie unter Schlüsselrichtlinienformat.

In der folgenden Beispiel-Schlüsselrichtlinie ist der Prinzipal, der die Berechtigungen erhält, der AWS CloudTrail -Service-Prinzipal `cloudtrail.amazonaws.com`. Um die geringsten Berechtigungen zu implementieren, verwendet diese Richtlinie die Bedingungsschlüssel `aws:SourceArn` und `kms:EncryptionContext:`*`context-key`* Die Richtlinienanweisung ermöglicht CloudTrail die Verwendung des KMS-Schlüssels zur Generierung des Datenschlüssels, der zur Verschlüsselung eines Trails verwendet wird. Die Bedingungen `aws:SourceArn` und `kms:EncryptionContext:`*`context-key`* werden unabhängig ausgewertet. Jede Anforderung, den KMS-Schlüssel für die angegebene Produktion zu verwenden, muss beide Bedingungen erfüllen.

Um die Berechtigung des Services auf den `finance`-Trail im Beispielkonto (111122223333) und die Region `us-west-2` zu beschränken, legt diese Richtlinienanweisung den Bedingungsschlüssel `aws:SourceArn` auf den ARN eines bestimmten Trail fest. Die Bedingungsanweisung verwendet den ArnEqualsOperator, um sicherzustellen, dass jedes Element im ARN beim Abgleich unabhängig ausgewertet wird. Das Beispiel verwendet den Bedingungsschlüssel `kms:EncryptionContext:`*`context-key`* auch, um die Berechtigung auf Trail in einem bestimmten Konto und einer bestimmten Region zu beschränken.

Bevor Sie diese Schlüsselrichtlinie verwenden, ersetzen Sie die Beispiel-Konto-ID, die Region und den Trail-Namen durch gültige Werte aus Ihrem Konto.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudTrailToEncryptLogs",
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "cloudtrail.amazonaws.com"
      },
      "Action": "kms:GenerateDataKey",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:cloudtrail:us-west-2:111122223333:trail/finance"
          ]
        },
        "StringLike": {
          "kms:EncryptionContext:aws:cloudtrail:arn": [
            "arn:aws:cloudtrail:*:111122223333:trail/*"
          ]
        }
      }
    }
  ]
}
```

ABAC für AWS KMS

Die attributebasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen auf der Grundlage von Attributen definiert werden. AWS KMS unterstützt ABAC, indem es Ihnen ermöglicht, den Zugriff auf Ihre vom Kunden verwalteten Schlüssel auf der Grundlage der mit den KMS-Schlüsseln verknüpften Tags und Aliasnamen zu kontrollieren. Die Tag- und Alias-Bedingungsschlüssel, die ABAC ermöglichen, AWS KMS bieten eine leistungsstarke und flexible Möglichkeit, Principals zur Verwendung von KMS-Schlüsseln zu autorisieren, ohne Richtlinien bearbeiten oder Zuschüsse verwalten zu müssen. Sie sollten diese Funktion jedoch mit Vorsicht verwenden, damit Prinzipalen der Zugriff nicht versehentlich zugelassen oder verweigert wird.

Wenn Sie ABAC verwenden, beachten Sie, dass die Berechtigung zum Verwalten von Tags und Aliassen jetzt eine Zugriffssteuerungs-Berechtigung ist. Stellen Sie sicher, dass Sie die vorhandenen Tags und Aliasse für alle KMS-Schlüssel kennen, bevor Sie eine Richtlinie bereitstellen, die von Tags oder Aliassen abhängt. Treffen Sie angemessene Vorsichtsmaßnahmen beim Hinzufügen, Löschen und Aktualisieren von Aliassen sowie beim Markieren und Entmarkieren von Schlüsseln. Erteilen Sie Berechtigungen zum Verwalten von Tags und Aliassen nur Prinzipalen, die sie benötigen, und beschränken Sie die Tags und Aliasse, die sie verwalten können.

> **Hinweise**
>
> Achten Sie bei der Verwendung von ABAC für darauf AWS KMS, den Prinzipalen die Erlaubnis zur Verwaltung von Tags und Aliasnamen zu erteilen. Wenn Sie ein Tag oder einen Alias ändern, wird die Berechtigung für einen KMS-Schlüssel eventuell erlaubt oder verweigert. Schlüsseladministratoren, die nicht über die Berechtigung zum Ändern von Schlüsselrichtlinien oder zum Erstellen von Erteilungen verfügen, können den Zugriff auf KMS-Schlüssel steuern, wenn sie über die Berechtigung zum Verwalten von Tags oder Aliassen verfügen.
> Es kann bis zu fünf Minuten dauern, bis Tag- und Alias-Änderungen Auswirkungen auf die KMS-Schlüsselautorisierung haben. Letzte Änderungen sind möglicherweise in API-Operationen sichtbar, bevor sie sich auf die Autorisierung auswirken.
> Um den Zugriff auf einen KMS-Schlüssel basierend auf seinem Alias zu steuern, müssen Sie einen Bedingungsschlüssel verwenden. Sie können keinen Alias verwenden, um einen KMS-Schlüssel im `Resource`-Element einer Richtlinienanweisung darzustellen. Wenn ein Alias im `Resource`-Element erscheint, gilt die Richtlinienanweisung für den Alias und nicht für den zugeordneten KMS-Schlüssel.

Weitere Informationen

- Einzelheiten zur AWS KMS Unterstützung von ABAC, einschließlich Beispielen, finden Sie unter und. Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern
- Weitere allgemeine Informationen zur Verwendung von Tags zur Steuerung des Zugriffs auf AWS Ressourcen finden Sie unter Wozu dient ABAC? AWS und Steuern des Zugriffs auf AWS Ressourcen mithilfe von Ressourcen-Tags im IAM-Benutzerhandbuch.

ABAC-Bedingungsschlüssel für AWS KMS

Verwenden Sie die folgenden Bedingungsschlüssel in einer Schlüsselrichtlinie oder IAM-Richtlinie, um den Zugriff auf KMS-Schlüssel basierend auf deren Tags und Aliassen zu autorisieren.

ABAC-Bedingungsschlüssel	Beschreibung	Richtlinientyp	AWS KMS Operationen
als: ResourceTag	Tag (Schlüssel und Wert) auf dem KMS-Schlüssel entspricht dem Tag (Schlüssel und Wert) oder dem Tagmuster in der Richtlinie	Nur IAM-Richtlinie	KMS-Schlüsselressourcen-Operationen [2]
aws:RequestTag/Tag-Schlüssel	Tag (Schlüssel und Wert) in der Anforderung entspricht dem Tag (Schlüssel und Wert) oder dem Tagmuster in der Richtlinie	Wichtige Richtlinien und IAM-Richtlinien [1]	TagResource, UntagResource
aws: TagKeys	Die Tag-Schlüssel in der Anforderung entsprechen den Tag-Schlüsseln in der Richtlinie	Schlüsselrichtlinien und IAM-Richtlinien [1]	TagResource, UntagResource
km: ResourceAliases	Aliasse, die dem KMS-Schlüssel zugeordnet sind, stimmen mit den Aliassen oder Aliasmustern in der Richtlinie überein	Nur IAM-Richtlinie	KMS-Schlüsselressourcen-Operationen [2]
km: RequestAlias	Der Alias, der den KMS-Schlüssel in der Anforderung darstellt, entspricht dem Alias	Schlüsselrichtlinien und IAM-Richtlinien [1]	Kryptografische Operationen, DescribeKeyGetPublicKey

ABAC-Bedingungsschlüssel	Beschreibung	Richtlinientyp	AWS KMS Operationen
	oder den Aliasmustern in der Richtlinie.		

[1] Jeder Bedingungsschlüssel, der in einer Schlüsselrichtlinie verwendet werden kann, kann auch in einer IAM-Richtlinie verwendet werden, jedoch nur, wenn die Schlüsselrichtlinie es erlaubt.

[2]Eine KMS-Schlüsselressourcen-Operation ist eine Operation, die für einen bestimmten KMS-Schlüssel autorisiert ist. Um die KMS-Schlüsselressourcen-Operationen zu identifizieren, suchen Sie in der Tabelle mit AWS KMS -Berechtigungen nach dem Wert des KMS-Schlüssels in der `Resources`-Spalte für die Operation.

Beispielsweise können Sie diese Bedingungsschlüssel verwenden, um die folgenden Richtlinien zu erstellen.

- Eine IAM-Richtlinie mit `kms:ResourceAliases`, die die Berechtigung zur Verwendung von KMS-Schlüsseln mit einem bestimmten Alias oder Aliasmuster ermöglicht. Dies unterscheidet sich ein wenig von Richtlinien, die auf Tags basieren: Sie können zwar Aliasmuster in einer Richtlinie verwenden, aber jeder Alias muss in einer AWS-Konto UND-Region eindeutig sein. Auf diese Weise können Sie eine Richtlinie auf einen ausgewählten Satz von KMS-Schlüsseln anwenden, ohne die Schlüssel ARNs der KMS-Schlüssel in der Richtlinienerklärung aufzulisten. Um KMS-Schlüssel aus dem Satz hinzuzufügen oder zu entfernen, ändern Sie den Alias des KMS-Schlüssels.
- Eine Schlüsselrichtlinie mit `kms:RequestAlias`, die es Prinzipalen ermöglicht, einen KMS-Schlüssel in einer `Encrypt`-Operation zu nutzen, aber nur dann, wenn die `Encrypt`-Anforderung diesen Alias verwendet, um den KMS-Schlüssel zu identifizieren.
- Eine IAM-Richtlinie mit `aws:ResourceTag/`*`tag-key`*, die die Berechtigung zur Verwendung von KMS-Schlüsseln mit einem bestimmten Tag-Schlüssel und Tag-Wert verweigert. Auf diese Weise können Sie eine Richtlinie auf einen ausgewählten Satz von KMS-Schlüsseln anwenden, ohne den Schlüssel ARNs der KMS-Schlüssel in der Richtlinienerklärung aufzulisten. Um KMS-Schlüssel aus dem Satz hinzuzufügen oder zu entfernen, markieren oder entmarkieren Sie den KMS-Schlüssel.
- Eine IAM-Richtlinie mit `aws:RequestTag/`*`tag-key`*, die es Prinzipalen erlaubt, nur `"Purpose"="Test"`-Tags von KMS-Schlüsseln zu löschen.

- Eine IAM-Richtlinie mit `aws:TagKeys`, die die Berechtigung zur Markierung oder Entmarkierung eines KMS-Schlüssels mit einem `Restricted`-Tag-Schlüssel verweigert.

ABAC macht das Zugriffsmanagement flexibel und skalierbar. Sie können beispielsweise den `aws:ResourceTag/`*`tag-key`*-Bedingungsschlüssel verwenden, um eine IAM-Richtlinie zu erstellen, die es Prinzipalen erlaubt, einen KMS-Schlüssel für bestimmte Operationen nur dann zu verwenden, wenn der KMS-Schlüssel einen `Purpose=Test`-Tag hat. Die Richtlinie gilt für alle KMS-Schlüssel in allen Regionen des AWS-Konto.

Wenn sie einem Benutzer oder einer Rolle zugeordnet ist, können Prinzipale mit der folgenden IAM-Richtlinie alle vorhandenen KMS-Schlüssel mit einem`Purpose=Test`-Tag für die angegebenen Operationen nutzen. Um diesen Zugriff auf neue oder vorhandene KMS-Schlüssel zu gewähren, müssen Sie die Richtlinie nicht ändern. Fügen Sie einfach das `Purpose=Test`-Tag de KMS-Schlüsseln an. Um diesen Zugriff von KMS-Schlüsseln mit einem `Purpose=Test`-Tag zu entfernen, bearbeiten oder löschen Sie das Tag.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Test"
        }
      }
    }
  ]
}
```

Wenn Sie diese Funktion verwenden, sollten Sie jedoch vorsichtig sein, wenn Sie Tags und Aliasse verwalten. Das Hinzufügen, Ändern oder Löschen eines Tags oder Aliasses kann versehentlich den Zugriff auf einen KMS-Schlüssel zulassen oder verweigern. Schlüsseladministratoren, die nicht über die Berechtigung zum Ändern von Schlüsselrichtlinien oder zum Erstellen von Erteilungen verfügen, können den Zugriff auf KMS-Schlüssel steuern, wenn sie über die Berechtigung zum Verwalten von Tags oder Aliassen verfügen. Um dieses Risiko zu mindern, sollten Sie Berechtigungen zum Verwalten von Tags und Aliassen beschränken. Sie können es beispielsweise nur ausgewählten Prinzipalen erlauben, `Purpose=Test`-Tags zu verwalten. Details dazu finden Sie unter Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern und Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern.

Tags oder Aliasse?

AWS KMS unterstützt ABAC mit Tags und Aliasen. Beide Optionen bieten eine flexible, skalierbare Strategie zur Zugriffssteuerung, unterscheiden sich jedoch geringfügig voneinander.

Sie können sich entscheiden, Tags oder Aliase zu verwenden, die auf Ihren speziellen AWS Verwendungsmustern basieren. Wenn Sie beispielsweise den meisten Administratoren bereits Markierungs-Berechtigungen erteilt haben, ist es möglicherweise einfacher, eine Autorisierungsstrategie basierend auf Aliasse zu steuern. Oder, wenn Sie nahe am Kontingent für Aliasse pro KMS-Schlüssel sind, verwenden Sie möglicherweise lieber eine Autorisierungs-Strategie, die auf Tags basiert.

Die folgenden Nutzen sind von allgemeinem Interesse.

Vorteile einer Tag-basierten Zugriffskontrolle

- Derselbe Autorisierungsmechanismus für verschiedene AWS Ressourcentypen.

  Sie können denselben Tag- oder Tag-Schlüssel verwenden, um den Zugriff auf mehrere Ressourcentypen zu steuern, z. B. einen Amazon-RDS (Amazon Relational Database Service)-Cluster, ein Amazon-EBS (Amazon Elastic Block Store)-Volume und einen KMS-Schlüssel. Diese Funktion ermöglicht verschiedene Autorisierungsmodelle, die flexibler sind als herkömmliche rollenbasierte Zugriffskontrolle.
- Autorisieren des Zugriffs auf eine Gruppe von KMS-Schlüsseln

  Sie können Tags verwenden, um den Zugriff auf eine Gruppe von KMS-Schlüsseln in demselben AWS-Konto und in derselben Region zu verwalten. Weisen Sie den ausgewählten KMS-Schlüsseln denselben Tag- oder Tag-Schlüssel zu. Erstellen Sie dann eine einfache easy-to-maintain

Richtlinienerklärung, die auf dem Tag oder dem Tag-Schlüssel basiert. Um einen KMS-Schlüssel aus Ihrer Autorisierungsgruppe hinzuzufügen oder zu entfernen, fügen Sie das Tag hinzu oder entfernen Sie es. Sie müssen die Richtlinie nicht bearbeiten.

Vorteile einer Alias-basierten Zugriffskontrolle

- Autorisieren Sie den Zugriff auf kryptografische Operationen basierend auf Aliassen.

  Die meisten anforderungsbasierten Richtlinienbedingungen für Attribute, einschließlich aws:RequestTag/tag-key, betreffen nur Operationen, die das Attribut hinzufügen, bearbeiten oder löschen. Der RequestAlias Bedingungsschlüssel kms: steuert jedoch den Zugriff auf kryptografische Operationen auf der Grundlage des Alias, der zur Identifizierung des KMS-Schlüssels in der Anfrage verwendet wird. Sie können beispielsweise einem Prinzipal die Berechtigung erteilen, einen KMS-Schlüssel in einer `Encrypt`-Operation zu nutzen, aber nur dann, wenn der Wert des `KeyId`-Parameters `alias/restricted-key-1` ist. Diese Bedingung zu erfüllen, erfordert Folgendes:

  - Der KMS-Schlüssel muss diesem Alias zugeordnet sein.
  - Die Anforderung muss den Alias verwenden, um den KMS-Schlüssel zu identifizieren.
  - Der Prinzipal muss über die Berechtigung zur Verwendung des KMS-Schlüssels verfügen, sofern die `kms:RequestAlias`-Bedingung es erlaubt.

  Dies ist besonders nützlich, wenn Ihre Anwendungen häufig Aliasnamen oder Alias verwenden ARNs , um auf KMS-Schlüssel zu verweisen.

- Stellen Sie sehr eingeschränkte Berechtigungen bereit.

  Ein Alias muss in einer AWS-Konto UND-Region eindeutig sein. Daher kann es wesentlich restriktiver sein, Prinzipalen Zugriff auf einen KMS-Schlüssel basierend auf einem Alias zu gewähren, als ihnen Zugriff basierend auf einem Tag zu gewähren. Im Gegensatz zu Aliassen können Tags mehreren KMS-Schlüsseln in demselben Konto und derselben Region zugewiesen werden. Wenn Sie auswählen, können Sie ein Aliasmuster verwenden, z. B.`alias/test*`, um Prinzipalen Zugriff auf eine Gruppe von KMS-Schlüsseln in demselben Konto und derselben Region zu gewähren. Allerdings ermöglicht das Erlauben oder Verweigern des Zugriffs auf einen bestimmten Alias eine sehr strenge Kontrolle über KMS-Schlüssel.

Problembehandlung bei ABAC für AWS KMS

Die Steuerung des Zugriffs auf KMS-Schlüssel basierend auf ihren Tags und Aliassen ist bequem und leistungsstark. Es ist jedoch anfällig für einige vorhersehbare Fehler, die Sie verhindern möchten.

Zugriff aufgrund von Tag-Änderung geändert

Wenn ein Tag gelöscht wird oder sein Wert geändert wird, wird Prinzipalen, dessen Zugriff auf einen KMS-Schlüssel nur auf diesem Tag basiert, der Zugriff auf den KMS-Schlüssel verweigert. Dies kann auch passieren, wenn ein Tag, das in einer Zugriffsverweigerungs-Richtlinienanweisung enthalten ist, einem KMS-Schlüssel hinzugefügt wird. Das Hinzufügen eines richtlinienbezogenen Tags zu einem KMS-Schlüssel kann Prinzipalen den Zugriff erlauben, denen der Zugriff auf einen KMS-Schlüssel verweigert werden soll.

Angenommen, ein Prinzipal hat Zugriff auf einen KMS-Schlüssel basierend auf dem `Project=Alpha`-Tag, z. B. die Berechtigung, die in der folgenden IAM-Richtlinienanweisung bereitgestellt wird.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:ap-southeast-1:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "Alpha"
        }
      }
    }
  ]
}
```

Wenn das Tag aus diesem KMS-Schlüssel gelöscht wird oder der Tag-Wert geändert wird, hat der Prinzipal keine Berechtigung mehr, den KMS-Schlüssel für die angegebenen Operationen zu verwenden. Dies kann deutlich werden, wenn der Principal versucht, Daten in einem AWS Dienst zu lesen oder zu schreiben, der einen vom Kunden verwalteten Schlüssel verwendet. Um die Änderung des Kennworts nachzuverfolgen, überprüfen Sie Ihre CloudTrail Logs auf UntagResource Einträge. TagResource

Um den Zugriff wiederherzustellen, ohne die Richtlinie zu aktualisieren, ändern Sie die Tags auf dem KMS-Schlüssel. Diese Aktion hat minimale Auswirkungen außer für einen kurzen Zeitraum, während sie in allen Bereichen von AWS KMS in Kraft tritt. Um einen Fehler wie diesen zu vermeiden, erteilen Sie Berechtigungen zum Markieren und Entmarkieren nur an Prinzipale, die sie benötigen, und beschränken Sie ihrer Markierungs-Berechtigungen auf Tags, die sie verwalten müssen. Bevor Sie einen Tag ändern, durchsuchen Sie Richtlinien, um den Zugriff zu erkennen, der vom Tag abhängt, und erhalten Sie KMS-Schlüssel in allen Regionen, die das Tag enthalten. Sie könnten erwägen, einen CloudWatch Amazon-Alarm zu erstellen, wenn bestimmte Tags geändert werden.

Änderung des Zugriffs aufgrund von Aliasänderungen

Wenn ein Alias gelöscht oder einem anderen KMS-Schlüssel zugeordnet wird, wird Prinzipalen, dessen Zugriff auf den KMS-Schlüssel nur auf diesem Alias basiert, der Zugriff auf den KMS-Schlüssel verweigert. Dies kann auch passieren, wenn ein Tag, das in einer Zugriffsverweigerungs-Richtlinienanweisung enthalten ist, einem KMS-Schlüssel hinzugefügt wird. Das Hinzufügen eines richtlinienbezogenen Tags zu einem KMS-Schlüssel kann Prinzipalen den Zugriff erlauben, denen der Zugriff auf einen KMS-Schlüssel verweigert werden soll.

In der folgenden IAM-Richtlinienerklärung wird beispielsweise der ResourceAliases Bedingungsschlüssel kms: verwendet, um den Zugriff auf KMS-Schlüssel in verschiedenen Regionen des Kontos mit einem der angegebenen Aliase zu ermöglichen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:List*",
```

```
        "kms:Describe*",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "kms:ResourceAliases": [
            "alias/ProjectAlpha",
            "alias/ProjectAlpha_Test",
            "alias/ProjectAlpha_Dev"
          ]
        }
      }
    }
  ]
}
```

Um die Aliasänderung nachzuverfolgen, überprüfen Sie Ihre CloudTrail Logs auf CreateAliasEinträge UpdateAlias, und DeleteAlias.

Um den Zugriff wiederherzustellen, ohne die Richtlinie zu aktualisieren, ändern Sie den Alias, der dem KMS-Schlüssel zugeordnet ist. Da jeder Alias nur einem KMS-Schlüssel in einem Konto und einer Region zugeordnet werden kann, ist die Verwaltung von Aliassen etwas schwieriger als die Verwaltung von Tags. Das Wiederherstellen des Zugriffs auf einige Prinzipale auf einem KMS-Schlüssel kann denselben oder anderen Prinzipalen den Zugriff auf einen anderen KMS-Schlüssel verweigern.

Um diesen Fehler zu vermeiden, erteilen Sie Alias-Verwaltungs-Berechtigungen nur für Prinzipale, die diese benötigen, und schränken Sie ihre Alias-Verwaltungs-Berechtigungen auf Aliasse ein, die sie verwalten müssen. Suchen Sie vor dem Aktualisieren oder Löschen eines Aliasses Richtlinien, um den Zugriff zu erkennen, der vom Alias abhängt, und finden Sie die KMS-Schlüssel in allen Regionen, die dem Alias zugeordnet sind.

Zugriff aufgrund eines Aliaskontingents verweigert

Benutzer, die aufgrund einer KMS: ResourceAliases -Bedingung zur Verwendung eines KMS-Schlüssels autorisiert sind, erhalten eine `AccessDenied` Ausnahme, wenn der KMS-Schlüssel die Standardaliase pro KMS-Schlüsselkontingent für dieses Konto und diese Region überschreitet.

Um den Zugriff wiederherzustellen, löschen Sie Aliasse, die dem KMS-Schlüssel zugeordnet sind, damit er dem Kontingent entspricht. Oder verwenden Sie einen alternativen Mechanismus, um Benutzern Zugriff auf den KMS-Schlüssel zu gewähren.

Verzögerte Autorisierungsänderung

Es kann bis zu fünf Minuten dauern, bis sich die Änderungen, die Sie an Tags und Aliassen vornehmen, auf die Autorisierung von KMS-Schlüsseln auswirken. Infolgedessen kann eine Tag- oder Aliasänderung in den Antworten von API-Operationen widergespiegelt werden, bevor sie sich auf die Autorisierung auswirken. Diese Verzögerung ist wahrscheinlich länger als die kurze eventuelle Konsistenzverzögerung, von der die meisten AWS KMS Vorgänge betroffen sind.

Beispielsweise können Sie eine IAM-Richtlinie verwenden, die es bestimmten Prinzipalen erlaubt, einen KMS-Schlüssel mit einem `"Purpose"="Test"`-Tag zu nutzen. Dann fügen Sie das `"Purpose"="Test"`-Tag einem KMS-Schlüssel hinzu. Obwohl der TagResourceVorgang abgeschlossen ist und die ListResourceTagsAntwort bestätigt, dass das Tag dem KMS-Schlüssel zugewiesen ist, haben die Prinzipale möglicherweise bis zu fünf Minuten lang keinen Zugriff auf den KMS-Schlüssel.

Um Fehler zu vermeiden, bauen Sie diese erwartete Verzögerung in Ihren Code ein.

Fehlgeschlagene Anforderungen aufgrund von Alias-Aktualisierungen

Sie können Aliasse auch aktualisieren, wodurch ein vorhandenes Alias einem anderen KMS-Schlüssel zugeordnet wird.

Entschlüsselungen und ReEncryptAnfragen, die den Aliasnamen oder Alias-ARN angeben, schlagen möglicherweise fehl, da der Alias jetzt mit einem KMS-Schlüssel verknüpft ist, der den Chiffretext nicht verschlüsselt hat. Diese Situation gibt in der Regel `IncorrectKeyException` oder `NotFoundException` zurück. Oder wenn die Anforderung keinen `KeyId`- oder `DestinationKeyId`-Parameter enthält, schlägt die Operation möglicherweise mit einer `AccessDenied`-Ausnahme fehl, da der Aufrufer keinen Zugriff mehr auf den KMS-Schlüssel hat, der den Chiffretext verschlüsselt hat.

Sie können die Änderung nachverfolgen, indem Sie in den CloudTrail Protokollen nach CreateAliasUpdateAlias, und nach Protokolleinträgen suchen. DeleteAlias Sie können auch den Wert des `LastUpdatedDate` Felds in der ListAliasesAntwort verwenden, um eine Änderung zu erkennen.

Die folgende Beispielantwort zeigt ListAliasesbeispielsweise, dass der `ProjectAlpha_Test` Alias in der `kms:ResourceAliases` Bedingung aktualisiert wurde. Daher verlieren die Prinzipale, dessen

Zugriff auf dem Alias basiert, den Zugriff auf den zuvor zugeordneten KMS-Schlüssel. Stattdessen haben sie Zugriff auf den neu zugeordneten KMS-Schlüssel.

```
$ aws kms list-aliases --query 'Aliases[?starts_with(AliasName, `alias/ProjectAlpha`)]'

{
    "Aliases": [
        {
            "AliasName": "alias/ProjectAlpha_Test",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ProjectAlpha_Test",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1566518783.394,
            "LastUpdatedDate": 1605308931.903
        },
        {
            "AliasName": "alias/ProjectAlpha_Restricted",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/
ProjectAlpha_Restricted",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1553410800.010,
            "LastUpdatedDate": 1553410800.010
        }
    ]
}
```

Die Lösung für diese Änderung ist nicht einfach. Sie können den Alias erneut aktualisieren, um ihn dem ursprünglichen KMS-Schlüssel zuzuordnen. Bevor Sie jedoch handeln, müssen Sie die Auswirkungen dieser Änderung auf den aktuell zugeordneten KMS-Schlüssel berücksichtigen. Wenn Prinzipale den letzteren KMS-Schlüssel in kryptografischen Operationen verwendet haben, müssen sie möglicherweise weiterhin darauf zugreifen. In diesem Fall sollten Sie die Richtlinie aktualisieren, um sicherzustellen, dass Prinzipale über die Berechtigung verfügen, beide KMS-Schlüssel zu verwenden.

Sie können einen Fehler wie diesen verhindern: Bevor Sie einen Alias aktualisieren, suchen Sie Richtlinien, um den Zugriff zu erkennen, der vom Alias abhängt. Rufen Sie dann KMS-Schlüssel in allen Regionen ab, die dem Alias zugeordnet sind. Erteilen Sie Alias-Verwaltungs-Berechtigungen nur für Prinzipale, die diese benötigen, und schränken Sie ihre Alias-Verwaltungs-Berechtigungen auf Aliasse ein, die sie verwalten müssen.

RBAC für AWS KMS

Bei der rollenbasierten Zugriffskontrolle (RBAC) handelt es sich um eine Autorisierungsstrategie, die Benutzern nur die für die Erfüllung ihrer Aufgaben erforderlichen Berechtigungen gewährt, mehr nicht. AWS KMS unterstützt RBAC, indem es Ihnen ermöglicht, den Zugriff auf Ihre Schlüssel zu kontrollieren, indem Sie innerhalb von Schlüsselrichtlinien detaillierte Berechtigungen für die Schlüsselverwendung festlegen. In Schlüsselrichtlinien werden eine Ressource, eine Aktion, eine Wirkung, ein Hauptprinzip und optionale Bedingungen für die Gewährung des Zugriffs auf Schlüssel festgelegt.

Um RBAC zu implementieren, empfehlen wir AWS KMS, die Berechtigungen für Schlüsselbenutzer und Schlüsseladministratoren voneinander zu trennen.

Key users

Das folgende Beispiel für eine wichtige Richtlinie ermöglicht es der `ExampleUserRole` IAM-Rolle, den KMS-Schlüssel zu verwenden.

```
{
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws: iam::111122223333:role/ExampleUserRole"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
  }
```

Ihre Hauptbenutzer benötigen möglicherweise weniger Berechtigungen als der Benutzer in diesem Beispiel. Weisen Sie nur die Berechtigungen zu, die der Benutzer benötigt. Verwenden Sie die folgenden Fragen, um die Berechtigungen weiter zu verfeinern.

- Welche IAM-Prinzipale (Rollen oder Benutzer) benötigen Zugriff auf den Schlüssel?

- Welche Aktionen muss jeder Principal mit dem Schlüssel ausführen? Benötigt der Principal beispielsweise nur die Berechtigungen Verschlüsseln und Signieren?
- Ist der Benutzer ein Mensch oder ein AWS Dienst? Wenn es sich um einen AWS Dienst handelt, können Sie den Bedingungsschlüssel verwenden, um die Schlüsselverwendung auf einen bestimmten AWS Dienst zu beschränken.

Key administrators

Das folgende Beispiel für eine wichtige Richtlinie ermöglicht es der `ExampleAdminRole` IAM-Rolle, den KMS-Schlüssel zu verwalten.

```
{
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws: iam::111122223333:role/ExampleAdminRole"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
    }
```

In diesem Beispiel benötigen Ihre Schlüsseladministratoren möglicherweise weniger Berechtigungen als der Administrator. Weisen Sie Ihren Schlüsseladministratoren nur die Berechtigungen zu, die sie benötigen.

Erteilen Sie Benutzern nur die Berechtigungen, die sie zur Erfüllung ihrer Rollen benötigen. Die Berechtigungen eines Benutzers können variieren, je nachdem, ob der Schlüssel in Test- oder Produktionsumgebungen verwendet wird. Wenn Sie in bestimmten Umgebungen, die nicht zur Produktion gehören, weniger restriktive Berechtigungen verwenden, implementieren Sie einen Prozess, um die Richtlinien zu testen, bevor sie für die Produktion freigegeben werden.

Weitere Informationen

- IAM-Identitäten (Benutzer, Benutzergruppen und Rollen)
- Arten der Zugriffskontrolle

Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben

Sie können Benutzern oder Rollen in einer anderen AWS-Konto Umgebung die Verwendung eines KMS-Schlüssels in Ihrem Konto gestatten. Der kontoübergreifende Zugriff erfordert die Berechtigung in der Schlüsselrichtlinie des KMS-Schlüssels und in einer IAM-Richtlinie im Konto des externen Benutzers.

Die kontenübergreifende Berechtigung gilt nur für die folgenden Operationen:

- Kryptografische Operationen
- CreateGrant
- DescribeKey
- GetKeyRotationStatus
- GetPublicKey
- ListGrants
- RetireGrant
- RevokeGrant

Wenn Sie einem Benutzer in einem anderen Konto die Berechtigung für andere Operationen erteilen, haben diese Berechtigungen keine Auswirkungen. Wenn Sie beispielsweise einem Prinzipal in einem anderen Konto die ListKeys Berechtigung kms: in einer IAM-Richtlinie oder kms: ScheduleKeyDeletion -Berechtigung für einen KMS-Schlüssel in einer Schlüsselrichtlinie erteilen,

schlagen die Versuche des Benutzers, diese Operationen auf Ihren Ressourcen aufzurufen, immer noch fehl.

Einzelheiten zur Verwendung von KMS-Schlüsseln in verschiedenen Konten für AWS KMS Operationen finden Sie in der Spalte Kontoübergreifende Verwendung unter und. AWS KMS Berechtigungen Verwenden von KMS-Schlüsseln in anderen Konten Es gibt auch einen Abschnitt über kontenübergreifende Nutzung in jeder API-Beschreibung in der AWS Key Management Service - API-Referenz.

> **Warning**
>
> Seien Sie vorsichtig, wenn Sie Prinzipalen Berechtigungen zur Verwendung Ihrer KMS-Schlüssel erteilen. Wenn möglich, befolgen Sie dem Prinzip der geringsten Berechtigung. Geben Sie Benutzern nur Zugriff auf die KMS-Schlüssel, die sie für die erforderlichen Vorgänge benötigen.
> Seien Sie auch vorsichtig bei der Verwendung eines unbekannten KMS-Schlüssels, insbesondere eines KMS-Schlüssels in einem anderen Konto. Bösartige Benutzer geben Ihnen möglicherweise die Berechtigung, ihren KMS-Schlüssel zu verwenden, um Informationen über Sie oder Ihr Konto abzurufen.
> Weitere Informationen zur Verwendung von Richtlinien zum Steuern des Zugriffs auf die Ressourcen in Ihrem Konto finden Sie unter Bewährte Methoden für IAM-Richtlinien.

Um Benutzern und Rollen in einem anderen Konto die Berechtigung zur Verwendung eines KMS-Schlüssels zu erteilen, müssen Sie zwei verschiedene Arten von Richtlinien verwenden:

- Die Schlüsselrichtlinie für den KMS-Schlüssel muss dem externen Konto (oder den Benutzern und Rollen im externen Konto) die Berechtigung zur Verwendung des KMS-Schlüssels erteilen. Die Schlüsselrichtlinie ist in dem Konto definiert, in dem der KMS-Schlüssel hinterlegt ist.
- IAM-Richtlinien im externen Konto müssen die Schlüsselrichtlinien-Berechtigungen an seine Benutzer und Rollen delegieren. Diese Richtlinien werden im externen Konto festgelegt und erteilen Berechtigungen für Benutzer und Rollen in diesem Konto.

Die Schlüsselrichtlinie bestimmt, wer Zugriff auf den KMS-Schlüssel haben kann. Die IAM-Richtlinie bestimmt, wer Zugriff auf den KMS-Schlüssel hat. Weder die Schlüsselrichtlinie noch die IAM-Richtlinie allein sind ausreichend – Sie müssen beides ändern.

Um die Schlüsselrichtlinie zu bearbeiten, können Sie die Richtlinienansicht in den PutKeyPolicyOperationen AWS Management Console oder CreateKeyoder verwenden.

Hilfe zum Bearbeiten von IAM-Richtlinien finden Sie unter Verwenden von IAM-Richtlinien mit AWS KMS.

Ein Beispiel, das zeigt, wie die Schlüsselrichtlinie und IAM-Richtlinien zusammenarbeiten, um die Verwendung eines KMS-Schlüssel in einem anderen Konto zu ermöglichen, finden Sie unter Beispiel 2: Der Benutzer übernimmt die Rolle mit der Erlaubnis, einen KMS-Schlüssel in einem anderen System zu verwenden AWS-Konto.

Sie können die resultierenden kontenübergreifenden AWS KMS Operationen mit dem KMS-Schlüssel in Ihren AWS CloudTrail Protokollen einsehen. Operationen, die KMS-Schlüssel in anderen Konten verwenden, werden sowohl im Konto des Anrufers als auch im Konto des KMS-Schlüsselbesitzers protokolliert.

Themen

- Schritt 1: Hinzufügen einer Schlüsselrichtlinienanweisung im lokalen Konto
- Schritt 2: Hinzufügen von IAM-Richtlinien im externen Konto
- Zulassen der Verwendung externer KMS-Schlüssel mit AWS-Services
- Verwenden von KMS-Schlüsseln in anderen Konten

> **Note**
>
> Die Beispiele in diesem Thema zeigen, wie Sie eine Schlüsselrichtlinie und eine IAM-Richtlinie zusammen verwenden, um den Zugriff auf einen KMS-Schlüssel bereitzustellen und einzuschränken. Diese allgemeinen Beispiele sollen nicht die Berechtigungen darstellen, die für einen bestimmten KMS-Schlüssel AWS-Service erforderlich sind. Informationen zu den Berechtigungen, die für einen AWS-Service erforderlich sind, finden Sie im Thema Verschlüsselung in der Servicedokumentation.

Schritt 1: Hinzufügen einer Schlüsselrichtlinienanweisung im lokalen Konto

Die Schlüsselrichtlinie für einen KMS-Schlüssel ist der erste Bestimmungsfaktor dafür, wer auf den KMS-Schlüssel zugreifen kann und welche Operationen sie ausführen können. Die Schlüsselrichtlinie

ist immer in dem Konto, in dem der KMS-Schlüssel hinterlegt ist. Anders als IAM-Richtlinien geben Schlüsselrichtlinien keine Ressource an. Die Ressource ist der KMS-Schlüssel, der die Schlüsselrichtlinie zugeordnet ist. Wenn Sie eine kontenübergreifende Berechtigung erteilen, muss die Schlüsselrichtlinie für den KMS-Schlüssel dem externen Konto (oder den Benutzern und Rollen im externen Konto) die Berechtigung erteilen, den KMS-Schlüssel zu verwenden.

Um einem externen Konto die Berechtigung zur Verwendung des KMS-Schlüssels zu erteilen, fügen Sie der Schlüsselrichtlinie eine Anweisung hinzu, die das externe Konto angibt. Geben Sie im `Principal`-Element der Schlüsselrichtlinie den Amazon-Ressourcennamen (ARN) des externen Kontos ein.

Wenn Sie ein externes Konto in einer Schlüsselrichtlinie angeben, können IAM-Administratoren im externen Konto IAM-Richtlinien verwenden, um diese Berechtigungen an alle Benutzer und Rollen im externen Konto zu delegieren. Sie können auch entscheiden, welche der in der Schlüsselrichtlinie angegebenen Aktionen die Benutzer und Rollen ausführen dürfen.

Berechtigungen, die dem externen Konto und seinen Prinzipalen erteilt werden, sind nur wirksam, wenn das externe Konto in der Region aktiviert ist, die den KMS-Schlüssel und seine Schlüsselrichtlinie hostet. Informationen zu Regionen, die standardmäßig nicht aktiviert sind („Opt-In-Regionen“), finden Sie unter Verwalten von AWS-Regionen in Allgemeine AWS-Referenz.

Angenommen, Sie möchten Konto `444455556666` erlauben, einen symmetrischen KMS-Schlüssel im Konto `111122223333` zu verwenden. Fügen Sie dazu der Schlüsselrichtlinie für den KMS-Schlüssel in Konto `111122223333` eine Schlüsselrichtlinie wie die im folgenden Beispiel hinzu. Diese Richtlinienanweisung erteilt dem externen Konto `444455556666` die Berechtigung, den KMS-Schlüssel in kryptografischen Operationen für KMS-Schlüssel mit symmetrischer Verschlüsselung zu verwenden.

> **Note**
>
> Das folgende Beispiel stellt ein Beispiel für eine Schlüsselrichtlinie für die gemeinsame Nutzung eines KMS-Schlüssels mit einem anderen Konto dar. Ersetzen Sie die Werte `Sid`, `Principal` und `Action` im Beispiel durch gültige Werte für die vorgesehene Verwendung Ihres KMS-Schlüssels.

```
{
    "Sid": "Allow an external account to use this KMS key",
```

```
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::444455556666:root"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

Anstatt dem externen Konto die Berechtigung zu erteilen, können Sie bestimmte externe Benutzer und Rollen in der Schlüsselrichtlinie angeben. Diese Benutzer und Rollen können den KMS-Schlüssel jedoch erst verwenden, wenn IAM-Administratoren im externen Konto ihren Identitäten die richtigen IAM-Richtlinien anfügen. Die IAM-Richtlinien können allen oder einer Teilmenge der externen Benutzer und Rollen, die in der Schlüsselrichtlinie angegeben sind, die Berechtigung erteilen. Und sie können alle oder eine Teilmenge der Aktionen zulassen, die in der Schlüsselrichtlinie angegeben sind.

Die Angabe von Identitäten in einer Schlüsselrichtlinie beschränkt die Berechtigungen, die IAM-Administratoren im externen Konto bereitstellen können. Die Richtlinienverwaltung mit zwei Konten wird jedoch komplexer. Angenommen, Sie müssen einen Benutzer oder eine Rolle hinzufügen. Sie müssen diese Identität der Schlüsselrichtlinie in dem Konto hinzufügen, das den KMS-Schlüssel besitzt, und IAM-Richtlinien im Konto der Identität erstellen.

Um bestimmte externe Benutzer oder Rollen in einer Schlüsselrichtlinie anzugeben, geben Sie im `Principal`-Element den Amazon-Ressourcennamen (ARN) eines Benutzers oder einer Rolle im externen Konto ein.

Die folgende Schlüsselrichtlinienanweisung erlaubt es beispielsweise `ExampleRole` im Konto `444455556666`, einen KMS-Schlüssel im Konto `111122223333` zu verwenden. Diese Schlüssel-Richtlinienanweisung erteilt dem externen Konto `444455556666` die Berechtigung, den KMS-Schlüssel in kryptografischen Operationen für KMS-Schlüssel mit symmetrischer Verschlüsselung zu verwenden.

> **Note**
>
> Das folgende Beispiel stellt ein Beispiel für eine Schlüsselrichtlinie für die gemeinsame Nutzung eines KMS-Schlüssels mit einem anderen Konto dar. Ersetzen Sie die Werte `Sid`, `Principal` und `Action` im Beispiel durch gültige Werte für die vorgesehene Verwendung Ihres KMS-Schlüssels.

```
{
    "Sid": "Allow an external account to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::444455556666:role/ExampleRole"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

> **Note**
>
> Setzen Sie den Prinzipal nicht auf ein Sternchen (*) in einer Schlüsselrichtlinienanweisung, die Berechtigungen erlaubt, es sei denn, Sie verwenden Bedingungen, um die Schlüsselrichtlinie einzuschränken. Ein Sternchen gibt jede Identität in jeder AWS-Konto Berechtigung zur Verwendung des KMS-Schlüssels an, sofern dies nicht in einer anderen Richtlinienanweisung ausdrücklich verweigert wird. Benutzer in anderen Ländern AWS-Konten können Ihren KMS-Schlüssel immer dann verwenden, wenn sie über entsprechende Berechtigungen in ihrem eigenen Konto verfügen.

Sie müssen auch entscheiden, welche Berechtigungen Sie dem externen Konto erteilen möchten. Sie können beispielsweise Benutzern die Berechtigung zum Entschlüsseln, aber nicht zum Verschlüsseln erteilen, oder die Berechtigung zum Anzeigen des KMS-Schlüssels, aber nicht zur Verwendung. Eine Liste der Berechtigungen für KMS-Schlüssel finden Sie unter AWS KMS Berechtigungen.

Festlegung der Schlüsselrichtlinie beim Erstellen eines KMS-Schlüssels

Wenn Sie den CreateKeyVorgang zum Erstellen eines KMS-Schlüssels verwenden, können Sie seinen `Policy` Parameter verwenden, um eine Schlüsselrichtlinie anzugeben, die einem externen Konto oder externen Benutzern und Rollen die Berechtigung zur Verwendung des KMS-Schlüssels erteilt.

Wenn Sie einen KMS-Schlüssel in der erstellen AWS Management Console, erstellen Sie auch die zugehörige Schlüsselrichtlinie. Wenn Sie Identitäten im Abschnitt Key Administrators (Schlüsseladministratoren) und Key Users (Schlüsselbenutzer) auswählen, fügt AWS KMS Richtlinienanweisungen für diese Identitäten zur Schlüsselrichtlinie des KMS-Schlüssels hinzu. Im Abschnitt Key Users (Schlüsselbenutzer) können Sie auch externe Konten als Schlüsselbenutzer hinzufügen.

Wenn Sie die Konto-ID eines externen Kontos eingeben, AWS KMS fügt der Schlüsselrichtlinie zwei Anweisungen hinzu. Diese Aktion wirkt sich nur auf die Schlüsselrichtlinie aus. Benutzer und Rollen im externen Konto können den KMS-Schlüssel erst verwenden, wenn Sie IAM-Richtlinien anfügen, um ihnen einige oder alle dieser Berechtigungen zu erteilen.

Die erste Schlüsselrichtlinienanweisung erteilt dem externen Konto die Berechtigung, den KMS-Schlüssel in kryptografischen Operationen zu verwenden. Die zweite wichtige Richtlinienanweisung ermöglicht es dem externen Konto, Zuweisungen für den KMS-Schlüssel zu erstellen, anzuzeigen und zu widerrufen, aber nur, wenn die Anfrage von einem AWS Dienst kommt, der in integriert ist AWS KMS. Diese Berechtigungen ermöglichen anderen AWS Diensten, die Benutzerdaten verschlüsseln, die Verwendung des KMS-Schlüssels. Diese Berechtigungen sind für KMS-Schlüssel konzipiert, die Benutzerdaten in Diensten verschlüsseln AWS

Schritt 2: Hinzufügen von IAM-Richtlinien im externen Konto

Die Schlüsselrichtlinie im Konto, dem der KMS-Schlüssel gehört, legt den gültigen Bereich für Berechtigungen fest. Benutzer und Rollen im externen Konto können den KMS-Schlüssel erst verwenden, wenn Sie IAM-Richtlinien anfügen, die diese Berechtigungen delegieren, oder Erteilungen verwenden, um den Zugriff auf den KMS-Schlüssel zu verwalten. Die IAM-Richtlinien werden im externen Konto festgelegt.

Wenn die Schlüsselrichtlinie dem externen Konto die Berechtigung erteilt, können Sie jedem Benutzer oder jeder Rolle im Konto IAM-Richtlinien zuweisen. Aber wenn die Schlüsselrichtlinie den angegebenen Benutzern oder Rollen die Berechtigung erteilt, kann die IAM-Richtlinie diese

Berechtigungen nur allen oder einer Teilmenge der angegebenen Benutzer und Rollen erteilen. Wenn eine IAM-Richtlinie anderen externen Benutzern oder Rollen den Zugriff auf KMS-Schlüssel gewährt, hat dies keine Auswirkungen.

Die Schlüsselrichtlinie beschränkt auch die Aktionen in der IAM-Richtlinie. Die IAM-Richtlinie kann alle oder eine Teilmenge der Aktionen delegieren, die in der Schlüsselrichtlinie angegeben sind. Wenn die IAM-Richtlinie Aktionen auflistet, die nicht in der Schlüsselrichtlinie angegeben sind, sind diese Berechtigungen nicht wirksam.

Die folgende IAM-Beispielrichtlinie erlaubt es dem Prinzipal, den KMS-Schlüssel im Konto `111122223333` für kryptografische Operationen zu verwenden. Um diese Berechtigung Benutzern und Rollen im Konto `444455556666` zu erteilen, fügen Sie die Richtlinie den Benutzern oder Rollen im Konto `444455556666` an.

> **Note**
>
> Das folgende Beispiel stellt eine IAM-Richtlinie für die gemeinsame Nutzung eines KMS-Schlüssels mit einem anderen Konto dar. Ersetzen Sie die Werte `Sid`, `Resource` und `Action` im Beispiel durch gültige Werte für die vorgesehene Verwendung Ihres KMS-Schlüssels.

```
{
    "Sid": "AllowUseOfKeyInAccount111122223333",
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:GenerateDataKey*",
      "kms:DescribeKey"
    ],
    "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Beachten Sie die folgenden Details über diese Richtlinie:

- Im Gegensatz zu Schlüsselrichtlinien enthalten IAM-Richtlinienanweisungen das `Principal`-Element nicht. In IAM-Richtlinien ist der Prinzipal die Identität, der die Richtlinie angefügt ist.

- Das `Resource`-Element in der IAM-Richtlinie identifiziert den KMS-Schlüssel, den der Prinzipal verwenden kann. Um einen KMS-Schlüssel anzugeben, fügen Sie dessen Schüssel-ARN dem `Resource`-Element hinzu.
- Sie können mehrere KMS-Schlüssel im `Resource`-Element angeben. Wenn Sie jedoch keine bestimmten KMS-Schlüssel im `Resource`-Element angeben, können Sie versehentlich Zugriff auf mehr KMS-Schlüssel gewähren, als Sie beabsichtigen.
- Damit der externe Benutzer den KMS-Schlüssel mit AWS -Services verwenden kann, die in AWS KMS integriert sind, müssen Sie möglicherweise Berechtigungen zur Schlüsselrichtlinie oder der IAM-Richtlinie hinzufügen. Details hierzu finden Sie unter Zulassen der Verwendung externer KMS-Schlüssel mit AWS-Services.

Weitere Informationen zur Arbeit mit IAM-Richtlinien finden Sie unter IAM-Richtlinien.

Zulassen der Verwendung externer KMS-Schlüssel mit AWS-Services

Sie können einem Benutzer in einem anderen Konto die Berechtigung erteilen, Ihren KMS-Schlüssel mit einem Service zu verwenden, der in AWS KMS integriert ist. Beispielsweise kann ein Benutzer mit einem externen Konto Ihren KMS-Schlüssel verwenden, um die Objekte in einem Amazon S3 S3-Bucket oder die Geheimnisse zu verschlüsseln, in denen sie gespeichert sind. AWS Secrets Manager

Die Schlüsselrichtlinie muss dem externen Benutzer oder dem Konto des externen Benutzers die Berechtigung zur Verwendung des KMS-Schlüssels erteilen. Darüber hinaus müssen Sie IAM-Richtlinien an die Identität anfügen, die dem Benutzer die Berechtigung zur Nutzung des AWS-Service erteilt. Außerdem erfordert der Service möglicherweise, dass Benutzer über zusätzliche Berechtigungen in der Schlüsselrichtlinie oder IAM-Richtlinie verfügen. Eine Liste der Berechtigungen, die für einen vom Kunden verwalteten Schlüssel AWS-Service erforderlich sind, finden Sie im Thema Datenschutz im Kapitel Sicherheit des Benutzer- oder Entwicklerhandbuchs für den Service.

Verwenden von KMS-Schlüsseln in anderen Konten

Wenn Sie berechtigt sind, einen KMS-Schlüssel in einem anderen zu verwenden AWS-Konto, können Sie den KMS-Schlüssel in AWS Management Console, AWS SDKs AWS CLI, und verwenden AWS -Tools für PowerShell.

Um einen KMS-Schlüssel in einem anderen Konto in einem Shell-Befehl oder einer API-Anforderung zu identifizieren, verwenden Sie die folgenden Schlüsselbezeichner.

- Verwenden Sie für kryptografische Operationen DescribeKeyGetPublicKey, und den Schlüssel-ARN oder den Alias-ARN des KMS-Schlüssels.
- Verwenden Sie für CreateGrantGetKeyRotationStatusListGrantsRevokeGrant, und den Schlüssel ARN des KMS-Schlüssels.

Wenn Sie nur eine Schlüssel-ID oder einen Aliasnamen eingeben, wird AWS davon ausgegangen, dass sich der KMS-Schlüssel in Ihrem Konto befindet.

Die AWS KMS Konsole zeigt keine KMS-Schlüssel in anderen Konten an, auch wenn Sie berechtigt sind, sie zu verwenden. Außerdem enthalten die Listen der KMS-Schlüssel, die in den Konsolen anderer AWS -Services angezeigt werden, keine KMS-Schlüssel in anderen Konten.

Um einen KMS-Schlüssel in einem anderen Konto in der Konsole eines AWS Dienstes anzugeben, müssen Sie den Schlüssel-ARN oder Alias-ARN des KMS-Schlüssels eingeben. Die erforderliche Schlüssel-ID variiert je nach Service und kann auch zwischen der Servicekonsole und ihren API-Operationen variieren. Weitere Informationen finden Sie in der Service-Dokumentation.

Steuern Sie den Zugriff auf Schlüssel für mehrere Regionen

Sie können multiregionale Schlüssel in Compliance-, Notfallwiederherstellungs- und Backup-Szenarien verwenden, die mit einzelregionalen Schlüsseln komplexer wären. Da sich die Sicherheitseigenschaften von multiregionalen Schlüsseln erheblich von denen von einzelregionalen Schlüsseln unterscheiden, empfehlen wir, bei der Autorisierung der Erstellung, Verwaltung und Verwendung von multiregionalen Schlüsseln Vorsicht walten zu lassen.

> **Note**
>
> Vorhandene IAM-Richtlinienanweisungen mit Platzhalterzeichen im `Resource`-Feld gelten jetzt sowohl für einzelregionale Schlüssel als auch für multiregionale Schlüssel. Um sie auf KMS-Schlüssel mit einer Region oder Schlüssel mit mehreren Regionen zu beschränken, verwenden Sie den Bedingungsschlüssel kms: MultiRegion.

Verwenden Sie Ihre Autorisierungstools, um die Erstellung und Verwendung von multiregionalen Schlüsseln in jedem Szenario zu verhindern, in dem ein einzelregionaler Schlüssel ausreicht. Erlauben Sie den Prinzipalen, einen Schlüssel mit mehreren Regionen nur in solche zu replizieren, die sie benötigen. AWS-Regionen Erteilen Sie Berechtigungen für multiregionale Schlüssel nur an Prinzipale, die sie benötigen, und nur für Aufgaben, für die sie erforderlich sind.

Mithilfe von Schlüsselrichtlinien, IAM-Richtlinien und Zuschüssen können Sie es IAM-Prinzipalen ermöglichen, Schlüssel für mehrere Regionen in Ihrem zu verwalten und zu verwenden. AWS-Konto Jeder multiregionaler Schlüssel ist eine unabhängige Ressource mit einem eindeutigen Schlüssel-ARN und einer Schlüsselrichtlinie. Sie müssen für jeden Schlüssel eine Schlüsselrichtlinie einrichten und pflegen und sicherstellen, dass neue und vorhandene IAM-Richtlinien Ihre Autorisierungsstrategie implementieren.

AWS KMS Verwendet eine mit dem IAM-Dienst verknüpfte Rolle, um Schlüssel für mehrere Regionen zu unterstützen. Diese Rolle gibt AWS KMS die Berechtigungen, die es zum Synchronisieren von gemeinsam genutzten Eigenschaften benötigt. Weitere Informationen finden Sie unter Autorisierung zur Synchronisation von AWS KMS Schlüsseln aus mehreren Regionen.

Themen

- Grundlagen der Autorisierung für multiregionale Schlüssel
- Autorisieren von Administratoren und Benutzern für multiregionale Schlüssel

Grundlagen der Autorisierung für multiregionale Schlüssel

Berücksichtigen Sie beim Entwerfen von Schlüsselrichtlinien und IAM-Richtlinien für multiregionale Schlüssel die folgenden Grundsätze.

- Schlüsselrichtlinie – Jeder multiregionale Schlüssel ist eine unabhängige KMS-Schlüsselressource mit einer eigenen Schlüsselrichtlinie. Sie können dieselbe oder eine andere Schlüsselrichtlinie auf jeden Schlüssel in dem Satz von verwandten multiregionalen Schlüsseln anwenden. Schlüsselrichtlinien sind keine gemeinsamen Eigenschaften von Schlüsseln mit mehreren Regionen. AWS KMS kopiert oder synchronisiert keine Schlüsselrichtlinien zwischen zugehörigen Schlüsseln mit mehreren Regionen.

  Wenn Sie einen Replikatschlüssel in der AWS KMS Konsole erstellen, zeigt die Konsole der Einfachheit halber die aktuelle Schlüsselrichtlinie des Primärschlüssels an. Sie können diese Schlüsselrichtlinie verwenden, bearbeiten oder löschen und ersetzen. Aber selbst wenn Sie die Primärschlüsselrichtlinie unverändert akzeptieren, werden die Richtlinien AWS KMS nicht synchronisiert. Wenn Sie beispielsweise die Schlüsselrichtlinie des Primärschlüssels ändern, bleibt die Schlüsselrichtlinie des Replikatschlüssels unverändert.
- Standardschlüsselrichtlinie — Wenn Sie Schlüssel für mehrere Regionen mithilfe der `ReplicateKey` Operationen CreateKeyund erstellen, wird die Standardschlüsselrichtlinie

angewendet, sofern Sie in der Anforderung keine Schlüsselrichtlinie angeben. Dies ist die gleiche Standard-Schlüsselrichtlinie, die auf einzelregionale Schlüssel angewendet wird.

- IAM-Richtlinien – Wie bei allen KMS-Schlüsseln können Sie IAM-Richtlinien verwenden, um den Zugriff auf multiregionale Schlüssel nur dann zu steuern, wenn die Schlüsselrichtlinie es erlaubt. IAM-Richtlinien gelten standardmäßig für alle AWS-Regionen . Sie können jedoch Bedingungsschlüssel wie aws: verwendenRequestedRegion, um Berechtigungen auf eine bestimmte Region zu beschränken.

  Zum Erstellen von Primär- und Replikatschlüsseln müssen Prinzipale die `kms:CreateKey`-Berechtigung in einer IAM-Richtlinie haben, die für die Region gilt, wo der Schlüssel erstellt wird.

- Zuschüsse — AWS KMS Zuschüsse sind regional. Jede Erteilung erlaubt Berechtigungen für einen KMS-Schlüssel. Sie können Erteilungen verwenden, um Berechtigungen für einen multiregionalen Primärschlüssel oder Replikatschlüssel zuzulassen. Sie können jedoch keine einzelne Erteilung verwenden, um Berechtigungen für mehrere KMS-Schlüssel zuzulassen, selbst wenn es sich um verwandte multiregionale Schlüssel handelt.

- Schüssel-ARN – Jeder multiregionale Schlüssel verfügt über einen Schüssel-ARN. Die Schlüssel ARNs verwandter Schlüssel für mehrere Regionen haben dieselbe Partition, dasselbe Konto und dieselbe Schlüssel-ID, aber unterschiedliche Regionen.

  Um eine IAM-Richtlinienanweisung auf einen bestimmten multiregionalen Schlüssel anzuwenden, verwenden Sie dessen Schlüssel-ARN oder ein Schlüssel-ARN-Muster, das die Region enthält. Verwenden Sie ein Platzhalterzeichen (*) im Region-Element des ARN, um eine IAM-Richtlinienanweisung auf alle verwandten multiregionalen Schlüssel.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Describe*",
    "kms:List*"
  ],
  "Resource": {
      "arn:aws:kms:*::111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab"
  }
}
```

  Um eine Richtlinienerklärung auf alle Schlüssel mit mehreren Regionen in Ihrem anzuwenden AWS-Konto, können Sie die MultiRegion Richtlinienbedingung kms: oder ein Schlüssel-ID-Muster verwenden, das das eindeutige Präfix enthält. `mrk-`

- Dienstbezogene Rolle — Principals, die Primärschlüssel für mehrere Regionen erstellen, müssen über die iam: -Berechtigung verfügen. CreateServiceLinkedRole

  AWS KMS Nimmt eine dienstbezogene IAM-Rolle an, um die gemeinsamen Eigenschaften verwandter Schlüssel mit mehreren Regionen zu synchronisieren. AWS KMS erstellt die dienstverknüpfte Rolle im, AWS-Konto wann immer Sie einen Primärschlüssel für mehrere Regionen erstellen. (Wenn die Rolle vorhanden ist, erstellt AWS KMS sie neu, was keine schädliche Wirkung hat.) Die Rolle ist in allen Regionen gültig. Damit die dienstverknüpfte Rolle erstellt (oder neu erstellt) werden kann, AWS KMS müssen Principals, die Primärschlüssel für mehrere Regionen erstellen, über die Berechtigung iam: verfügen. CreateServiceLinkedRole

Autorisieren von Administratoren und Benutzern für multiregionale Schlüssel

Prinzipale, die multiregionale Schlüssel erstellen und verwalten, benötigen die folgenden Berechtigungen in den primären und Replikatregionen:

- `kms:CreateKey`
- `kms:ReplicateKey`
- `kms:UpdatePrimaryRegion`
- `iam:CreateServiceLinkedRole`

Erstellen eines Primärschlüssels

Um einen Primärschlüssel mit mehreren Regionen zu erstellen, benötigt der Principal die CreateServiceLinkedRole Berechtigungen kms: CreateKey und iam: in einer IAM-Richtlinie, die in der Region des Primärschlüssels gilt. Prinzipale, die über diese Berechtigungen verfügen, können einzelregionale Schlüssel und multiregionale Schlüssel erstellen, es sei denn, Sie beschränken ihre Berechtigungen.

Mit dieser `iam:CreateServiceLinkedRole` Berechtigung kann die AWSServiceRoleForKeyManagementServiceMultiRegionKeysRolle AWS KMS zum Synchronisieren der gemeinsamen Eigenschaften verwandter Schlüssel mit mehreren Regionen erstellt werden.

Zum Beispiel kann mit dieser IAM-Richtlinie ein Prinzipal einen beliebigen KMS-Schlüsseltyp erstellen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement":{
      "Action": [
        "kms:CreateKey",
        "iam:CreateServiceLinkedRole"
      ],
      "Effect":"Allow",
      "Resource":"*"
  }
}
```

Verwenden Sie den MultiRegion Bedingungsschlüssel kms:, um die Erlaubnis zur Erstellung von Primärschlüsseln für mehrere Regionen zu gewähren oder zu verweigern. Gültige Werte sind `true` (multiregionaler Schlüssel) oder `false` (einzelregionaler Schlüssel). Beispielsweise verwendet die folgende IAM-Richtlinienanweisung eine `Deny`-Aktion mit dem `kms:MultiRegion`-Bedingungsschlüssel, um zu verhindern, dass Prinzipale multiregionale Schlüssel erstellen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement":{
      "Action":"kms:CreateKey",
      "Effect":"Deny",
      "Resource":"*",
      "Condition": {
          "Bool": {
            "kms:MultiRegion": true
          }
      }
  }
}
```

Replizieren von Schlüsseln

Um einen multiregionalen Replikatschlüssel zu erstellen, benötigt der Prinzipal die folgenden Berechtigungen:

- kms: ReplicateKey Berechtigung in der Schlüsselrichtlinie des Primärschlüssels.
- kms: CreateKey Berechtigung in einer IAM-Richtlinie, die in der Region des Replikatschlüssels wirksam ist.

Seien Sie vorsichtig, wenn Sie diese Berechtigungen zulassen. Sie ermöglichen es Prinzipalen, KMS-Schlüssel und die Schlüsselrichtlinien zu erstellen, die ihre Verwendung autorisieren. Die `kms:ReplicateKey`-Berechtigung autorisiert auch die Übertragung von Schlüsselmaterial über Regionsgrenzen innerhalb von AWS KMS.

Verwenden Sie den Bedingungsschlüssel kms:, um einzuschränken, AWS-Regionen in welchen Bereichen ein Schlüssel aus mehreren Regionen repliziert werden kann. ReplicaRegion Es begrenzt nur die`kms:ReplicateKey`-Berechtigung. Andernfalls hat es keine Auswirkungen. Beispielsweise ermöglicht die folgende Schlüsselrichtlinie dem Prinzipal, diesen Primärschlüssel zu replizieren, jedoch nur in den angegebenen Regionen.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Administrator"
  },
  "Action": "kms:ReplicateKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ReplicaRegion": [
         "us-east-1",
         "eu-west-3",
         "ap-southeast-2"
      ]
    }
  }
}
```

Aktualisieren der primären Region

Autorisierte Prinzipale können einen Replikatschlüssel in einen Primärschlüssel konvertieren, wodurch der frühere Primärschlüssel in ein Replikat umgewandelt wird. Diese Aktion ist bekannt als Aktualisieren der primären Region. Um die primäre Region zu aktualisieren, benötigt der Principal die UpdatePrimaryRegionkms:-Berechtigung in beiden Regionen. Sie können diese Berechtigungen in einer Schlüsselrichtlinie oder einer IAM-Richtlinie bereitstellen.

- `kms:UpdatePrimaryRegion` für den Primärschlüssel. Diese Berechtigung muss in der Region des Primärschlüssels wirksam sein.
- `kms:UpdatePrimaryRegion` für den Replikatschlüssel. Diese Berechtigung muss in der Region des Replikatschlüssels wirksam sein.

Die folgende Schlüsselrichtlinie gibt Benutzern, die die Administratorrolle übernehmen können, die Berechtigung zum Aktualisieren der primären Region des KMS-Schlüssels. Dieser KMS-Schlüssel kann der Primärschlüssel oder ein Replikatschlüssel in dieser Operation sein.

```
{
  "Effect": "Allow",
  "Resource": "*",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Administrator"
  },
  "Action": "kms:UpdatePrimaryRegion"
}
```

Verwenden Sie den PrimaryRegion Bedingungsschlüssel kms: AWS-Regionen, um einzuschränken, wer einen Primärschlüssel hosten kann. Die folgende IAM-Richtlinienanweisung ermöglicht es den Prinzipalen beispielsweise, die primäre Region der Schlüssel für mehrere Regionen in der zu aktualisieren AWS-Konto, aber nur, wenn die neue primäre Region eine der angegebenen Regionen ist.

```
{
  "Effect": "Allow",
  "Action": "kms:UpdatePrimaryRegion",
  "Resource": {
      "arn:aws:kms:*:111122223333:key/*"
  },
  "Condition": {
```

```
    "StringEquals": {
      "kms:PrimaryRegion": [
         "us-west-2",
         "sa-east-1",
         "ap-southeast-1"
      ]
    }
  }
}
```

Verwenden und Verwalten von multiregionalen Schlüsseln

Standardmäßig haben Prinzipale, die über die Berechtigung zum Verwenden und Verwalten von KMS-Schlüsseln in einem AWS-Konto und einer Region haben, auch die Berechtigung, multiregionale Schlüssel zu verwenden und zu verwalten. Sie können jedoch den MultiRegion Bedingungsschlüssel kms: verwenden, um nur Schlüssel mit einer Region oder nur Schlüssel mit mehreren Regionen zuzulassen. Oder verwenden Sie den MultiRegionKeyType Bedingungsschlüssel kms:, um nur Primärschlüssel mit mehreren Regionen oder nur Replikatschlüssel zuzulassen. Beide Bedingungsschlüssel steuern den Zugriff auf den CreateKeyVorgang und auf alle Operationen, die einen vorhandenen KMS-Schlüssel verwenden, z. B. Verschlüsseln oder. EnableKey

Beispielsweise verwendet die folgende IAM-Richtlinienanweisung den `kms:MultiRegion`-Bedingungsschlüssel, um zu verhindern, dass Prinzipale multiregionale Schlüssel verwenden oder verwalten.

```
{
  "Effect": "Deny",
  "Action": "kms:*",
  "Resource": "*",
  "Condition": {
    "Bool": "kms:MultiRegion": true
  }
}
```

In diesem Beispiel verwendet die IAM-Richtlinienanweisung die `kms:MultiRegionKeyType`-Bedingung, damit Prinzipale das Löschen von Schlüsseln planen und abbrechen können, jedoch nur für multiregionale Replikatschlüssel.

```
{
  "Effect": "Allow",
```

```
  "Action": [
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": {
      "arn:aws:kms:us-west-2:111122223333:key/*"
  },
  "Condition": {
    "StringEquals": "kms:MultiRegionKeyType": "REPLICA"
  }
}
```

Bestimmung des Zugriffs auf AWS KMS keys

Um zu ermitteln, wer oder was derzeit Zugriff auf eine hat AWS KMS key, müssen Sie die Schlüsselrichtlinie des KMS-Schlüssels, alle Zuweisungen, die für den KMS-Schlüssel gelten, und möglicherweise alle AWS Identity and Access Management (IAM-) Richtlinien untersuchen. Sie können damit den Umfang der potenziellen Nutzung eines KMS-Schlüssels bestimmen, oder Compliance- oder Auditing-Anforderungen erfüllen. Die folgenden Themen können Ihnen dabei helfen, eine vollständige Liste der AWS -Prinzipale (Identitäten) zu erzeugen, die derzeit Zugriff auf einen KMS-Schlüssel haben.

Themen

- Untersuchen der Schlüsselrichtlinie
- Untersuchen von IAM-Richtlinien
- Prüfen von Erteilungen

Untersuchen der Schlüsselrichtlinie

Schlüsselrichtlinien sind die primäre Methode zur Zugriffssteuerung für KMS-Schlüssel. Jeder KMS-Schlüssel besitzt genau eine Schlüsselrichtlinie.

Wenn eine Schlüsselrichtlinie aus der Standard-Schlüsselrichtlinie besteht oder sie enthält, berechtigt die Schlüsselrichtlinie IAM-Administratoren im Konto dazu, mithilfe von IAM-Richtlinien den Zugriff auf den KMS-Schlüssel zu steuern. Wenn die Schlüsselrichtlinie einem anderen AWS-Konto die Berechtigung zur Verwendung des KMS-Schlüssels erteilt, können die IAM-Administratoren im externen Konto diese Berechtigungen anhand von IAM-Richtlinien delegieren. Um die komplette Liste der Prinzipale mit Zugriff auf den KMS-Schlüssel zu bestimmen, untersuchen Sie die IAM-Richtlinien.

Um die wichtigsten Richtlinien für einen vom AWS KMS Kunden verwalteten Schlüssel oder Von AWS verwalteter Schlüsselin Ihrem Konto einzusehen, verwenden Sie die Operation AWS Management Console oder die GetKeyPolicyOperation in der AWS KMS API. Um die Schlüsselrichtlinie anzeigen zu können, müssen Sie über `kms:GetKeyPolicy`-Berechtigungen für den KMS-Schlüssel verfügen. Anweisungen zum Anzeigen der Schlüsselrichtlinie für einen KMS-Schlüssel finden Sie unter the section called “Wichtige Richtlinien anzeigen”.

Überprüfen Sie das Schlüssel-Richtliniendokument und notieren Sie alle Hauptpunkte, die in jedem `Principal` Element der Richtlinie aufgeführt ist. In einer Richtlinienerklärung mit `Allow` Wirkung haben die IAM-Benutzer, die IAM-Rollen und AWS-Konten das `Principal` Element Zugriff auf diesen KMS-Schlüssel.

> **Note**
>
> Setzen Sie den Prinzipal nicht auf ein Sternchen (*) in einer Schlüsselrichtlinienanweisung, die Berechtigungen erlaubt, es sei denn, Sie verwenden Bedingungen, um die Schlüsselrichtlinie einzuschränken. Ein Sternchen gibt jede Identität in jeder AWS-Konto Berechtigung zur Verwendung des KMS-Schlüssels an, sofern dies nicht in einer anderen Richtlinienanweisung ausdrücklich verweigert wird. Benutzer in anderen Ländern AWS-Konten können Ihren KMS-Schlüssel immer dann verwenden, wenn sie über entsprechende Berechtigungen in ihrem eigenen Konto verfügen.

Die folgenden Beispiele verwenden die Richtlinien-Anweisungen in der Standard-Schlüsselrichtlinie, um dies zu demonstrieren.

Example Richtlinienanweisung 1

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
  "Action": "kms:*",
  "Resource": "*"
}
```

In Grundsatzerklärung 1 `arn:aws:iam::111122223333:root` ist dies ein AWS Kontoprinzipal, der sich auf AWS-Konto 111122223333 bezieht. (Er ist nicht der Root-Benutzer des Kontos.) Standardmäßig ist eine Richtlinienerklärung wie diese im Dokument mit den wichtigsten Richtlinien

enthalten, wenn Sie einen neuen KMS-Schlüssel mit dem erstellen oder programmgesteuert einen neuen KMS-Schlüssel erstellen AWS Management Console, aber keine Schlüsselrichtlinie angeben.

Ein wichtiges Richtliniendokument mit einer Erklärung, die den Zugriff auf die ermöglicht, AWS-Konto aktiviert die IAM-Richtlinien im Konto, um den Zugriff auf den KMS-Schlüssel zu ermöglichen. Das bedeutet, dass Benutzer und Rollen im Konto Zugriff auf den KMS-Schlüssel haben könnten, auch wenn sie explizit nicht als Prinzipale im Schlüsselrichtliniendokument aufgelistet sind. Prüfen Sie sorgfältig alle IAM-Richtlinien in allen als Principals AWS-Konten aufgelisteten Richtlinien, um festzustellen, ob sie den Zugriff auf diesen KMS-Schlüssel ermöglichen.

Example Richtlinienanweisung 2

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/KMSKeyAdmins"},
  "Action": [
    "kms:Describe*",
    "kms:Put*",
    "kms:Create*",
    "kms:Update*",
    "kms:Enable*",
    "kms:Revoke*",
    "kms:List*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

`arn:aws:iam::111122223333:role/KMSKeyAdmins`Bezieht sich in Grundsatzerklärung 2 auf die IAM-Rolle mit dem Namen KMSKey Admins in 111122223333. AWS-Konto Benutzer, die diese Rolle annehmen dürfen, dürfen die in der Richtlinienanweisung aufgeführten Aktionen ausführen, die die administrativen Aktionen für die Verwaltung eines KMS-Schlüssels sind.

Example Richtlinienanweisung 3

```
{
  "Sid": "Allow use of the key",
```

```
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/EncryptionApp"},
  "Action": [
    "kms:DescribeKey",
    "kms:GenerateDataKey*",
    "kms:Encrypt",
    "kms:ReEncrypt*",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

`arn:aws:iam::111122223333:role/EncryptionApp`Bezieht sich in Grundsatzerklärung 3 auf die in 111122223333 genannte IAM-Rolle. EncryptionApp AWS-Konto Prinzipale, die diese Rolle annehmen dürfen, dürfen die in der Richtlinienanweisung aufgeführten Aktionen durchführen, die die kryptografischen Operationen für einen symmetrischen KMS-Schlüssel enthalten.

Example Richtlinienanweisung 4

```
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/EncryptionApp"},
  "Action": [
    "kms:ListGrants",
    "kms:CreateGrant",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

`arn:aws:iam::111122223333:role/EncryptionApp`Bezieht sich in Grundsatzerklärung 4 auf die in 111122223333 genannte IAM-Rolle. EncryptionApp AWS-Konto Prinzipal, die diese Rolle annehmen dürfen, dürfen die in der Richtlinienanweisung aufgeführten Aktionen ausführen. Diese Aktionen in Kombination mit den erlaubten Aktionen in der Beispiel-Richtlinienanweisung 3 sind diejenigen, die zum Delegieren der Verwendung des KMS-Schlüssels für die meisten AWS -Services, die mit AWS KMS integriert sind, erforderlich sind, insbesondere für die Services, die Erteilungen verwenden. Der GrantIsFor AWSResource Wert kms: im `Condition` Element stellt sicher, dass die Delegierung nur zulässig ist, wenn es sich bei dem Delegaten um einen AWS Dienst handelt, der in die Autorisierung integriert ist und Grants für deren Autorisierung verwendet. AWS KMS

Die unterschiedlichen Möglichkeiten zum Angeben eines Prinzipals in einem Schlüsselrichtlinien-Dokument finden Sie unter Angeben eines Prinzipals im IAM-Benutzerhandbuch.

Weitere Informationen zu den AWS KMS wichtigsten Richtlinien finden Sie unterDie wichtigsten Richtlinien in AWS KMS.

Untersuchen von IAM-Richtlinien

Zusätzlich zu den Schlüsselrichtlinien und Berechtigungen können Sie auch IAM-Richtlinien verwenden, um den Zugriff auf einen KMS-Schlüssel zu erlauben. Weitere Informationen darüber, wie IAM-Richtlinien und Schlüsselrichtlinien zusammen funktionieren, finden Sie unter Problembehandlung bei Berechtigungen AWS KMS.

Um zu bestimmen, welche Prinzipale derzeit über IAM-Richtlinien auf einen KMS-Schlüssel zugreifen können, verwenden Sie das browserbasierte IAM-Richtliniensimulator-Tool. Alternativ können Sie Anforderungen an die IAM-API durchführen.

Untersuchen von IAM-Richtlinien

- Untersuchen von IAM-Richtlinien mit dem IAM-Richtliniensimulator
- Untersuchen von IAM-Richtlinien mit der IAM-API

Untersuchen von IAM-Richtlinien mit dem IAM-Richtliniensimulator

Mit dem IAM-Richtliniensimulator können Sie erfahren, welche Prinzipale über eine IAM-Richtlinie Zugriff auf einen KMS-Schlüssel haben.

So verwenden Sie den IAM-Richtliniensimulator, um den Zugriff auf einen KMS-Schlüssel zu bestimmen

1. Melden Sie sich beim an AWS Management Console und öffnen Sie dann den IAM Policy Simulator unterhttps://policysim.aws.amazon.com/.
2. Wählen Sie im Feld Benutzer, Gruppen und Rollen den Benutzer, Gruppe oder Rolle, deren Richtlinien Sie simulieren möchten.
3. (Optional) Deaktivieren Sie das Kontrollkästchen neben den Richtlinien, die Sie von der Simulation auslassen möchten. Um alle Richtlinien zu simulieren, markieren Sie alle Richtlinien.
4. Führen Sie im Bereich Richtliniensimulator die folgenden Schritte aus:

    a. Wählen Sie für Service wählen die Option Key Management Service.

   b. Um bestimmte AWS KMS Aktionen zu simulieren, wählen Sie unter Aktionen auswählen die zu simulierenden Aktionen aus. Um alle AWS KMS Aktionen zu simulieren, wählen Sie Alle auswählen.

5. (Optional) Der Richtliniensimulator simuliert standardmäßig einen Zugriff auf alle KMS-Schlüssel. Um den Zugriff auf einen bestimmten KMS-Schlüssel zu simulieren, wählen Sie Simulation Settings (Simulationseinstellungen) und geben Sie dann den Amazon-Ressourcennamen (ARN) des zu simulierenden KMS-Schlüssels ein.
6. Wählen Sie Run Simulation (Simulation ausführen).

Sie können die Ergebnisse der Simulation im Abschnitt Ergebnisse sehen. Wiederholen Sie die Schritte 2 bis 6 für alle Benutzer, Gruppen und Rollen im AWS-Konto.

Untersuchen von IAM-Richtlinien mit der IAM-API

Sie können mit der IAM-API programmgesteuert IAM-Richtlinien untersuchen. Die folgenden Schritte bieten eine allgemeine Übersicht darüber:

1. Verwenden Sie für jede, die in der Schlüsselrichtlinie als Prinzipal AWS-Konto aufgeführt ist (d. h. für jeden AWS Kontoprinzipal, der in diesem Format angegeben ist:`"Principal": {"AWS": "arn:aws:iam::111122223333:root"}`), die ListRolesOperationen ListUsersund in der IAM-API, um alle Benutzer und Rollen im Konto abzurufen.
2. Verwenden Sie für jeden Benutzer und jede Rolle in der Liste den SimulatePrincipalPolicyVorgang in der IAM-API und übergeben Sie dabei die folgenden Parameter:
   - Geben Sie für `PolicySourceArn` den Amazon-Ressourcennamen (ARN) für einen Benutzer oder eine Rolle aus der Liste an. Sie können nur jeweils einen `PolicySourceArn` pro `SimulatePrincipalPolicy`-Anforderung angeben. Deshalb müssen Sie diese Operation mehrfach aufrufen – jeweils einmal für jeden Benutzer und jede Rolle in der Liste.
   - Geben Sie für die `ActionNames` Liste jede AWS KMS API-Aktion an, die simuliert werden soll. Um alle AWS KMS API-Aktionen zu simulieren, verwenden Sie`kms:*`. Um einzelne AWS KMS API-Aktionen zu testen, stellen Sie jeder API-Aktion "`kms:`" voran, zum Beispiel "`kms:ListKeys`“. Eine vollständige Liste aller AWS KMS -API-Aktionen finden Sie unter Aktionen in der AWS Key Management Service -API-Referenz.
   - (Optional) Um zu ermitteln, ob die Benutzer oder Rollen Zugriff auf bestimmte KMS-Schlüssel haben, verwenden Sie den `ResourceArns` Parameter, um eine Liste der Amazon-Ressourcennamen (ARNs) der KMS-Schlüssel anzugeben. Vermeiden Sie den

`ResourceArns`-Parameter, um zu prüfen, ob die Benutzer oder Rollen Zugriff auf irgendwelche KMS-Schlüssel haben.

IAM antwortet auf jede `SimulatePrincipalPolicy`-Anforderung mit einer Bewertungsentscheidung: `allowed`, `explicitDeny`, oder `implicitDeny`. Für jede Antwort, die eine Bewertungsentscheidung von enthält`allowed`, enthält die Antwort den Namen des spezifischen AWS KMS API-Vorgangs, der zulässig ist. Darüber hinaus enthält sie den ARN des KMS-Schlüssels, der in der Bewertung verwendet wurde, falls vorhanden.

Prüfen von Erteilungen

Grants sind erweiterte Mechanismen zur Angabe von Berechtigungen, mit denen Sie oder ein integrierter AWS Dienst angeben AWS KMS können, wie und wann ein KMS-Schlüssel verwendet werden kann. Erteilungen werden einem KMS-Schlüssel angefügt. Jede Erteilung enthält den Prinzipal, dem die Berechtigung zur Verwendung des KMS-Schlüssels erteilt wird, sowie eine Liste der erlaubten Produktionen. Berechtigungen sind eine Alternative zu den Schlüsselrichtlinien und eignen sich für bestimmte Anwendungsfälle. Weitere Informationen finden Sie unter Zuschüsse in AWS KMS.

Verwenden Sie den AWS KMS ListGrantsVorgang, um eine Liste der Berechtigungen für einen KMS-Schlüssel abzurufen. Sie können die Erteilungen für einen KMS-Schlüssel untersuchen, um zu bestimmen, wer oder was derzeit über diese Erteilungen Zugriff auf den KMS-Schlüssel hat. Das folgende Beispiel ist eine JSON-Darstellung einer Erteilung, die vom Befehl list-grants in der AWS CLI abgerufen wurde.

```
{"Grants": [{
  "Operations": ["Decrypt"],
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Name": "0d8aa621-43ef-4657-b29c-3752c41dc132",
  "RetiringPrincipal": "arn:aws:iam::123456789012:root",
  "GranteePrincipal": "arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/
i-5d476fab",
  "GrantId": "dc716f53c93acacf291b1540de3e5a232b76256c83b2ecb22cdefa26576a2d3e",
  "IssuingAccount": "arn:aws:iam::111122223333:root",
  "CreationDate": 1.444151834E9,
  "Constraints": {"EncryptionContextSubset": {"aws:ebs:id": "vol-5cccfb4e"}}
}]}
```

Um herauszufinden, wer oder was Zugriff auf den KMS-Schlüssel hat, suchen Sie nach dem `"GranteePrincipal"`-Element. Im vorherigen Beispiel ist der Principal des Empfängers ein angenommener Rollenbenutzer, der der EC2 Instanz i-5d476fab zugeordnet ist. Die EC2 Infrastruktur verwendet diese Rolle, um das verschlüsselte EBS-Volume vol-5cccfb4e an die Instance anzuhängen. In diesem Fall ist die EC2 Infrastrukturrolle berechtigt, den KMS-Schlüssel zu verwenden, da Sie zuvor ein verschlüsseltes EBS-Volume erstellt haben, das durch diesen KMS-Schlüssel geschützt ist. Anschließend haben Sie das Volume an eine EC2 Instance angehängt.

Hier ein weiteres Beispiel einer JSON-Darstellung einer Erteilung, die vom Befehl list-grants in der AWS CLI abgerufen wurde. Im folgenden Beispiel ist der Principal des Stipendiaten ein weiterer AWS-Konto.

```
{"Grants": [{
  "Operations": ["Encrypt"],
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Name": "",
  "GranteePrincipal": "arn:aws:iam::444455556666:root",
  "GrantId": "f271e8328717f8bde5d03f4981f06a6b3fc18bcae2da12ac38bd9186e7925d11",
  "IssuingAccount": "arn:aws:iam::111122223333:root",
  "CreationDate": 1.444151269E9
}]}
```

Verschlüsselungskontext

> **Note**
>
> Sie können mit einem asymmetrische KMS-Schlüssel oder einem HMAC-KMS-Schlüssel keinen Verschlüsselungskontext in einer kryptografischen Produktion angeben. Asymmetrische Algorithmen und MAC-Algorithmen unterstützen keinen Verschlüsselungskontext.

Alle AWS KMS kryptografischen Operationen mit symmetrischer Verschlüsselung von KMS-Schlüsseln akzeptieren einen Verschlüsselungskontext, einen optionalen Satz nicht geheimer Schlüssel-Wert-Paare, die zusätzliche Kontextinformationen zu den Daten enthalten können. Sie können Verschlüsselungskontext in `Encrypt` Operationen einfügen, um die Autorisierung und Überprüfbarkeit Ihrer API-Entschlüsselungsaufrufe AWS KMS zu verbessern. AWS KMS AWS

KMS verwendet den Verschlüsselungskontext als zusätzliche authentifizierte Daten (AAD), um die authentifizierte Verschlüsselung zu unterstützen. Der Verschlüsselungskontext ist kryptografisch an den Chiffretext gebunden, sodass derselbe Verschlüsselungskontext zum Entschlüsseln der Daten erforderlich ist.

Der Verschlüsselungskontext ist nicht geheim und nicht verschlüsselt. Er erscheint als Klartext in AWS CloudTrail -Protokollen, kann also zum Identifizieren und Kategorisieren der kryptografischen Operationen herangezogen werden. Ihr Verschlüsselungskontext sollte keine sensiblen Informationen enthalten. Wir empfehlen, dass Ihr Verschlüsselungskontext die zu verschlüsselnden oder zu entschlüsselnden Daten beschreibt. Wenn Sie z. B. eine Datei verschlüsseln, können Sie einen Teil des Dateipfads als Verschlüsselungskontext verwenden.

```
"encryptionContext": {
    "department": "10103.0"
}
```

Bei der Verschlüsselung von Volumes und Snapshots, die mit dem Amazon Elastic Block Store (Amazon EBS) CreateSnapshot-Vorgang erstellt wurden, verwendet Amazon EBS beispielsweise die Volume-ID als Wert für den Verschlüsselungskontext.

```
"encryptionContext": {
  "aws:ebs:id": "vol-abcde12345abc1234"
}
```

Sie können den Verschlüsselungskontext auch verwenden, um den Zugriff in Ihrem Konto zu verfeinern oder einzuschränken. AWS KMS keys Sie können den Verschlüsselungskontext als Einschränkung in Erteilungen und als Bedingung in Richtlinien verwenden.

Informationen zur Verwendung des Verschlüsselungskontextes zum Schutz der Integrität verschlüsselter Daten finden Sie im Beitrag So schützen Sie die Integrität Ihrer verschlüsselten Daten durch Verwendung AWS Key Management Service und EncryptionContext im AWS Sicherheitsblog.

Regeln für den Verschlüsselungskontext

AWS KMS erzwingt die folgenden Regeln für Schlüssel und Werte im Verschlüsselungskontext.

- Der Schlüssel und der Wert in einem Verschlüsselungskontextpaar müssen einfache Literalzeichenfolgen sein. Wenn Sie einen anderen Typ verwenden, z. B. eine Ganzzahl oder Gleitkommazahl, interpretiert AWS KMS ihn als Zeichenfolge.

- Die Schlüssel und Werte in einem Verschlüsselungskontext können Unicode-Zeichen enthalten. Wenn ein Verschlüsselungskontext Zeichen enthält, die in Schlüsselrichtlinien oder IAM-Richtlinien nicht zulässig sind, können Sie den Verschlüsselungskontext nicht in Richtlinienbedingungsschlüsseln angeben, z. B. `kms:EncryptionContext:context-key` und `kms:EncryptionContextKeys`. Weitere Informationen zu Dokumentenregeln für Schlüsselrichtlinien finden Sie unter Schlüsselrichtlinienformat. Weitere Informationen zu Regeln für IAM-Richtliniendokumente finden Sie unter Anforderungen für den IAM-Namen im IAM Benutzerhandbuch.

Verschlüsselungskontext in Richtlinien

Der Verschlüsselungskontext wird hauptsächlich verwendet, um Integrität und Authentizität zu überprüfen. Sie können den Verschlüsselungskontext aber auch verwenden, um den Zugriff auf AWS KMS keys mit symmetrischer Verschlüsselung in Schlüsselrichtlinien und IAM-Richtlinien zu kontrollieren.

Die EncryptionContextKeys Bedingungsschlüssel kmsEncryptionContext:: und kms: erlauben (oder verweigern) eine Berechtigung nur, wenn die Anforderung bestimmte Verschlüsselungskontextschlüssel oder Schlüssel-Wert-Paare enthält.

Die folgende Schlüsselrichtlinienanweisung erlaubt es beispielsweise der `RoleForExampleApp`-Rolle, den KMS-Schlüssel in `Decrypt`-Operationen zu verwenden. Sie verwendet den `kms:EncryptionContext:context-key`-Bedingungsschlüssel, um diese Berechtigung nur zu erlauben, wenn der Verschlüsselungskontext in der Anforderung ein `AppName:ExampleApp`-Verschlüsselungskontextpaar enthält.

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
}
```

Weitere Informationen zu diesen Bedingungskontextschlüsseln finden Sie unter Zustandstasten für AWS KMS.

Verschlüsselungskontext in Erteilungen

Wenn Sie eine Erteilung erstellen, können Sie Beschränkungen für die Gewährung einbeziehen, die Bedingungen für die Gewährungsberechtigungen festlegen. AWS KMS unterstützt zwei Erteilungsbeschränkungen`EncryptionContextSubset`, `EncryptionContextEquals` und beide beziehen den Verschlüsselungskontext in einer Anforderung für einen kryptografischen Vorgang mit ein. Wenn Sie diese Erteilungseinschränkungen verwenden, sind die Berechtigungen in der Erteilung nur dann wirksam, wenn der Verschlüsselungskontext in der Anforderung für die kryptografische Operation die Anforderungen der Erteilungseinschränkungen erfüllt.

Sie können beispielsweise einer `EncryptionContextEquals` Grant, die den GenerateDataKeyVorgang ermöglicht, eine Grant-Beschränkung hinzufügen. Bei dieser Einschränkung erlaubt die Erteilung die Operation nur dann, wenn der Verschlüsselungskontext in der Anforderung mit dem Verschlüsselungskontext in der Erteilungseinschränkung übereinstimmen (inkl. Groß-/Kleinschreibung).

```
$ aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:user/exampleUser \
    --retiring-principal arn:aws:iam::111122223333:role/adminRole \
    --operations GenerateDataKey \
    --constraints EncryptionContextEquals={Purpose=Test}
```

Eine Anforderung wie die folgende vom Empfänger-Prinzipal würde die `EncryptionContextEquals`-Einschränkung zulassen.

```
$ aws kms generate-data-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --key-spec AES_256 \
    --encryption-context Purpose=Test
```

Weitere Informationen über die Erteilungseinschränkungen finden Sie unter Verwenden von Erteilungs-Einschränkungen. Ausführliche Informationen zu Erteilungen finden Sie unter the section called “Gewährungen”.

Protokollieren des Verschlüsselungskontexts

AWS KMS verwendet AWS CloudTrail , um den Verschlüsselungskontext zu protokollieren, sodass Sie feststellen können, auf welche KMS-Schlüssel und -Daten zugegriffen wurde. Das bedeutet, dass der Protokolleintrag genau anzeigt, welcher KMS-Schlüssel verwendet wurde, um spezifische Daten, auf die der Verschlüsselungskontext im Protokolleintrag hinweist, zu verschlüsseln oder entschlüsseln.

> **Important**
>
> Da der Verschlüsselungskontext protokolliert wird, sollte er keine vertraulichen Informationen enthalten.

Speichern des Verschlüsselungskontexts

Um die Verwendung eines beliebigen Verschlüsselungskontextes beim Aufruf der Operationen `Decrypt` oder `ReEncrypt` zu vereinfachen, können Sie den Verschlüsselungskontext neben den verschlüsselten Daten speichern. Wir empfehlen, dass Sie nur so viel vom Verschlüsselungskontext speichern, dass Sie den vollständigen Verschlüsselungskontext erstellen können, wenn Sie ihn für die Ver- oder Entschlüsselung benötigen.

Wenn der Verschlüsselungskontext beispielsweise der vollständig qualifizierte Pfad zu einer Datei ist, speichern Sie nur einen Teil dieses Pfades mit dem verschlüsselten Dateiinhalt. Wenn Sie dann den vollständigen Verschlüsselungstext benötigen, können Sie ihn aus dem gespeicherten Fragment rekonstruieren. Wenn jemand die Datei verändert (sie beispielsweise umbenennt oder an einen anderen Ort verschiebt), ändert sich der Wert des Verschlüsselungskontextes und die Entschlüsselungsanforderung schlägt fehl.

Testen der Berechtigungen

Für die Verwendung AWS KMS benötigen Sie Anmeldeinformationen, mit denen Sie Ihre API-Anfragen authentifizieren AWS können. Die Anmeldedaten müssen die Berechtigung zum Zugriff auf KMS-Schlüssel und Aliase enthalten. Die Berechtigungen werden durch Schlüsselrichtlinien, IAM-Richtlinien, Zuschüsse und kontoübergreifende Zugriffskontrollen bestimmt. Sie können nicht nur den Zugriff auf KMS-Schlüssel steuern, sondern auch den Zugriff auf Ihr CloudHSM und auf Ihre benutzerdefinierten Schlüsselspeicher.

Sie können den `DryRun`-API-Parameter angeben, um zu überprüfen, ob Sie über die erforderlichen Berechtigungen verfügen, um AWS KMS -Schlüssel zu verwenden. Sie können damit auch überprüfen`DryRun`, ob die Anforderungsparameter in einem AWS KMS API-Aufruf korrekt angegeben sind.

Themen

- Was ist der DryRun Parameter?
- DryRun Mit der API spezifizieren

Was ist der DryRun Parameter?

`DryRun` ist ein optionaler API-Parameter, den Sie angeben, um zu überprüfen, ob AWS KMS -API-Aufrufe erfolgreich sind. Verwenden Sie `DryRun`, um Ihren API-Aufruf zu testen, bevor Sie AWS KMS tatsächlich aufrufen. Sie können die folgenden Punkte überprüfen.

- Dass Sie über die erforderlichen Berechtigungen verfügen, um AWS KMS -Schlüssel zu verwenden.
- Dass Sie die Parameter im Aufruf korrekt angegeben haben.

AWS KMS unterstützt die Verwendung des `DryRun` Parameters in bestimmten API-Aktionen:

- CreateGrant
- Decrypt
- DeriveSharedSecret
- Encrypt
- GenerateDataKey
- GenerateDataKeyPair
- GenerateDataKeyPairWithoutPlaintext
- GenerateDataKeyWithoutPlaintext
- GenerateMac
- ReEncrypt
- RetireGrant
- RevokeGrant
- Sign

- Verify
- VerifyMac

Die Verwendung des `DryRun`-Parameters ist kostenpflichtig und wird als Standard-API-Anfrage in Rechnung gestellt. Weitere Informationen zur AWS KMS Preisgestaltung finden Sie unter AWS Key Management Service Preisgestaltung.

Alle API-Anfragen, die den `DryRun`-Parameter verwenden, beziehen sich auf das Anforderungskontingent der API und können zu einer Drosselungsausnahme führen, wenn Sie ein API-Anforderungskontingent überschreiten. Beispielsweise wird der Aufruf von Decrypt mit `DryRun` oder ohne `DryRun` demselben Kontingent für kryptografische Operationen angerechnet. Weitere Informationen hierzu finden Sie unter Drosselung AWS KMS von Anfragen.

Jeder Aufruf eines AWS KMS API-Vorgangs wird als Ereignis erfasst und in einem AWS CloudTrail Protokoll aufgezeichnet. Die Ausgabe aller Operationen, die den `DryRun` Parameter angeben, wird in Ihrem CloudTrail Protokoll angezeigt. Weitere Informationen finden Sie unter AWS KMS API-Aufrufe protokollieren mit AWS CloudTrail.

DryRun Mit der API spezifizieren

Geben Sie zur Verwendung `DryRun` den `–dry-run` Parameter in AWS CLI Befehlen und AWS KMS API-Aufrufen an, die den Parameter unterstützen. Wenn Sie dies tun, AWS KMS wird überprüft, ob Ihr Anruf erfolgreich sein wird. AWS KMS Anrufe, die ich verwende, schlagen immer `DryRun` fehl und geben eine Meldung mit Informationen über den Grund zurück, warum der Anruf fehlgeschlagen ist. Die Nachricht kann die folgenden Ausnahmen enthalten:

- `DryRunOperationException` – Die Anfrage wäre erfolgreich, wenn `DryRun` nicht angegeben wäre.
- `ValidationException` – Die Anfrage schlug fehl, weil ein falscher API-Parameter angegeben wurde.
- `AccessDeniedException` – Sie sind nicht berechtigt, die angegebene API-Aktion auf der KMS-Ressource auszuführen.

Der folgende Befehl verwendet beispielsweise den CreateGrantVorgang und erstellt eine Zuweisung, die es Benutzern, die berechtigt sind, die `keyUserRole` Rolle zu übernehmen, ermöglicht, den Decrypt-Vorgang für einen angegebenen symmetrischen KMS-Schlüssel aufzurufen. Der `DryRun`-Parameter ist angegeben.

```
$  aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \
    --operations Decrypt \
    --dry-run
```

Problembehandlung bei Berechtigungen AWS KMS

Bei der Autorisierung des Zugriffs auf einen KMS-Schlüssel AWS KMS wird Folgendes geprüft:

- Die Schlüsselrichtlinie, die dem KMS-Schlüssel angefügt ist. Die Schlüsselrichtlinie wird immer in der Region AWS-Konto und definiert, der der KMS-Schlüssel gehört.
- Alle IAM-Richtlinien, die dem Benutzer oder der Rolle, die die Anforderung stellt, angefügt sind Die IAM-Richtlinien, die den Gebrauch eines KMS-Schlüssels durch einen Prinzipal regeln, sind immer in dem AWS-Konto des Prinzipals definiert.
- Alle Erteilungen für den KMS-Schlüssel.
- Andere Richtlinientypen, die möglicherweise für die Anforderung zur Verwendung des KMS-Schlüssels gelten, z. B. AWS Organizations -Service-Kontrollrichtlinien und VPC-Endpunktrichtlinien. Diese Richtlinien sind optional und erlauben standardmäßig alle Aktionen. Sie können sie jedoch verwenden, um Berechtigungen zu beschränken, die ansonsten Prinzipalen gewährt werden.

AWS KMS bewertet diese Richtlinienmechanismen gemeinsam, um festzustellen, ob der Zugriff auf den KMS-Schlüssel erlaubt oder verweigert wird. AWS KMS Verwendet dazu ein Verfahren, das dem im folgenden Flussdiagramm dargestellten ähnelt. Das folgende Flussdiagramm bietet eine visuelle Darstellung des Richtlinienauswertungsprozesses.

Dieses Flussdiagramm ist in zwei Teile unterteilt. Die Teile sind sequenziell angeordnet, werden in der Regel aber gleichzeitig ausgewertet.

- Use Authorization (Autorisierung verwenden) legt fest, ob Sie zur Verwendung eines KMS-Schlüssels, basierend auf dessen Schlüsselrichtlinie, IAM-Richtlinien und Erteilungen, berechtigt sind.
- Key trust (Schlüsselbasierte Vertrauensstellung) legt fest, ob Sie einem KMS-Schlüssel vertrauen sollten, den Sie verwenden dürfen. Im Allgemeinen vertrauen Sie den Ressourcen in Ihrem AWS-Konto. Sie können sich aber auch darauf verlassen, KMS-Schlüssel in einer anderen Umgebung zu

verwenden, AWS-Konto wenn eine Grant- oder IAM-Richtlinie in Ihrem Konto die Verwendung des KMS-Schlüssels zulässt.

Aus diesem Flussdiagramm geht hervor, warum einem Aufrufer die Berechtigung für die Verwendung eines KMS-Schlüssels gewährt oder verweigert wurde. Sie können sie auch zur Bewertung Ihrer Richtlinien und Genehmigungen heranziehen. Das Flussdiagramm veranschaulicht beispielsweise, dass einem Aufrufer der Zugriff durch eine explizite `DENY`-Anweisung oder durch das Weglassen einer expliziten `ALLOW`-Anweisung in der Schlüsselrichtlinie, IAM-Richtlinie oder Erteilung verwehrt werden kann.

Das Flussdiagramm kann einige gängige Berechtigungsszenarien erläutern.

Beispiele für Berechtigungen

- Beispiel 1: Dem Benutzer wird der Zugriff auf einen KMS-Schlüssel in seinem AWS-Konto
- Beispiel 2: Der Benutzer übernimmt die Rolle mit der Erlaubnis, einen KMS-Schlüssel in einem anderen System zu verwenden AWS-Konto

Beispiel 1: Dem Benutzer wird der Zugriff auf einen KMS-Schlüssel in seinem AWS-Konto

Alice ist eine IAM-Benutzerin im AWS-Konto 111122223333. Ihr wurde der Zugriff auf einen KMS-Schlüssel im selben AWS-Konto verweigert. Warum kann Alice den KMS-Schlüssel nicht verwenden?

Alice erhält in diesem Fall keinen Zugriff auf den KMS-Schlüssel, da keine Schlüsselrichtlinie, IAM-Richtlinie oder Erteilung vorliegt, die ihr die erforderlichen Berechtigungen erteilen könnte. Die Schlüsselrichtlinie des KMS-Schlüssels ermöglicht die Verwendung von IAM-Richtlinien AWS-Konto zur Steuerung des Zugriffs auf den KMS-Schlüssel, aber keine IAM-Richtlinie gibt Alice die Erlaubnis, den KMS-Schlüssel zu verwenden.

Beachten Sie die relevanten Richtlinien für dieses Beispiel.

- Der KMS-Schlüssel, den Alice verwenden möchte, untersteht der Standardschlüsselrichtlinie. Diese Richtlinie erlaubt es dem AWS-Konto, das den KMS-Schlüssel besitzt, IAM-Richtlinien zum Steuern des Zugriffs auf den KMS-Schlüssel zu verwenden. Dieser Schlüssel erfüllt die im Abschnitt ERLAUBT die Schlüsselrichtlinie dem Aufruferkonto die Verwendung von IAM-Richtlinien zur Kontrolle des Zugriffs auf den Schlüssel? des Flussdiagramms beschriebene Bedingung.

JSON

```
{
  "Version" : "2012-10-17",
  "Id" : "key-test-1",
  "Statement" : [ {
    "Sid" : "DelegateToIAMPolicies",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : "kms:*",
    "Resource" : "*"
  } ]
}
```

- Keine Schlüsselrichtlinie, IAM-Richtlinie oder Erteilung erteilt Alice jedoch die Berechtigung zur Verwendung des KMS-Schlüssels. Daher wird Alice die Berechtigung zur Verwendung des KMS-Schlüssels verweigert.

Beispiel 2: Der Benutzer übernimmt die Rolle mit der Erlaubnis, einen KMS-Schlüssel in einem anderen System zu verwenden AWS-Konto

Bob ist ein Benutzer in Konto 1 (111122223333). Er ist berechtigt, in Konto 2 (444455556666) einen KMS-Schlüssel in kryptografischen Produktionen zu verwenden. Wie ist das möglich?

> **Tip**
>
> Denken Sie beim Auswerten von kontoübergreifenden Berechtigungen daran, dass die Schlüsselrichtlinie im Konto des KMS-Schlüssels angegeben wird. Die IAM-Richtlinie wird im Konto des Aufrufers angegeben, auch wenn sich der Aufrufer in einem anderen Konto befindet. Weitere Informationen zum Bereitstellen des kontoübergreifenden Zugriffs auf KMS-Schlüssel finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

- Die Schlüsselrichtlinie für den KMS-Schlüssel in Konto 2 erlaubt Konto 2 die Verwendung von IAM-Richtlinien zum Steuern des Zugriffs auf den KMS-Schlüssel.

- Die Schlüsselrichtlinie für den KMS-Schlüssel in Konto 2 erlaubt Konto 1 die Verwendung des KMS-Schlüssels in kryptografischen Produktionen. Konto 1 muss jedoch seinen Prinzipalen anhand von IAM-Richtlinien Zugriff auf den KMS-Schlüssel gewähren.
- Eine IAM-Richtlinie in Konto 1 erlaubt es der `Engineering`-Rolle, den KMS-Schlüssel in Konto 2 für kryptografische Produktionen zu verwenden.
- Bob, ein Benutzer in Konto 1, verfügt über die Berechtigung, die `Engineering`-Rolle zu übernehmen.
- Bob kann diesem KMS-Schlüssel vertrauen, denn obwohl sich dieser nicht in seinem Konto befindet, ist es ihm über eine IAM-Richtlinie in seinem Konto explizit erlaubt, diesen KMS-Schlüssel zu verwenden.

Betrachten wir einmal die Richtlinien, die es Bob, einem Benutzer in Konto 1, ermöglichen, den KMS-Schlüssel in Konto 2 zu verwenden.

- Die Schlüsselrichtlinie für den KMS-Schlüssel erlaubt es Konto 2 (444455556666, das Konto, das den KMS-Schlüssel besitzt), den Zugriff auf den KMS-Schlüssel anhand von IAM-Richtlinien zu steuern. Diese Schlüsselrichtlinie erlaubt außerdem Konto 1 (111122223333) die Verwendung des KMS-Schlüssels in kryptografischen Produktionen (angegeben im `Action`-Element der Richtlinienanweisung). Niemand in Konto 1 kann jedoch den KMS-Schlüssel in Konto 2 verwenden, sofern in Konto 1 keine IAM-Richtlinien definiert sind, die den Prinzipalen Zugriff auf den KMS-Schlüssel gewähren.

  Im Flussdiagramm erfüllt diese Schlüsselrichtlinie in Konto 2 die Bedingung Does the key policy ALLOW the caller's account to use IAM policies to control access to the key? (ERLAUBT die Schlüsselrichtlinie dem Aufrufer-Konto die Verwendung von IAM-Richtlinien zur Steuerung des Zugriffs auf den Schlüssel?)

  JSON

```
{
    "Id": "key-policy-acct-2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionUseIAMpolicies",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "AllowAccount1UseThisKMSKeys",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo",
                "kms:GenerateDataKey",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        }
    ]
}
```

- Eine IAM-Richtlinie im Konto des Anrufers AWS-Konto (Konto 1, 111122223333) erteilt dem Prinzipal die Erlaubnis, kryptografische Operationen mithilfe des KMS-Schlüssels in Konto 2 (444455556666) durchzuführen. Das `Action`-Element delegiert dem Prinzipal die gleichen Berechtigungen, die Konto 1 durch die Schlüsselrichtlinie in Konto 2 erhalten hat. Um diese Berechtigung für die `Engineering`-Rolle in Konto 1, ist diese Inline-Richtlinie eingebettet in der `Engineering`-Rolle.

  Kontoübergreifende IAM-Richtlinien wie diese werden erst dann wirksam, wenn die Schlüsselrichtlinie für den KMS-Schlüssel in Konto 2 dem Konto 1 die Berechtigung zur Verwendung des KMS-Schlüssels erteilt. Konto 1 kann seinen Prinzipalen außerdem nur Berechtigungen zum Ausführen von Aktionen erteilen, die dem Konto über die Schlüsselrichtlinie gewährt wurden.

  Im Flussdiagramm erfüllt dies die Bedingung Does an IAM policy allow the caller to perform this action? (Erlaubt eine IAM-Richtlinie dem Aufrufer die Ausführung dieser Aktion?).

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo",
                "kms:GenerateDataKey",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:DescribeKey"
            ],
            "Resource": [
            "arn:aws:kms:us-
west-2:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            ]
        }
    ]
}
```

- Das letzte erforderliche Element ist die Definition der `Engineering`-Rolle in Konto 1. Das `AssumeRolePolicyDocument` in der Rolle erlaubt Bob, die `Engineering`-Rolle zu übernehmen.

```
{
    "Role": {
        "Arn": "arn:aws:iam::111122223333:role/Engineering",
        "CreateDate": "2019-05-16T00:09:25Z",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": {
                "Principal": {
                    "AWS": "arn:aws:iam::111122223333:user/bob"
                },
                "Effect": "Allow",
                "Action": "sts:AssumeRole"
            }
        },
        "Path": "/",
        "RoleName": "Engineering",
```

```
        "RoleId": "AROA4KJY2TU23Y7NK62MV"
    }
}
```

AWS KMS Glossar zur Zugriffskontrolle

Im folgenden Thema werden wichtige Begriffe und Konzepte der AWS KMS Zugriffskontrolle beschrieben.

Authentifizierung

Authentifizierung ist der Prozess der Überprüfung Ihrer Identität. Um eine Anfrage an zu senden AWS KMS, müssen Sie sich AWS mit Ihren AWS Anmeldeinformationen anmelden.

Autorisierung

Die Autorisierung gewährt die Erlaubnis, Anfragen zum Erstellen, Verwalten oder Verwenden von AWS KMS Ressourcen zu senden. Sie müssen beispielsweise autorisiert sein, um einen KMS-Schlüssel in einer kryptografischen Operation zu verwenden.

Verwenden Sie wichtige Richtlinien, IAM-Richtlinien und Zuschüsse, um den Zugriff auf Ihre AWS KMS Ressourcen zu kontrollieren. Jeder KMS-Schlüssel muss über eine Schlüsselrichtlinie verfügen. Wenn die Schlüsselrichtlinie es zulässt, können Sie auch IAM-Richtlinien und Erteilungen verwenden, um Prinzipalen Zugriff auf den KMS-Schlüssel zu geben. Um Ihre Autorisierung zu verfeinern, können Sie Bedingungsschlüssel verwenden, die den Zugriff nur dann erlauben oder verweigern, wenn eine Anforderung oder Ressource die von Ihnen festgelegten Bedingungen erfüllt. Sie können den Zugriff für Prinzipale gewähren, denen Sie in anderen AWS-Konten-Konten vertrauen.

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen

Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode für die Selbstsignierung von Anforderungen finden Sie unter AWS Signature Version 4 für API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen bereitstellen. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und AWS Multi-Faktor-Authentifizierung (MFA) in IAM im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Der Zugriff erfolgt, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen. Verwenden Sie diese nur, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem

beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise eine Gruppe benennen IAMAdminsund dieser Gruppe Berechtigungen zur Verwaltung von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb von Ihnen AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, jedoch nicht mit einer bestimmten Person verknüpft. Um vorübergehend eine IAM-Rolle in der zu übernehmen AWS Management

Console, können Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Methoden für die Übernahme einer Rolle im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige AWS-Services verwenden Funktionen in anderen AWS-Services. Wenn Sie beispielsweise einen Service aufrufen, ist es üblich, dass dieser Service Anwendungen in Amazon ausführt EC2 oder Objekte in Amazon S3 speichert. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen

mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Auf Amazon ausgeführte Anwendungen EC2 — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2 Instance ausgeführt werden und AWS API-Anfragen stellen AWS CLI . Dies ist dem Speichern von Zugriffsschlüsseln innerhalb der EC2 Instance vorzuziehen. Um einer EC2 Instanz eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instanzprofil, das an die Instanz angehängt ist. Ein Instanzprofil enthält die Rolle und ermöglicht Programmen, die auf der EC2 Instanz ausgeführt werden, temporäre Anmeldeinformationen abzurufen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Verwenden einer IAM-Rolle, um Berechtigungen für Anwendungen zu gewähren, die auf EC2 Amazon-Instances ausgeführt werden.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Die Berechtigungen in den Richtlinien legen fest, ob eine Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer Inline-Richtlinie wählen, finden Sie unter Auswählen zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Eine AWS KMS Schlüsselrichtlinie ist eine ressourcenbasierte Richtlinie, die den Zugriff auf einen KMS-Schlüssel steuert. Jeder KMS-Schlüssel muss über eine Schlüsselrichtlinie verfügen. Sie können einen anderen Autorisierungsmechanismus verwenden, um den Zugriff auf den KMS-Schlüssel zu ermöglichen, jedoch nur, wenn die Schlüsselrichtlinie dies zulässt. (Sie können eine

IAM-Richtlinie verwenden, um den Zugriff auf einen KMS-Schlüssel zu verweigern, auch wenn die Schlüsselrichtlinie dies nicht explizit zulässt.)

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource wie einen KMS-Schlüssel anfügen. Die ressourcenbasierte Richtlinie definiert die Aktionen, die ein bestimmter Prinzipal auf dieser Ressource durchführen kann und unter welchen Bedingungen. Sie geben die Ressource nicht in einer ressourcenbasierten Richtlinie an, sondern Sie müssen einen Prinzipal angeben, z. B. Konten, Benutzer, Rollen, Verbundbenutzer oder. AWS-Services Ressourcenbasierte Richtlinien sind Richtlinien, die sich in dem Service befinden, der die Ressource verwaltet. Sie können AWS verwaltete Richtlinien von IAM, wie z. B. die `AWSKeyManagementServicePowerUser`verwaltete Richtlinie, nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Dienststeuerungsrichtlinien (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie alle Funktionen in einer Organisation aktivieren, können Sie Richtlinien zur Servicesteuerung (SCPs) auf einige oder alle Ihre Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Root-Benutzer des AWS-Kontos Entitäten. Weitere Informationen zu Organizations und SCPs finden Sie unter Richtlinien zur Servicesteuerung im AWS Organizations Benutzerhandbuch.

- Ressourcenkontrollrichtlinien (RCPs) — RCPs sind JSON-Richtlinien, mit denen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten festlegen können, ohne die IAM-Richtlinien aktualisieren zu müssen, die jeder Ressource zugeordnet sind, deren Eigentümer Sie sind. Das RCP schränkt die Berechtigungen für Ressourcen in Mitgliedskonten ein und kann sich auf die effektiven Berechtigungen für Identitäten auswirken, einschließlich der Root-Benutzer des AWS-Kontos, unabhängig davon, ob sie zu Ihrer Organisation gehören. Weitere Informationen zu Organizations RCPs, einschließlich einer Liste AWS-Services dieser Support-Leistungen RCPs, finden Sie unter Resource Control Policies (RCPs) im AWS Organizations Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

AWS KMS Ressourcen

AWS KMS In ist die primäre Ressource ein AWS KMS key. AWS KMS unterstützt auch einen Alias, eine unabhängige Ressource, die einen benutzerfreundlichen Namen für einen KMS-Schlüssel bereitstellt. Bei einigen AWS KMS Vorgängen können Sie einen Alias verwenden, um einen KMS-Schlüssel zu identifizieren.

Jede Instance eines KMS-Schlüssels oder Aliases hat einen eindeutigen Amazon-Ressourcennamen (ARN) mit einem Standardformat. In AWS KMS Ressourcen lautet der AWS Dienstname`kms`.

- AWS KMS key

  ARN-Format:

```
arn:AWS partition name:AWS service name:AWS-Region:AWS-Konto ID:key/key
ID
```

Beispiel-ARN:

```
arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

- Alias

  ARN-Format:

  ```
  arn:AWS partition name:AWS service name:AWS-Region:AWS-Konto
  ID:alias/alias name
  ```

  Beispiel-ARN:

  ```
  arn:aws:kms:us-west-2:111122223333:alias/example-alias
  ```

AWS KMS bietet eine Reihe von API-Operationen für die Arbeit mit Ihren AWS KMS Ressourcen. Weitere Informationen zur Identifizierung von KMS-Schlüsseln in den AWS Management Console und AWS KMS API-Vorgängen finden Sie unterSchlüsselkennungen () KeyId. Eine Liste der AWS KMS Operationen finden Sie in der AWS Key Management Service API-Referenz.

Erstellen Sie einen KMS-Schlüssel

Sie können AWS KMS keys in oder mithilfe des AWS Management ConsoleCreateKeyVorgangs oder der AWS::KMS::Key AWS CloudFormation Ressource erstellen. Während dieses Vorgangs legen Sie die Schlüsselrichtlinie für den KMS-Schlüssel fest, die Sie jederzeit ändern können. Sie wählen auch die folgenden Werte aus, die den Typ des KMS-Schlüssels definieren, den Sie erstellen. Diese Eigenschaften können nicht geändert werden, nachdem der KMS-Schlüssel erstellt wurde.

KMS-Schlüsseltyp

Der Schlüsseltyp ist eine Eigenschaft, die bestimmt, welcher Typ von kryptografischem Schlüssel erstellt wird. AWS KMS bietet drei Schlüsseltypen zum Schutz von Daten:

- Symmetrische Schlüssel nach dem Advanced Encryption Standard (AES)

  256-Bit-Schlüssel, die im Galois Counter Mode (GCM) -Modus von AES verwendet werden, um authentifizierte Daten mit einer Größe von weniger als 4 KB encryption/decryption bereitzustellen. Dies ist der gängigste Schlüsseltyp und wird verwendet, um andere Datenverschlüsselungsschlüssel zu schützen, die in Ihren Anwendungen verwendet werden, und AWS-Services damit Ihre Daten in Ihrem Namen zu verschlüsseln.

- RSA-, elliptische Kurve- oder SM2 (nur China Regionen) asymmetrische Schlüssel

  Diese Schlüssel sind in verschiedenen Größen erhältlich und unterstützen viele Algorithmen. Sie können je nach ausgewähltem Algorithmus zur Ver- und Entschlüsselung, zum Signieren und Überprüfen oder zum Ableiten von Operationen mit gemeinsamen Geheimnissen verwendet werden.

- Symmetrische Schlüssel für die Durchführung von Hash-basierten HMAC-Vorgängen (Message Authentication Codes)

  Bei diesen Schlüsseln handelt es sich um 256-Bit-Schlüssel, die für Signier- und Überprüfungsvorgänge verwendet werden.

  KMS-Schlüssel können nicht im Klartext aus dem Dienst exportiert werden. Sie werden von den vom Dienst verwendeten Hardware-Sicherheitsmodulen (HSMs) generiert und können nur innerhalb dieser verwendet werden. Dies ist die grundlegende Sicherheitseigenschaft, mit der sichergestellt werden AWS KMS soll, dass Schlüssel nicht kompromittiert werden.

Verwendung von Schlüsseln

Die Schlüsselnutzung ist eine Eigenschaft, die bestimmt, welche kryptografischen Vorgänge der Schlüssel unterstützt. KMS-Schlüssel können eine Schlüsselverwendung von`ENCRYPT_DECRYPT`, `SIGN_VERIFYGENERATE_VERIFY_MAC`, oder haben`KEY_AGREEMENT`. Jeder KMS-Schlüssel kann nur eine Schlüsselnutzung haben. Dabei werden bewährte Verfahren zur Schlüsselnutzung gemäß der Sonderveröffentlichung 800-57 des National Institute of Standards and Technology (NIST), Empfehlungen für die Schlüsselverwaltung, Abschnitt 5.2, Schlüsselnutzung, eingehalten. Die Verwendung eines KMS-Schlüssels für mehr als eine Art von Operation macht das Produkt beider Operationen anfälliger gegenüber Angriffen.

Wichtige technische Daten

Die Schlüsselspezifikation ist eine Eigenschaft, die die kryptografische Konfiguration des KMS-Schlüssels repräsentiert. Die Bedeutung der Schlüsselspezifikation unterscheidet sich vom Schlüsseltyp.

Bei KMS-Schlüsseln bestimmt die Schlüsselspezifikation, ob der KMS-Schlüssel symmetrisch oder asymmetrisch ist. Sie bestimmt auch die Art des Schlüsselmaterials und die unterstützten Algorithmen.

Die Standardschlüsselspezifikation, SYMMETRIC_DEFAULT, steht für einen symmetrischen 256-Bit-Verschlüsselungsschlüssel. Eine ausführliche Beschreibung aller unterstützten Schlüsselspezifikationen finden Sie unter. Referenz zu wichtigen Spezifikationen

Herkunft der wichtigsten Materialien

Ursprung des Schlüsselmaterials ist eine KMS-Schlüssel-Eigenschaft, die die Quelle des Schlüsselmaterials im KMS-Schlüssel identifiziert. Sie wählen den Ursprung des Schlüsselmaterials, wenn Sie den KMS-Schlüssel erstellen. Danach kann er nicht mehr geändert werden. Die Quelle des Schlüsselmaterials wirkt sich auf die Sicherheit, Dauerhaftigkeit, Verfügbarkeit, Latenzzeit und Durchsatzmerkmale des KMS-Schlüssels aus.

Jeder KMS-Schlüssel enthält in seinen Metadaten einen Verweis auf sein Schlüsselmaterial. Die Herkunft des Schlüsselmaterials des KMS-Schlüssels mit symmetrischer Verschlüsselung kann variieren. Sie können Schlüsselmaterial verwenden, das AWS KMS generiert wird, Schlüsselmaterial, das in einem benutzerdefinierten Schlüsselspeicher generiert wurde, oder Ihr eigenes Schlüsselmaterial importieren.

Standardmäßig verfügt jeder KMS-Schlüssel über eindeutiges Schlüsselmaterial. Sie können jedoch eine Reihe von multiregionalen Schlüsseln mit demselben Schlüsselmaterial erstellen.

KMS-Schlüssel können einen der folgenden wichtigen Materialursprungswerte haben:`AWS_KMS`, `EXTERNAL` (importiertes Schlüsselmaterial), `AWS_CLOUDHSM` (AWS CloudHSM KMS-Schlüssel in einem Schlüsselspeicher) oder `EXTERNAL_KEY_STORE` (KMS-Schlüssel in einem externen Schlüsselspeicher).

Themen

- Berechtigungen zum Erstellen von KMS-Schlüsseln
- Wählen Sie aus, welche Art von KMS-Schlüssel erstellt werden soll
- Erstellen Sie einen KMS-Schlüssel für die symmetrische Verschlüsselung
- Erstellen eines asymmetrischen KMS-Schlüssels
- Erstellen Sie einen HMAC-KMS-Schlüssel
- Primärschlüssel für mehrere Regionen erstellen
- Erstellen Sie Replikatschlüssel für mehrere Regionen
- Erstellen Sie einen KMS-Schlüssel mit importiertem Schlüsselmaterial
- Erstellen Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher
- Erstellen Sie einen KMS-Schlüssel in externen Schlüsselspeichern

Berechtigungen zum Erstellen von KMS-Schlüsseln

Um einen KMS-Schlüssel in der Konsole oder mithilfe von zu erstellen APIs, benötigen Sie die folgenden Berechtigungen in einer IAM-Richtlinie. Wenn möglich, verwenden Sie Bedingungsschlüssel, um die Berechtigungen einzuschränken. Sie können beispielsweise den KeySpec Bedingungsschlüssel kms: in einer IAM-Richtlinie verwenden, um es Prinzipalen zu ermöglichen, nur symmetrische Verschlüsselungsschlüssel zu erstellen.

Ein Beispiel für eine IAM-Richtlinie für Prinzipale, die Schlüssel erstellen, finden Sie unter Einem Benutzer das Erstellen von KMS-Schlüsseln erlauben.

> **Note**
>
> Seien Sie vorsichtig, wenn Sie Prinzipalen die Berechtigung zum Verwalten von Tags und Aliasen erteilen. Wenn Sie einen Tag oder einen Alias ändern, wird dadurch die Berechtigung für den kundenverwalteten Schlüssel erteilt oder verweigert. Details hierzu finden Sie unter ABAC für AWS KMS.

- kms: CreateKey ist erforderlich.
- kms: CreateAlias ist erforderlich, um einen KMS-Schlüssel in der Konsole zu erstellen, wobei für jeden neuen KMS-Schlüssel ein Alias erforderlich ist.
- kms: TagResource ist erforderlich, um beim Erstellen des KMS-Schlüssels Tags hinzuzufügen.
- iam: CreateServiceLinkedRole ist erforderlich, um Primärschlüssel für mehrere Regionen zu erstellen. Details hierzu finden Sie unter Steuern Sie den Zugriff auf Schlüssel für mehrere Regionen.

Die kms: PutKeyPolicy -Berechtigung ist nicht erforderlich, um den KMS-Schlüssel zu erstellen. Die `kms:CreateKey`-Berechtigung enthält die Berechtigung zum Festlegen der ursprünglichen Schlüsselrichtlinie. Sie müssen diese Berechtigung jedoch der Schlüsselrichtlinie hinzufügen, während Sie den KMS-Schlüssel erstellen, um sicherzustellen, dass Sie den Zugriff auf den KMS-Schlüssel steuern können. Die Alternative besteht darin, den BypassLockoutSafetyCheckParameter zu verwenden, was nicht empfohlen wird.

KMS-Schlüssel gehören zu dem AWS Konto, in dem sie erstellt wurden. Der IAM-Benutzer, der einen KMS-Schlüssel erstellt, gilt nicht als Schlüsselbesitzer und hat nicht automatisch die Berechtigung, den von ihm erstellten KMS-Schlüssel zu verwenden oder zu verwalten. Wie jeder andere Prinzipal muss der Schlüsselersteller eine Berechtigung über eine Schlüsselrichtlinie, IAM-Richtlinie oder Erteilung erhalten. Prinzipale, die die `kms:CreateKey`-Berechtigung haben, können jedoch die ursprüngliche Schlüsselrichtlinie festlegen und sich selbst die Berechtigung zur Verwendung oder Verwaltung des Schlüssels erteilen.

Wählen Sie aus, welche Art von KMS-Schlüssel erstellt werden soll

Der Typ des KMS-Schlüssels, den Sie erstellen, hängt weitgehend davon ab, wie Sie den KMS-Schlüssel verwenden möchten, von Ihren Sicherheitsanforderungen und Ihren Autorisierungsanforderungen. Der Schlüsseltyp und die Schlüsselverwendung eines KMS-Schlüssels bestimmen, welche kryptografischen Operationen der Schlüssel ausführen kann. Jeder KMS-Schlüssel kann nur eine Schlüsselverwendung haben. Die Verwendung eines KMS-Schlüssels für mehr als eine Art von Operation macht das Produkt aller Operationen anfälliger gegenüber Angriffen.

Verwenden Sie den KeyUsage Bedingungsschlüssel kms:, damit Prinzipale KMS-Schlüssel nur für eine bestimmte Schlüsselverwendung erstellen können. Sie können mit dem Bedingungsschlüssel `kms:KeyUsage` Prinzipalen auch, basierend auf seiner Schlüsselnutzung, den Aufruf von API-Operationen für einen KMS-Schlüssel gewähren. Beispielsweise können Sie die Berechtigung zum

Deaktivieren eines KMS-Schlüssels nur dann zulassen, wenn die Schlüsselnutzung SIGN_VERIFY lautet.

Verwenden Sie die folgende Anleitung, um zu bestimmen, welchen Typ von KMS-Schlüssel Sie für Ihren Anwendungsfall benötigen.

Verschlüsseln und Entschlüsseln von Daten

Verwenden Sie einen symmetrischen KMS-Schlüssel für die meisten Anwendungsfälle, bei denen Daten verschlüsselt und entschlüsselt werden müssen. Der symmetrische Verschlüsselungsalgorithmus, den AWS KMS verwendet, ist schnell, effizient und gewährleistet die Vertraulichkeit und Authentizität von Daten. Er unterstützt authentifizierte Verschlüsselung mit zusätzlichen authentifizierten Daten (AAD), die als Verschlüsselungskontext definiert sind. Bei dieser Art von KMS-Schlüssel müssen sowohl der Absender als auch der Empfänger verschlüsselter Daten über gültige AWS Anmeldeinformationen für den Anruf AWS KMS verfügen.

Wenn Ihr Anwendungsfall eine Verschlüsselung außerhalb AWS von Benutzern erfordert, die nicht anrufen können AWS KMS, sind asymmetrische KMS-Schlüssel eine gute Wahl. Sie können den öffentlichen Schlüssel des asymmetrischen KMS-Schlüssels verteilen, damit diese Benutzer Daten verschlüsseln können. Ihre Anwendungen, die diese Daten entschlüsseln müssen, können den privaten Schlüssel des asymmetrischen KMS-Schlüssels innerhalb von AWS KMS verwenden.

Signieren von Nachrichten und Verifizieren von Signaturen

Um Nachrichten zu signieren und Signaturen zu überprüfen, müssen Sie einen asymmetrischen KMS-Schlüssel verwenden. Sie können einen KMS-Schlüssel mit einer Schlüsselspezifikation verwenden, die ein RSA-Schlüsselpaar, ein ECC-Schlüsselpaar (Elliptic Curve), ein ML-DSA-Schlüsselpaar oder ein SM2 key pair (nur Regionen China) darstellt. Die ausgewählte Schlüsselspezifikation wird durch den Signaturalgorithmus bestimmt, den Sie verwenden möchten. Die ECDSA-Signaturalgorithmen, die von ECC-Schlüsselpaaren unterstützt werden, sind den RSA-Signaturalgorithmen vorzuziehen. Verwenden Sie ein ML-DSA-Schlüsselpaar, wenn Sie von RSA- oder ECC-Schlüsseln zu Post-Quantum-Schlüsseln migrieren. Möglicherweise müssen Sie jedoch eine bestimmte Schlüsselspezifikation und einen bestimmten Signaturalgorithmus verwenden, um Benutzer zu unterstützen, die Signaturen außerhalb von AWSüberprüfen.

Verschlüsseln Sie mit asymmetrischen Schlüsselpaaren

Um Daten mit einem asymmetrischen key pair zu verschlüsseln, müssen Sie einen asymmetrischen KMS-Schlüssel mit einer RSA-Schlüsselspezifikation oder einer SM2

Schlüsselspezifikation verwenden (nur Regionen China). Um Daten in AWS KMS mit dem öffentlichen Schlüssel eines KMS-Schlüsselpaars zu verschlüsseln, verwenden Sie die Encrypt-Operation. Sie können den öffentlichen Schlüssel auch herunterladen und ihn mit den Parteien teilen, die Daten außerhalb von verschlüsseln müssen. AWS KMS

Wenn Sie den öffentlichen Schlüssel eines asymmetrischen KMS-Schlüssels herunterladen, können Sie ihn außerhalb von AWS KMS verwenden. Er unterliegt jedoch nicht mehr den Sicherheitskontrollen, die den KMS-Schlüssel schützen. AWS KMS Sie können beispielsweise keine AWS KMS Schlüsselrichtlinien oder Zuschüsse verwenden, um die Verwendung des öffentlichen Schlüssels zu kontrollieren. Sie können auch nicht kontrollieren, ob der Schlüssel nur für die Verschlüsselung und Entschlüsselung verwendet wird, indem Sie die Verschlüsselungsalgorithmen verwenden, die ihn AWS KMS unterstützen. Weitere Informationen finden Sie unter Besondere Überlegungen zum Herunterladen öffentlicher Schlüssel.

Um Daten zu entschlüsseln, die mit dem öffentlichen Schlüssel außerhalb von verschlüsselt wurden AWS KMS, rufen Sie den Vorgang Decrypt auf. Der `Decrypt` Vorgang schlägt fehl, wenn die Daten unter einem öffentlichen Schlüssel aus einem KMS-Schlüssel mit der Schlüsselverwendung von verschlüsselt wurden. `SIGN_VERIFY` Der Vorgang schlägt auch fehl, wenn er mit einem Algorithmus verschlüsselt wurde, der die von Ihnen gewählte Schlüsselspezifikation AWS KMS nicht unterstützt. Weitere Informationen zu wichtigen Spezifikationen und unterstützten Algorithmen finden Sie unterReferenz zu wichtigen Spezifikationen.

Um diese Fehler zu vermeiden, AWS KMS muss jeder, der einen öffentlichen Schlüssel außerhalb von verwendet, die Schlüsselkonfiguration speichern. Die AWS KMS Konsole und die GetPublicKeyAntwort enthalten die Informationen, die Sie angeben müssen, wenn Sie den öffentlichen Schlüssel teilen.

Leitet gemeinsame Geheimnisse ab

Um gemeinsame geheime Schlüssel abzuleiten, verwenden Sie einen KMS-Schlüssel mit von NIST empfohlenem elliptischem Kurvenmaterial oder SM2(nur Regionen China) Schlüsselmaterial. AWS KMS verwendet den Elliptic Curve Cryptography Cofactor Diffie-Hellman Primitive (ECDH), um eine wichtige Vereinbarung zwischen zwei Peers herzustellen, indem ein gemeinsames Geheimnis aus ihren öffentlich-privaten Schlüsselpaaren mit elliptischer Kurve abgeleitet wird. Sie können den unverarbeiteten gemeinsamen geheimen Schlüssel, den der DeriveSharedSecretVorgang zurückgibt, verwenden, um einen symmetrischen Schlüssel abzuleiten, der Daten, die zwischen zwei Parteien gesendet werden, ver- und entschlüsseln oder generieren und verifizieren kann. HMACs AWS KMS empfiehlt, dass Sie die Empfehlungen

von NIST zur Schlüsselableitung befolgen, wenn Sie das unformatierte gemeinsame geheime Geheimnis zur Ableitung eines symmetrischen Schlüssels verwenden.

Generieren und überprüfen von HMAC-Codes.

Verwenden Sie einen HMAC-KMS-Schlüssel, um Hash-basierte Nachrichtenauthentifizierungscodes zu generieren und zu überprüfen. Wenn Sie einen HMAC-Schlüssel erstellen AWS KMS, AWS KMS erstellt und schützt es Ihr Schlüsselmaterial und stellt sicher, dass Sie die richtigen MAC-Algorithmen für Ihren Schlüssel verwenden. HMAC-Codes können auch als Pseudozufallszahlen und in bestimmten Szenarien zum symmetrischen Signieren und Tokenisieren verwendet werden.

HMAC-KMS-Schlüssel sind symmetrische Schlüssel. Beim Erstellen eines HMAC-KMS-Schlüssels in der AWS KMS -Konsole wählen Sie den Schlüsseltyp `Symmetric`.

Zusammen mit AWS Diensten verwenden

Informationen zum Erstellen eines KMS-Schlüssels für die Verwendung mit einem AWS Dienst, der in integriert ist AWS KMS, finden Sie in der Dokumentation des Dienstes. AWS Dienste, die Ihre Daten verschlüsseln, benötigen einen symmetrischen KMS-Schlüssel.

Zusätzlich zu diesen Überlegungen haben kryptografische Operationen zu KMS-Schlüsseln mit unterschiedlichen Schlüsselspezifikationen unterschiedliche Preise und unterschiedliche Anforderungskontingente. Informationen zu AWS KMS -Preisen erhalten Sie unter AWS Key Management Service Pricing (Preise für WAF). Weitere Informationen zu Anforderungskontingenten finden Sie unter Anforderungskontingente.

Erstellen Sie einen KMS-Schlüssel für die symmetrische Verschlüsselung

In diesem Thema wird erklärt, wie Sie den grundlegenden KMS-Schlüssel erstellen, einen KMS-Schlüssel mit symmetrischer Verschlüsselung für eine einzelne Region mit Schlüsselmaterial von. AWS KMS Sie können diesen KMS-Schlüssel verwenden, um Ihre Ressourcen in einer AWS-Service zu schützen.

Sie können KMS-Schlüssel mit symmetrischer Verschlüsselung in der AWS KMS Konsole, mithilfe der CreateKeyAPI oder mithilfe der AWS::KMS::Key AWS CloudFormation Vorlage erstellen.

Die Standard-Schlüsselspezifikation, SYMMETRIC_DEFAULT, ist die Schlüsselspezifikation für KMS-Schlüssel mit symmetrischer Verschlüsselung. Wenn Sie in der Konsole den symmetrischen

Schlüsseltyp und die Verwendung des Schlüssels Verschlüsseln und Entschlüsseln auswählen, wird die Schlüsselspezifikation ausgewählt. AWS KMS `SYMMETRIC_DEFAULT` Wenn Sie bei der CreateKeyOperation keinen `KeySpec` Wert angeben, wird SYMMETRIC_DEFAULT ausgewählt. Wenn Sie keinen Grund haben, eine andere Schlüsselspezifikation zu verwenden, ist SYMMETRIC_DEFAULT eine gute Wahl.

Weitere Informationen zu Kontingenten, die für KMS-Schlüssel gelten, finden Sie unter Kontingente.

Verwenden der Konsole AWS KMS

Sie können das AWS Management Console zum Erstellen AWS KMS keys (KMS-Schlüssel) verwenden.

> **Important**
>
> Nehmen Sie keine vertraulichen oder sensiblen Informationen in den Alias, in der Beschreibung oder in den Tags auf. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Klicken Sie auf Create key.
5. Um einen KMS-Schlüssel mit symmetrischer Verschlüsselung zu erstellen, wählen Sie für Key type (Schlüsseltyp) die Option Symmetric (Symmetrisch) aus.
6. Unter Key usage (Schlüsselverwendung) ist die Option Encrypt and decrypt (Verschlüsseln und Entschlüsseln) für Sie ausgewählt.
7. Wählen Sie Weiter aus.
8. Geben Sie einen Alias für den Replikatschlüssel ein. Der Aliasname darf nicht mit **`aws/`** beginnen. Das **`aws/`** Präfix ist von Amazon Web Services reserviert, um es Von AWS verwaltete Schlüssel in Ihrem Konto darzustellen.

> **Note**
>
> Wenn Sie einen Alias hinzufügen, löschen oder aktualisieren, wird dadurch möglicherweise eine Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Details dazu finden Sie unter ABAC für AWS KMS und Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern.

Ein Alias ist ein Anzeigename, den Sie verwenden können, um einen KMS-Schlüssel zu identifizieren. Wir empfehlen, dass Sie einen Alias wählen, der auf die Art von Daten, die Sie schützen möchten, oder die Anwendung, die Sie mit dem KMS-Schlüssel verwenden möchten, hindeutet.

Zum Erstellen eines KMS-Schlüssels in der Konsole benötigen Sie Aliase AWS Management Console. Sie sind optional, wenn Sie den CreateKeyVorgang verwenden.

9. (Optional) Geben Sie eine Beschreibung für den KMS-Schlüssel ein.

   Sie können jetzt eine Beschreibung hinzufügen oder sie jederzeit aktualisieren, es sei denn, der Schlüsselstatus lautet `Pending Deletion` oder `Pending Replica Deletion`. Um die Beschreibung eines vorhandenen, vom Kunden verwalteten Schlüssels hinzuzufügen, zu ändern oder zu löschen, bearbeiten Sie die Beschreibung auf der Detailseite des KMS-Schlüssels AWS Management Console oder verwenden Sie den UpdateKeyDescriptionVorgang.

10. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Add tag (Tag hinzufügen), wenn Sie mehr als ein Tag zum KMS-Schlüssel hinzufügen möchten.

    > **Note**
    >
    > Wenn Sie einen KMS-Schlüssel markieren oder entmarkieren, wird dadurch möglicherweise die Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Details dazu finden Sie unter ABAC für AWS KMS und Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern.

    Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenverteilungsbericht generiert, in dem die Nutzung und die Kosten nach Stichwörtern zusammengefasst sind. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu

steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Schlagworte in AWS KMS und ABAC für AWS KMS.

11. Wählen Sie Weiter aus.
12. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel verwalten können.

> **Hinweise**
>
> Diese Schlüsselrichtlinie ermöglicht die AWS-Konto vollständige Kontrolle über diesen KMS-Schlüssel. Kontoadministratoren können damit anderen Prinzipalen mithilfe von IAM-Richtlinien die Berechtigung zum Verwalten des KMS-Schlüssels erteilen. Details hierzu finden Sie unter the section called “Standardschlüsselrichtlinie”.
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter der Anweisungs-ID wichtige Administratoren hinzu`"Allow access for Key Administrators"`. Eine Änderung dieser Anweisungs-ID kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Anweisung vornehmen.

13. (Optional) Um zu verhindern, dass die ausgewählten IAM-Benutzer und -Rollen diesen KMS-Schlüssel löschen, deaktivieren Sie unten auf der Seite im Abschnitt Key deletion (Schlüssellöschung) das Kontrollkästchen Allow key administrators to delete this key (Administratoren erlauben, diesen Schlüssel zu löschen).
14. Wählen Sie Weiter aus.
15. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel für kryptographische Operationen verwenden können

> **Hinweise**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter den Anweisungskennungen `"Allow use of the key"` und `"Allow attachment of persistent`

`resources"` wichtige Benutzer hinzu. Das Ändern dieser Anweisungskennungen kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Aussage vornehmen.

16. (Optional) Sie können anderen erlauben, diesen KMS-Schlüssel für kryptografische Operationen AWS-Konten zu verwenden. Wählen Sie dazu unten auf der Seite im AWS-Konten Abschnitt Andere die Option Weiteres hinzufügen aus AWS-Konto und geben Sie die AWS-Konto Identifikationsnummer eines externen Kontos ein. Wiederholen Sie diesen Schritt, um weitere externe Konten hinzufügen.

> **Note**
>
> Um auch Prinzipalen aus den externen Konten Zugriff auf den KMS-Schlüssel zu erlauben, müssen die Administratoren der externen Konten IAM-Richtlinien erstellen, die diese Berechtigungen bereitstellen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

17. Wählen Sie Weiter aus.
18. Lesen Sie die wichtigsten Grundsatzerklärungen für den Schlüssel. Um Änderungen an der wichtigsten Richtlinie vorzunehmen, wählen Sie Bearbeiten aus.
19. Wählen Sie Weiter aus.
20. Überprüfen Sie die gewählten Einstellungen. Sie können immer noch zurückgehen und alle Einstellungen ändern.
21. Wählen Sie Finish (fertigstellen) aus, um den KMS-Schlüssel zu erstellen.

Verwenden der AWS KMS API

Sie können den CreateKeyVorgang verwenden, um alle AWS KMS keys Typen zu erstellen. In diesen Beispielen wird das AWS Command Line Interface (AWS CLI) verwendet. Beispiele in verschiedenen Programmiersprachen finden Sie unter Verwendung `CreateKey` mit einem AWS SDK oder CLI.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in die Felder `Description` oder `Tags` ein. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben im Klartext vorkommen.

Der folgende Vorgang erstellt einen symmetrischen Verschlüsselungsschlüssel in einer einzelnen Region, der durch Schlüsselmaterial unterstützt wird, das von AWS KMS generiert wurde. Diese Produktion hat keine erforderlichen Parameter. Gegebenenfalls können Sie auch mit dem Parameter `Policy` eine Schlüsselrichtlinie angeben. Sie können die Schlüsselrichtlinie (PutKeyPolicy) jederzeit ändern und optionale Elemente wie eine Beschreibung und Tags hinzufügen. Darüber hinaus können Sie Asymmetrische Schlüssel, Schlüssel zu mehreren Regionen, Schlüssel mit Importiertes Schlüsselmaterial und Schlüssel in Custom Key Store erstellen. Verwenden Sie den Vorgang, um Datenschlüssel für die clientseitige Verschlüsselung zu erstellen. GenerateDataKey

Bei `CreateKey` diesem Vorgang können Sie keinen Alias angeben, aber Sie können den CreateAliasVorgang verwenden, um einen Alias für Ihren neuen KMS-Schlüssel zu erstellen.

Es folgt ein Beispiel für einen Aufruf der Produktion `CreateKey` ohne Parameter. Dieser Befehl verwendet alle Standardwerte. Es erstellt einen KMS-Schlüssel mit symmetrischer Verschlüsselung zum Verschlüsseln und Entschlüsseln mit von AWS KMS generiertem Schlüsselmaterial.

```
$ aws kms create-key
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1502910355.475,
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "MultiRegion": false
        "EncryptionAlgorithms": [
```

```
            "SYMMETRIC_DEFAULT"
        ],
    }
}
```

Wenn Sie keine Schlüsselrichtlinie für Ihren neuen KMS-Schlüssel angeben, unterscheidet sich die Standard-Schlüsselrichtlinie, die `CreateKey` verwendet, von der Standard-Schlüsselrichtlinie, die die Konsole anwendet, wenn Sie einen neuen KMS-Schlüssel erstellen.

Bei diesem Aufruf des GetKeyPolicyVorgangs wird beispielsweise die geltende Schlüsselrichtlinie zurückgegeben. `CreateKey` Sie gewährt AWS-Konto Zugriff auf den KMS-Schlüssel und ermöglicht das Erstellen von AWS Identity and Access Management (IAM-) Richtlinien für den KMS-Schlüssel. Detaillierte Informationen zu IAM-Richtlinien und Schlüsselrichtlinien für KMS-Schlüssel finden Sie unter KMS-Schlüsselzugriff und Berechtigungen

```
$ aws kms get-key-policy --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --policy-name
 default --output text
```

JSON

```
{
  "Version" : "2012-10-17",
  "Id" : "key-default-1",
  "Statement" : [ {
    "Sid" : "EnableIAMUserPermissions",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : "kms:*",
    "Resource" : "*"
  } ]
}
```

Erstellen eines asymmetrischen KMS-Schlüssels

Sie können asymmetrische KMS-Schlüssel in der AWS KMS Konsole, mithilfe der CreateKeyAPI oder mithilfe der AWS::KMS::Key AWS CloudFormation Vorlage erstellen. Ein asymmetrischer KMS-Schlüssel stellt ein öffentliches und privates key pair dar, das zum Verschlüsseln, Signieren oder

Ableiten gemeinsamer Geheimnisse verwendet werden kann. Der private Schlüssel verbleibt drinnen. AWS KMS Informationen zum Herunterladen des öffentlichen Schlüssels zur Verwendung außerhalb von AWS KMS finden Sie unterLaden Sie den öffentlichen Schlüssel herunter.

Wenn Sie einen asymmetrischen KMS-Schlüssel erstellen, müssen Sie eine Schlüsselspezifikation auswählen. Häufig wird die von Ihnen ausgewählte Schlüsselspezifikation durch gesetzliche, Sicherheits- oder geschäftliche Anforderungen bestimmt. Sie kann auch von der Größe der Nachrichten beeinflusst werden, die Sie verschlüsseln oder signieren müssen. Im Allgemeinen sind längere Verschlüsselungsschlüssel Brute-Force-Angriffen gegenüber weniger anfällig. Eine ausführliche Beschreibung aller unterstützten Schlüsselspezifikationen finden Sie unter. Referenz zu wichtigen Spezifikationen

AWS Dienste, die in integriert sind, unterstützen AWS KMS keine asymmetrischen KMS-Schlüssel. Wenn Sie einen KMS-Schlüssel erstellen möchten, der Daten verschlüsselt, die Sie in einem AWS Dienst speichern oder verwalten, erstellen Sie einen KMS-Schlüssel mit symmetrischer Verschlüsselung.

Weitere Informationen über die Berechtigungen, die zum Erstellen von KMS-Schlüsseln erforderlich sind, finden Sie unter Berechtigungen zum Erstellen von KMS-Schlüsseln.

Verwenden der Konsole AWS KMS

Sie können das verwenden AWS Management Console , um asymmetrische AWS KMS keys (KMS-Schlüssel) zu erstellen. Jeder asymmetrische KMS-Schlüssel repräsentiert ein Schlüsselpaar aus einem öffentlichen und einem privaten Schlüssel.

> **Important**
>
> Nehmen Sie keine vertraulichen oder sensiblen Informationen in den Alias, in der Beschreibung oder in den Tags auf. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Klicken Sie auf Create key.

5. Um einen asymmetrischen KMS-Schlüssel zu erstellen, wählen Sie unter Key type (Schlüsseltyp) die Option Asymmetric (Asymmetrisch) aus.
6. Um einen asymmetrischen KMS-Schlüssel für die Verschlüsselung öffentlicher Schlüssel zu erstellen, wählen Sie unter Key usage (Schlüsselnutzung) Encrypt and decrypt (Verschlüsseln und Entschlüsseln) aus.

   Um einen asymmetrischen KMS-Schlüssel zum Signieren von Nachrichten und Überprüfen von Signaturen zu erstellen, wählen Sie unter Schlüsselverwendung die Option Signieren und überprüfen aus.

   Um einen asymmetrischen KMS-Schlüssel zum Ableiten gemeinsamer geheimer Schlüssel zu erstellen, wählen Sie unter Schlüsselverwendung die Option Schlüsselvereinbarung aus.

   Hilfe bei der Auswahl eines Schlüsselnutzungswerts finden Sie unter Wählen Sie aus, welche Art von KMS-Schlüssel erstellt werden soll.
7. Wählen Sie eine Spezifikation (Key spec (Schlüsselspezifikation)) für Ihren asymmetrischen KMS-Schlüssel aus.
8. Wählen Sie Weiter aus.
9. Geben Sie einen Alias für den KMS-Schlüssel ein. Der Aliasname darf nicht mit **`aws/`** beginnen. Das Präfix **`aws/`** ist von Amazon Web Services reserviert und steht für Von AWS verwaltete Schlüssel in Ihrem Konto.

   Ein Alias ist ein benutzerfreundlicher Name, mit dem Sie den KMS-Schlüssel in der Konsole und in einigen Konsolen identifizieren können. AWS KMS APIs Wir empfehlen, dass Sie einen Alias wählen, der auf die Art von Daten, die Sie schützen möchten, oder die Anwendung, die Sie mit dem KMS-Schlüssel verwenden möchten, hindeutet.

   Zum Erstellen eines KMS-Schlüssels in der Konsole benötigen Sie Aliase AWS Management Console. Sie können keinen Alias angeben, wenn Sie den CreateKeyVorgang verwenden, aber Sie können die Konsole oder den CreateAliasVorgang verwenden, um einen Alias für einen vorhandenen KMS-Schlüssel zu erstellen. Details hierzu finden Sie unter Aliase in AWS KMS.
10. (Optional) Geben Sie eine Beschreibung für den KMS-Schlüssel ein.

    Geben Sie eine Beschreibung ein, die die Art von Daten, die Sie schützen möchten, oder die Anwendung, die Sie mit dem KMS-Schlüssel verwenden möchten, erklärt.

    Sie können jetzt eine Beschreibung hinzufügen oder sie jederzeit aktualisieren, es sei denn, der Schlüsselstatus lautet `Pending Deletion` oder `Pending Replica Deletion`. Um die

Beschreibung eines vorhandenen, vom Kunden verwalteten Schlüssels hinzuzufügen, zu ändern oder zu löschen, bearbeiten Sie die Beschreibung auf der Detailseite für den KMS-Schlüssel im AWS Management Console oder verwenden Sie den UpdateKeyDescriptionVorgang.

11. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Add tag (Tag hinzufügen), wenn Sie mehr als ein Tag zum KMS-Schlüssel hinzufügen möchten.

    Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenverteilungsbericht generiert, in dem die Nutzung und die Kosten nach Stichwörtern zusammengefasst sind. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Schlagworte in AWS KMS und ABAC für AWS KMS.

12. Wählen Sie Weiter aus.

13. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel verwalten können.

    > **Hinweise**
    >
    > Diese Schlüsselrichtlinie ermöglicht die AWS-Konto vollständige Kontrolle über diesen KMS-Schlüssel. Kontoadministratoren können damit anderen Prinzipalen mithilfe von IAM-Richtlinien die Berechtigung zum Verwalten des KMS-Schlüssels erteilen. Details hierzu finden Sie unter the section called “Standardschlüsselrichtlinie”.
    > Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
    > Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter der Anweisungs-ID wichtige Administratoren hinzu`"Allow access for Key Administrators"`. Eine Änderung dieser Anweisungs-ID kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Anweisung vornehmen.

14. (Optional) Um zu verhindern, dass die ausgewählten IAM-Benutzer und -Rollen diesen KMS-Schlüssel löschen, deaktivieren Sie unten auf der Seite im Abschnitt Key deletion (Schlüssellöschung) das Kontrollkästchen Allow key administrators to delete this key (Administratoren erlauben, diesen Schlüssel zu löschen).

15. Wählen Sie Weiter aus.

16. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel für kryptographische Operationen verwenden können.

> **Hinweise**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
>
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter den Anweisungskennungen `"Allow use of the key"` und `"Allow attachment of persistent resources"` wichtige Benutzer hinzu. Das Ändern dieser Anweisungskennungen kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Aussage vornehmen.

17. (Optional) Sie können anderen erlauben, diesen KMS-Schlüssel für kryptografische Operationen AWS-Konten zu verwenden. Wählen Sie dazu im Abschnitt Other AWS-Konten(Andere Konten) unten auf der Seite die Option Add another AWS-Konto(Weiteres Konto hinzufügen) und geben Sie die AWS-Konto -ID eines externen Kontos ein. Wiederholen Sie diesen Schritt, um weitere externe Konten hinzufügen.

> **Note**
>
> Um auch Prinzipalen aus den externen Konten Zugriff auf den KMS-Schlüssel zu erlauben, müssen die Administratoren der externen Konten IAM-Richtlinien erstellen, die diese Berechtigungen bereitstellen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

18. Wählen Sie Weiter aus.
19. Lesen Sie die wichtigsten Richtlinienerklärungen für den Schlüssel. Um Änderungen an der wichtigsten Richtlinie vorzunehmen, wählen Sie Bearbeiten aus.
20. Wählen Sie Weiter aus.
21. Überprüfen Sie die gewählten Einstellungen. Sie können immer noch zurückgehen und alle Einstellungen ändern.
22. Wählen Sie Finish (fertigstellen) aus, um den KMS-Schlüssel zu erstellen.

Verwenden der AWS KMS API

Sie können die CreateKeyOperation verwenden, um eine Asymmetrie AWS KMS key zu erstellen. Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Wenn Sie einen asymmetrischen KMS-Schlüssel erstellen, müssen Sie den `KeySpec`-Parameter angeben, der den Typ der von Ihnen erstellten Schlüssel bestimmt. Außerdem müssen Sie den `KeyUsage` Wert ENCRYPT_DECRYPT, SIGN_VERIFY oder KEY_AGREEMENT angeben. Diese Eigenschaften können nicht geändert werden, nachdem der KMS-Schlüssel erstellt wurde.

Bei diesem `CreateKey` Vorgang können Sie keinen Alias angeben, aber Sie können den CreateAliasVorgang verwenden, um einen Alias für Ihren neuen KMS-Schlüssel zu erstellen.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in die Felder `Description` oder `Tags` ein. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

Erstellen Sie ein asymmetrisches KMS-Schlüsselpaar für die öffentliche Verschlüsselung

Im folgenden Beispiel wird die `CreateKey`-Produktion verwendet, um einen asymmetrischen KMS-Schlüssel von 4096-Bit-RSA-Schlüsseln für die Verschlüsselung öffentlicher Schlüssel zu erstellen.

```
$ aws kms create-key --key-spec RSA_4096 --key-usage ENCRYPT_DECRYPT
{
    "KeyMetadata": {
        "KeyState": "Enabled",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "Description": "",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": 1569973196.214,
        "MultiRegion": false,
        "KeySpec": "RSA_4096",
        "CustomerMasterKeySpec": "RSA_4096",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "EncryptionAlgorithms": [
```

```
            "RSAES_OAEP_SHA_1",
            "RSAES_OAEP_SHA_256"
        ],
        "AWSAccountId": "111122223333",
        "Origin": "AWS_KMS",
        "Enabled": true
    }
}
```

Erstellen Sie ein asymmetrisches KMS-Schlüsselpaar zum Signieren und Überprüfen

Der folgende Beispielbefehl erstellt einen asymmetrischen KMS-Schlüssel, der ein Paar von ECC-Schlüsseln darstellt, die zum Signieren und Überprüfen verwendet werden. Sie können kein Elliptic Curve-Schlüsselpaar für die Verschlüsselung und Entschlüsselung erstellen.

```
$ aws kms create-key --key-spec ECC_NIST_P521 --key-usage SIGN_VERIFY
{
    "KeyMetadata": {
        "KeyState": "Enabled",
        "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "CreationDate": 1570824817.837,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ECDSA_SHA_512"
        ],
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "AWSAccountId": "111122223333",
        "KeySpec": "ECC_NIST_P521",
        "CustomerMasterKeySpec": "ECC_NIST_P521",
        "KeyManager": "CUSTOMER",
        "Description": "",
        "Enabled": true,
        "MultiRegion": false,
        "KeyUsage": "SIGN_VERIFY"
    }
}
```

Erstellen Sie ein asymmetrisches KMS-Schlüsselpaar zum Ableiten gemeinsamer Geheimnisse

Der folgende Beispielbefehl erstellt einen asymmetrischen KMS-Schlüssel, der ein Paar von ECDH-Schlüsseln darstellt, die zum Ableiten von gemeinsamen Geheimnissen verwendet werden. Sie können kein Elliptic Curve-Schlüsselpaar für die Verschlüsselung und Entschlüsselung erstellen.

```
$ aws kms create-key --key-spec ECC_NIST_P256 --key-usage KEY_AGREEMENT
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "CreationDate": "2023-12-27T19:10:15.063000+00:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "KEY_AGREEMENT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "ECC_NIST_P256",
        "KeySpec": "ECC_NIST_P256",
        "KeyAgreementAlgorithms": [
            "ECDH"
        ],
        "MultiRegion": false
    }
}
```

Erstellen Sie einen HMAC-KMS-Schlüssel

Sie können HMAC-KMS-Schlüssel in der AWS KMS Konsole, mithilfe der CreateKeyAPI oder mithilfe der AWS::KMS::Key AWS CloudFormation Vorlage erstellen.

Wenn Sie einen HMAC-KMS-Schlüssel erstellen, müssen Sie eine Schlüsselspezifikation auswählen. AWS KMS unterstützt mehrere Schlüsselspezifikationen für HMAC-KMS-Schlüssel. Die von Ihnen ausgewählte Schlüsselspezifikation wird möglicherweise durch gesetzliche, Sicherheits- oder geschäftliche Anforderungen bestimmt. Im Allgemeinen sind längere Schlüssel Brute-Force-Angriffen gegenüber weniger anfällig.

Weitere Informationen über die Berechtigungen, die zum Erstellen von KMS-Schlüsseln erforderlich sind, finden Sie unter Berechtigungen zum Erstellen von KMS-Schlüsseln.

Verwenden der Konsole AWS KMS

Sie können den verwenden AWS Management Console , um HMAC-KMS-Schlüssel zu erstellen. HMAC-KMS-Schlüssel sind symmetrische Schlüssel mit einer Schlüsselverwendung von Generate

and verify MAC (MAC generieren und überprüfen). Sie können auch multiregionale HMAC-KMS-Schlüssel erstellen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Klicken Sie auf Create key.
5. Wählen Sie für Key type (Schlüsseltyp) Symmetric (Symmetrisch).

   HMAC-KMS-Schlüssel sind symmetrisch. Sie verwenden denselben Schlüssel, um HMAC-Tags zu generieren und zu überprüfen.
6. Wählen Sie für Schlüsselverwendung Generate and verify MAC (MAC Generieren und überprüfen) aus.

   MAC generieren und überprüfen ist die einzig gültige Schlüsselverwendung für HMAC-KMS-Schlüssel.

   > **Note**
   >
   > Key usage (Schlüsselverwendung) wird für symmetrische Schlüssel nur angezeigt, wenn HMAC-KMS-Schlüssel in Ihrer ausgewählten Region unterstützt werden.

7. Wählen Sie eine Spezifikation (Key spec (Schlüsselspezifikation)) für Ihren HMAC-KMS-Schlüssel aus.

   Die von Ihnen ausgewählte Schlüsselspezifikation kann möglicherweise durch gesetzliche, Sicherheits- oder geschäftliche Anforderungen bestimmt werden. Im Allgemeinen sind längere Schlüssel sicherer.
8. Um einen multiregionalen primären HMAC-Schlüssel zu erstellen, wählen Sie unter Advanced Options (Advanced Optionen) Multi-Region key (Multiregionaler Schlüssel) aus. Die freigegebenen Eigenschaften, die Sie für diesen KMS-Schlüssel definieren, wie z. B. den Schlüsseltyp und die Schlüsselverwendung, wird mit seinen Replikatschlüsseln geteilt.

Sie können dieses Verfahren nicht verwenden, um einen Replikatschlüssel zu erstellen. Um einen multiregionalen Replikat-HMAC-Schlüssel, befolgen Sie die Anweisungen zum Erstellen eines Replikatschlüssels.

9. Wählen Sie Weiter aus.
10. Geben Sie einen Alias für den KMS-Schlüssel ein. Der Aliasname darf nicht mit **`aws/`** beginnen. Das Präfix **`aws/`** ist von Amazon Web Services reserviert und steht für Von AWS verwaltete Schlüssel in Ihrem Konto.

    Wir empfehlen Ihnen, einen Alias zu verwenden, der den KMS-Schlüssel als HMAC-Schlüssel identifiziert, z. B. `HMAC/test-key`. Auf diese Weise können Sie Ihre HMAC-Schlüssel in der AWS KMS Konsole leichter identifizieren, wo Sie Schlüssel nach Tags und Aliasnamen sortieren und filtern können, aber nicht nach Schlüsselspezifikation oder Schlüsselverwendung.

    Zum Erstellen eines KMS-Schlüssels in der Konsole benötigen Sie Aliase AWS Management Console. Sie können keinen Alias angeben, wenn Sie den CreateKeyVorgang verwenden, aber Sie können die Konsole oder den CreateAliasVorgang verwenden, um einen Alias für einen vorhandenen KMS-Schlüssel zu erstellen. Details hierzu finden Sie unter Aliase in AWS KMS.
11. (Optional) Geben Sie eine Beschreibung für den KMS-Schlüssel ein.

    Geben Sie eine Beschreibung ein, die die Art von Daten, die Sie schützen möchten, oder die Anwendung, die Sie mit dem KMS-Schlüssel verwenden möchten, erklärt.

    Sie können jetzt eine Beschreibung hinzufügen oder sie jederzeit aktualisieren, es sei denn, der Schlüsselstatus lautet `Pending Deletion` oder `Pending Replica Deletion`. Um die Beschreibung eines vorhandenen, vom Kunden verwalteten Schlüssels hinzuzufügen, zu ändern oder zu löschen, bearbeiten Sie die Beschreibung auf der Detailseite für den KMS-Schlüssel AWS Management Console im AWS Management Console oder verwenden Sie den UpdateKeyDescriptionVorgang.
12. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Add tag (Tag hinzufügen), wenn Sie mehr als ein Tag zum KMS-Schlüssel hinzufügen möchten.

    Erwägen Sie, ein Tag hinzuzufügen, das den Schlüssel als HMAC-Schlüssel identifiziert, z. B. `Type=HMAC`. Auf diese Weise können Sie Ihre HMAC-Schlüssel in der AWS KMS Konsole leichter identifizieren, wo Sie Schlüssel nach Tags und Aliasnamen sortieren und filtern können, aber nicht nach Schlüsselspezifikation oder Schlüsselverwendung.

Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, wird ein Kostenverteilungsbericht AWS generiert, in dem die Nutzung und die Kosten nach Stichwörtern zusammengefasst sind. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Schlagworte in AWS KMS und ABAC für AWS KMS.

13. Wählen Sie Weiter aus.
14. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel verwalten können.

> **Hinweise**
>
> Diese Schlüsselrichtlinie ermöglicht die AWS-Konto vollständige Kontrolle über diesen KMS-Schlüssel. Kontoadministratoren können damit anderen Prinzipalen mithilfe von IAM-Richtlinien die Berechtigung zum Verwalten des KMS-Schlüssels erteilen. Details hierzu finden Sie unter the section called “Standardschlüsselrichtlinie”.
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter der Anweisungs-ID wichtige Administratoren hinzu`"Allow access for Key Administrators"`. Eine Änderung dieser Anweisungs-ID kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Anweisung vornehmen.

15. (Optional) Um zu verhindern, dass die ausgewählten IAM-Benutzer und -Rollen diesen KMS-Schlüssel löschen, deaktivieren Sie unten auf der Seite im Abschnitt Key deletion (Schlüssellöschung) das Kontrollkästchen Allow key administrators to delete this key (Administratoren erlauben, diesen Schlüssel zu löschen).
16. Wählen Sie Weiter aus.
17. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel für kryptographische Operationen verwenden können.

> **Hinweise**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre

Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter den Anweisungskennungen `"Allow use of the key"` und `"Allow attachment of persistent resources"` wichtige Benutzer hinzu. Das Ändern dieser Anweisungskennungen kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Aussage vornehmen.

18. (Optional) Sie können anderen erlauben, diesen KMS-Schlüssel für kryptografische Operationen AWS-Konten zu verwenden. Wählen Sie dazu im Abschnitt Other AWS-Konten(Andere Konten) unten auf der Seite die Option Add another AWS-Konto(Weiteres Konto hinzufügen) und geben Sie die AWS-Konto -ID eines externen Kontos ein. Wiederholen Sie diesen Schritt, um weitere externe Konten hinzufügen.

> **Note**
>
> Um auch Prinzipalen aus den externen Konten Zugriff auf den KMS-Schlüssel zu erlauben, müssen die Administratoren der externen Konten IAM-Richtlinien erstellen, die diese Berechtigungen bereitstellen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

19. Wählen Sie Weiter aus.
20. Lesen Sie die wichtigsten Richtlinienerklärungen für den Schlüssel. Um Änderungen an der wichtigsten Richtlinie vorzunehmen, wählen Sie Bearbeiten aus.
21. Wählen Sie Weiter aus.
22. Überprüfen Sie die gewählten Einstellungen. Sie können immer noch zurückgehen und alle Einstellungen ändern.
23. Wählen Sie Finish (fertigstellen) aus, um den HMAC-KMS-Schlüssel zu erstellen.

Verwenden der AWS KMS API

Sie können den CreateKeyVorgang verwenden, um einen HMAC-KMS-Schlüssel zu erstellen. Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Wenn Sie einen HMAC-KMS-Schlüssel erstellen, müssen Sie den `KeySpec`-Parameter angeben, der den Typ der von Ihnen erstellten KMS-Schlüssel bestimmt. Außerdem müssen Sie einen

`KeyUsage`-Wert von GENERATE_VERIFY_MAC angeben, obwohl es der einzig gültige Wert für die Schlüsselverwendung für HMAC-Schlüssel ist. Um einen multiregionalen HMAC-KMS-Schlüssel zu erstellen, fügen Sie den `MultiRegion`-Parameter mit einem Wert von `true` hinzu. Diese Eigenschaften können nicht geändert werden, nachdem der KMS-Schlüssel erstellt wurde.

Bei `CreateKey` diesem Vorgang können Sie keinen Alias angeben, aber Sie können den CreateAliasVorgang verwenden, um einen Alias für Ihren neuen KMS-Schlüssel zu erstellen. Wir empfehlen Ihnen, einen Alias zu verwenden, der den KMS-Schlüssel als HMAC-Schlüssel identifiziert, z. B. `HMAC/test-key`. Auf diese Weise können Sie Ihre HMAC-Schlüssel in der AWS KMS Konsole leichter identifizieren, wo Sie Schlüssel nach Alias sortieren und filtern können, aber nicht nach Schlüsselspezifikation oder Schlüsselverwendung.

Wenn Sie versuchen, einen HMAC-KMS-Schlüssel in einem zu erstellen, AWS-Region in dem HMAC-Schlüssel nicht unterstützt werden, gibt der Vorgang Folgendes zurück `CreateKey` `UnsupportedOperationException`

Im folgenden Beispiel wird die `CreateKey`-Operation zum Erstellen eines 512-Bit-HMAC-KMS-Schlüssels verwendet.

```
$ aws kms create-key --key-spec HMAC_512 --key-usage GENERATE_VERIFY_MAC
{
    "KeyMetadata": {
        "KeyState": "Enabled",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "Description": "",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": 1669973196.214,
        "MultiRegion": false,
        "KeySpec": "HMAC_512",
        "CustomerMasterKeySpec": "HMAC_512",
        "KeyUsage": "GENERATE_VERIFY_MAC",
        "MacAlgorithms": [
            "HMAC_SHA_512"
        ],
        "AWSAccountId": "111122223333",
        "Origin": "AWS_KMS",
        "Enabled": true
    }
}
```

Primärschlüssel für mehrere Regionen erstellen

Sie können einen Primärschlüssel für mehrere Regionen in der AWS KMS Konsole oder mithilfe der AWS KMS API erstellen. Sie können den Primärschlüssel überall dort erstellen, AWS-Region wo Schlüssel für mehrere Regionen AWS KMS unterstützt werden.

Um einen Primärschlüssel mit mehreren Regionen zu erstellen, benötigt der Principal dieselben Berechtigungen wie für die Erstellung eines beliebigen KMS-Schlüssels, einschließlich der CreateKey kms-Berechtigung in einer IAM-Richtlinie. Der Principal benötigt außerdem die iam: -Berechtigung. CreateServiceLinkedRole Sie können den MultiRegionKeyType Bedingungsschlüssel kms: verwenden, um die Erlaubnis zum Erstellen von Primärschlüsseln für mehrere Regionen zuzulassen oder zu verweigern.

> **Note**
>
> Berücksichtigen Sie bei der Erstellung Ihres Primärschlüssels für mehrere Regionen sorgfältig die IAM-Benutzer und -Rollen, die Sie für die Verwaltung und Verwendung des Schlüssels auswählen. IAM-Richtlinien können anderen IAM-Benutzern und -Rollen die erforderlichen Berechtigungen zum Verwalten des KMS-Schlüssels erteilen.
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Konsole AWS KMS

Um einen Primärschlüssel für mehrere Regionen in der AWS KMS Konsole zu erstellen, verwenden Sie den gleichen Prozess, den Sie für die Erstellung eines beliebigen KMS-Schlüssels verwenden würden. Wählen Sie einen multiregionalen Schlüssel unter Erweiterte Optionen aus. Vollständige Anweisungen finden Sie unter Erstellen Sie einen KMS-Schlüssel.

> **Important**
>
> Nehmen Sie keine vertraulichen oder sensiblen Informationen in den Alias, in der Beschreibung oder in den Tags auf. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Klicken Sie auf Create key.
5. Wählen Sie einen symmetrischen oder asymmetrischen Schlüsseltyp. Symmetrische Schlüssel sind die Standardeinstellung.

   Sie können multiregionale symmetrische und asymmetrische Schlüssel, einschließlich multiregionaler HMAC-KMS-Schlüssel, die symmetrisch sind, erstellen.
6. Wählen Sie Ihre Schlüsselverwendung aus. Encrypt and decrypt (Verschlüsseln und Entschlüsseln) ist die Standardeinstellung.

   Weitere Informationen finden Sie unter Erstellen Sie einen KMS-Schlüssel, the section called “Erstellen eines asymmetrischen KMS-Schlüssels” oder the section called “Erstellen Sie einen HMAC-KMS-Schlüssel”.
7. Erweitern Sie Advanced options (Erweiterte Optionen).
8. Wählen Sie unter Herkunft des Schlüsselmaterials die Option KMS aus, um das Schlüsselmaterial AWS KMS generieren zu lassen, das Ihre Primär- und Replikatschlüssel gemeinsam nutzen. Wenn Sie Schlüsselmaterial in die Primär- oder die Replikatschlüssel importieren, wählen Sie External (Extern) aus.
9. Wählen Sie unter Regionalität die Option Multi-Region-Schlüssel aus.

   Sie können diese Einstellung nicht ändern, nachdem Sie den KMS-Schlüssel erstellt haben.
10. Geben Sie einen Alias für den Primärschlüssel ein.

    Aliasse sind keine gemeinsame Eigenschaft von multiregionalen Schlüsseln. Sie können Ihrem Primärschlüssel für mehrere Regionen und seinen Replikaten denselben Alias oder verschiedene Aliase zuweisen. AWS KMS synchronisiert die Aliase von Schlüsseln mit mehreren Regionen nicht.

    > **Note**
    >
    > Wenn Sie einen Alias hinzufügen, löschen oder aktualisieren, wird dadurch möglicherweise eine Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Details

dazu finden Sie unter ABAC für AWS KMS und Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern.

11. (Optional) Geben Sie eine Beschreibung für den Primärschlüssel ein.

    Beschreibungen sind keine gemeinsame Eigenschaft von multiregionalen Schlüsseln. Sie können Ihrem Primärschlüssel mit mehreren Regionen und seinen Replikaten dieselbe oder unterschiedliche Beschreibungen geben. AWS KMS synchronisiert die Schlüsselbeschreibungen von Schlüsseln mit mehreren Regionen nicht.

12. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wenn Sie dem Primärschlüssel mehrere Tags zuweisen möchten, wählen Sie Add tag (Tag hinzufügen) aus.

    Tags sind keine gemeinsame Eigenschaft von multiregionalen Schlüsseln. Sie können Ihrem multiregionalen Primärschlüssel und seinen Replikaten dieselben Tags oder verschiedene Tags zuweisen. AWS KMS synchronisiert die Tags von multiregionalen Schlüsseln nicht. Sie können die Tags auf KMS-Schlüssel jederzeit ändern.

    > **Note**
    >
    > Wenn Sie einen KMS-Schlüssel markieren oder entmarkieren, wird dadurch möglicherweise die Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Details dazu finden Sie unter ABAC für AWS KMS und Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern.

13. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel verwalten können.

    > **Hinweise**
    >
    > - Dieser Schritt startet den Prozess zum Erstellen einer Schlüsselrichtlinie für den Primärschlüssel. Schlüsselrichtlinien sind keine gemeinsame Eigenschaft von multiregionalen Schlüsseln. Sie können Ihrem Primärschlüssel für mehrere Regionen und seinen Replikaten dieselbe Schlüsselrichtlinie oder unterschiedliche Schlüsselrichtlinien zuweisen. AWS KMS synchronisiert nicht die wichtigsten Richtlinien von Schlüsseln für mehrere Regionen. Sie können die Schlüsselrichtlinie eines KMS-Schlüssels jederzeit ändern.
    > - Wenn Sie einen Primärschlüssel für mehrere Regionen erstellen, sollten Sie die von der Konsole generierte Standardschlüsselrichtlinie verwenden. Wenn Sie

diese Richtlinie ändern, bietet die Konsole keine Schritte zur Auswahl wichtiger Administratoren und Benutzer bei der Erstellung von Replikatschlüsseln und fügt auch keine entsprechenden Richtlinienerklärungen hinzu. Daher müssen Sie diese manuell hinzufügen.

- Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter der Statement-ID wichtige Administratoren hinzu`"Allow access for Key Administrators"`. Eine Änderung dieser Anweisungs-ID kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Anweisung vornehmen.

14. (Optional) Um zu verhindern, dass die ausgewählten IAM-Benutzer und -Rollen diesen KMS-Schlüssel löschen, deaktivieren Sie unten auf der Seite im Abschnitt Key deletion (Schlüssellöschung) das Kontrollkästchen Allow key administrators to delete this key (Administratoren erlauben, diesen Schlüssel zu löschen).
15. Wählen Sie Weiter aus.
16. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel für kryptographische Operationen verwenden können.

> **Hinweise**
>
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter den Anweisungskennungen `"Allow use of the key"` und `"Allow attachment of persistent resources"` wichtige Benutzer hinzu. Das Ändern dieser Anweisungskennungen kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Aussage vornehmen.

17. (Optional) Sie können anderen erlauben, diesen KMS-Schlüssel für kryptografische Operationen AWS-Konten zu verwenden. Wählen Sie dazu im Abschnitt Other AWS-Konten(Andere Konten) unten auf der Seite die Option Add another AWS-Konto(Weiteres Konto hinzufügen) und geben Sie die AWS-Konto -ID eines externen Kontos ein. Wiederholen Sie diesen Schritt, um weitere externe Konten hinzufügen.

> **Note**
>
> Um auch Prinzipalen aus den externen Konten Zugriff auf den KMS-Schlüssel zu erlauben, müssen die Administratoren der externen Konten IAM-Richtlinien erstellen, die

diese Berechtigungen bereitstellen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

18. Wählen Sie Weiter aus.
19. Lesen Sie die wichtigsten Richtlinienerklärungen für den Schlüssel. Um Änderungen an der wichtigsten Richtlinie vorzunehmen, wählen Sie Bearbeiten aus.
20. Wählen Sie Weiter aus.
21. Überprüfen Sie die gewählten Einstellungen. Sie können immer noch zurückgehen und alle Einstellungen ändern.
22. Wählen Sie Fertig stellen, um den Primärschlüssel für mehrere Regionen zu erstellen.

Verwenden der API AWS KMS

Verwenden Sie den CreateKeyVorgang, um einen Primärschlüssel für mehrere Regionen zu erstellen. Sie müssen außerdem den `MultiRegion`-Parameter mit dem Wert `True` verwenden.

Der folgende Befehl erstellt beispielsweise einen Primärschlüssel für mehrere Regionen im Aufrufer AWS-Region (us-east-1). Er akzeptiert Standardwerte für alle anderen Eigenschaften, einschließlich der Schlüsselrichtlinie. Die Standardwerte für multiregionale Primärschlüssel sind dieselben wie die Standardwerte für alle anderen KMS-Schlüssel, einschließlich der Standard-Schlüsselrichtlinie. Dieses Verfahren erstellt einen symmetrischen Verschlüsselungsschlüssel, den Standard-KMS-Schlüssel.

Die Antwort enthält das `MultiRegion`-Element und das `MultiRegionConfiguration`-Element mit typischen Unterelementen und Werten für einen multiregionalen Primärschlüssel ohne Replikatschlüssel. Die Schlüssel-ID eines multiregionalen Schlüssels beginnt immer mit `mrk-`.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in die Felder `Description` oder `Tags` ein. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

```
$ aws kms create-key --multi-region
{
    "KeyMetadata": {
```

```
        "Origin": "AWS_KMS",
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1606329032.475,
        "Arn": "arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "AWSAccountId": "111122223333",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": true,
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "us-east-1"
            },
            "ReplicaKeys": [ ]
      }
    }
}
```

Erstellen Sie Replikatschlüssel für mehrere Regionen

Sie können einen Replikatschlüssel für mehrere Regionen in der AWS KMS Konsole mithilfe des ReplicateKeyVorgangs oder mithilfe einer Vorlage erstellen. AWS::KMS::ReplicaKey AWS CloudFormation Sie können den CreateKeyVorgang nicht verwenden, um einen Replikatschlüssel zu erstellen.

Sie können diese Verfahren verwenden, um jeden multiregionalen Primärschlüssel zu replizieren, einschließlich ein KMS-Schlüssel mit symmetrischer Verschlüsselung, ein asymmetrischer KMS-Schlüssel oder einen HMAC-KMS-Schlüssel.

Wenn diese Produktion abgeschlossen ist, verfügt der neue Replikatschlüssel über einen vorübergehenden Schlüsselstatus von `Creating`. Dieser Schlüsselstatus ändert sich nach einigen

Sekunden, `PendingImport` wenn der Vorgang zum Erstellen des neuen Replikatschlüssels abgeschlossen ist, zu `Enabled` (oder wenn Sie einen Schlüssel mit mehreren Regionen mit importiertem Schlüsselmaterial erstellen). Während der Schlüsselstatus `Creating` ist, können Sie den Schlüssel verwalten, Sie können ihn jedoch noch nicht in kryptografischen Operationen verwenden. Wenn Sie den Replikatschlüssel programmgesteuert erstellen und verwenden, versuchen Sie es erneut `KMSInvalidStateException` oder rufen Sie ihn auf, um seinen Wert DescribeKeyzu überprüfen, bevor Sie ihn verwenden. `KeyState`

Wenn Sie versehentlich einen Replikatschlüssel löschen, können Sie ihn mit diesem Verfahren neu erstellen. Wenn Sie denselben Primärschlüssel in derselben Region replizieren, hat der neu erstellte Replikatschlüssel dieselben gemeinsamen Eigenschaften wie der ursprüngliche Replikatschlüssel.

> **Important**
>
> Nehmen Sie keine vertraulichen oder sensiblen Informationen in den Alias, in der Beschreibung oder in den Tags auf. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

Informationen zur Verwendung einer AWS CloudFormation Vorlage zur Erstellung eines Replikatschlüssels finden Sie AWS::KMS::ReplicaKeyim AWS CloudFormation Benutzerhandbuch.

Schritt 1: Wählen Sie Replikatregionen

In der Regel entscheiden Sie sich dafür, einen Schlüssel für mehrere Regionen in eine zu replizieren, die auf Ihrem Geschäftsmodell und Ihren regulatorischen Anforderungen AWS-Region basiert. Sie können beispielsweise einen Schlüssel in Regionen replizieren, in denen Sie Ihre Ressourcen behalten. Oder Sie könnten, um eine Anforderung zur Notfallwiederherstellung zu erfüllen, einen Schlüssel in geografisch entfernte Regionen replizieren.

Im Folgenden sind die AWS KMS Anforderungen für Replikatregionen aufgeführt. Wenn die ausgewählte Region diese Anforderungen nicht erfüllt, schlagen Versuche, einen Schlüssel zu replizieren, fehl.

- Ein verwandter multiregionaler Schlüssel pro Region – Sie können keinen Replikatschlüssel in derselben Region wie sein Primärschlüssel oder in derselben Region wie ein anderes Replikat des Primärschlüssels erstellen.

Wenn Sie versuchen, einen Primärschlüssel in einer Region zu replizieren, die bereits über ein Replikat dieses Primärschlüssels verfügt, schlägt der Versuch fehl. Wenn sich der aktuelle Replikationsschlüssel in der Region im `PendingDeletion`-Schlüsselstatus befindet, können Sie die Löschung des Replikationsschlüssels abbrechen oder warten, bis der Replikationsschlüssel gelöscht ist.

- Mehrere nicht verwandte multiregionale Schlüssel in derselben Region – Sie können mehrere nicht verwandte multiregionale Schlüssel in derselben Region haben. Beispielsweise können Sie zwei multiregionale Primärschlüssel in der Region `us-east-1` haben. Jeder Primärschlüssel kann einen Replikatschlüssel in der Region `us-west-2` haben.
- Regionen in derselben Partition – Die Region des Replikatschlüssels muss sich in derselben AWS -Partition befinden, wie die Region des Primärschlüssels.
- Region muss aktiviert sein – Wenn eine Region standardmäßig deaktiviert ist, können Sie in dieser Region keine Ressourcen erstellen, bis sie für Ihr AWS-Konto aktiviert ist.

Schritt 2: Replikatschlüssel erstellen

> **Note**
>
> Berücksichtigen Sie beim Erstellen von Replikatschlüsseln sorgfältig die IAM-Benutzer und -Rollen, die Sie für die Verwaltung und Verwendung des Replikatschlüssels auswählen. IAM-Richtlinien können anderen IAM-Benutzern und -Rollen die erforderlichen Berechtigungen zum Verwalten des KMS-Schlüssels erteilen.
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Konsole AWS KMS

In der AWS KMS Konsole können Sie im selben Vorgang ein oder mehrere Replikate eines Primärschlüssels für mehrere Regionen erstellen.

Dieses Verfahren ähnelt dem Erstellen eines einzelregionalen KMS-Schlüssels in der Konsole. Da jedoch ein Replikatschlüssel auf dem Primärschlüssel basiert, wählen Sie keine Werte für gemeinsam

genutzte Eigenschaften aus, z. B. die Schlüsselspezifikation (symmetrisch oder asymmetrisch), die Schlüsselnutzung oder den Schlüsselursprung.

Sie geben jedoch Eigenschaften an, die nicht freigegeben werden, einschließlich eines Alias, Tags, einer Beschreibung und einer Schlüsselrichtlinie. Die Konsole zeigt die aktuellen Eigenschaftswerte des Primärschlüssels an, Sie können sie jedoch ändern. Auch wenn Sie die Primärschlüsselwerte beibehalten, werden diese Werte AWS KMS nicht synchronisiert.

> **Important**
>
> Nehmen Sie keine vertraulichen oder sensiblen Informationen in den Alias, in der Beschreibung oder in den Tags auf. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Wählen Sie die Schlüssel-ID oder den Alias eines multiregionalen Primärschlüssels aus. Dadurch wird die Seite mit den Schlüsseldetails des KMS-Schlüssels geöffnet.

   Um einen multiregionalen Primärschlüssel zu identifizieren, verwenden Sie das Werkzeugsymbol in der oberen rechten Ecke, um die Spalte Regionality (Regionalität) zur Tabelle hinzuzufügen.
5. Wählen Sie die Registerkarte Regionality (Regionalität) aus.
6. Unter Related multi-Region keys (Verwandte multiregionale Schlüssel) wählen Sie Create new replica keys (neue Replikatschlüssel erstellen) aus.

   Der Abschnitt Related multi-Region keys (Verwandte multiregionale Schlüssel) zeigt die Region des Primärschlüssels und seiner Replikatschlüssel an. Sie können diese Anzeige verwenden, um die Region für den neuen Replikatschlüssel auszuwählen.
7. Wählen Sie mindestens eine AWS-Regionen aus. Bei diesem Verfahren wird in jedem der ausgewählten Regionen ein Replikatschlüssel erstellt.

   Das Menü enthält nur Regionen in derselben AWS Partition wie der Primärschlüssel. Regionen, die bereits über einen verwandten multiregionalen Schlüssel verfügen, werden angezeigt,

können jedoch nicht ausgewählt werden. Möglicherweise sind Sie nicht berechtigt, einen Schlüssel in alle Regionen im Menü zu replizieren.

Wenn Sie die Auswahl von Regionen abgeschlossen haben, schließen Sie das Menü. Die ausgewählten Regionen werden angezeigt. Um die Replikation in eine Region abzubrechen, wählen Sie das Kontrollkästchen X neben dem Namen der Region.

8. Geben Sie einen Alias für den Replikatschlüssel ein.

   Die Konsole zeigt einen der aktuellen Aliasen des Primärschlüssels an, Sie können ihn jedoch ändern. Sie können Ihrem multiregionalen Primärschlüssel und seinen Replikaten denselben Alias oder verschiedene Aliase zuweisen. Aliase sind keine gemeinsame Eigenschaft von Schlüsseln mit mehreren Regionen. AWS KMS synchronisiert die Aliase von Schlüsseln mit mehreren Regionen nicht.

   Wenn Sie einen Alias hinzufügen, löschen oder aktualisieren, wird dadurch möglicherweise eine Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Details dazu finden Sie unter ABAC für AWS KMS und Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern.

9. (Optional) Geben Sie eine Beschreibung für den Replikatschlüssel ein.

   Die Konsole zeigt die aktuelle Beschreibung des Primärschlüssels an, Sie können sie jedoch ändern. Beschreibungen sind keine gemeinsame Eigenschaft von multiregionalen Schlüsseln. Sie können Ihrem Primärschlüssel mit mehreren Regionen und seinen Replikaten dieselbe oder unterschiedliche Beschreibungen geben. AWS KMS synchronisiert die Schlüsselbeschreibungen von Schlüsseln mit mehreren Regionen nicht.

10. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wenn Sie mehrere Tags zum Replikatschlüssel zuweisen möchten, wählen Sie Add tag (Tag hinzufügen) aus.

    Die Konsole zeigt die Tags an, die aktuell mit dem Primärschlüssel verknüpft sind, Sie können sie jedoch ändern. Tags sind keine gemeinsame Eigenschaft von multiregionalen Schlüsseln. Sie können Ihrem Primärschlüssel für mehrere Regionen und seinen Replikaten dieselben oder unterschiedliche Tags zuweisen. AWS KMS synchronisiert die Tags von Schlüsseln mit mehreren Regionen nicht.

    Wenn Sie einen KMS-Schlüssel markieren oder entmarkieren, wird dadurch möglicherweise die Berechtigung für den KMS-Schlüssel erteilt oder verweigert. Details dazu finden Sie unter ABAC für AWS KMS und Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern.

11. Wählen Sie die IAM-Benutzer und -Rollen aus, die den Replikatschlüssel verwalten können.

Hinweise

- Wenn Sie bei der Erstellung Ihres Primärschlüssels für mehrere Regionen die Standardschlüsselrichtlinie geändert haben, fordert Sie die Konsole bei der Erstellung des Replikatschlüssels nicht zur Auswahl von Schlüsseladministratoren oder Schlüsselbenutzern (Schritte 11-15) auf. In diesem Fall müssen Sie der Schlüsselrichtlinie manuell die erforderlichen Berechtigungen für Schlüsseladministratoren und Benutzer hinzufügen, indem Sie im Schritt Schlüsselrichtlinie bearbeiten (Schritt 17) die Option Bearbeiten auswählen.
- Dieser Schritt startet den Prozess zum Erstellen einer Schlüsselrichtlinie für den Replikatschlüssel. Die Konsole zeigt die aktuelle Schlüsselrichtlinie des Primärschlüssels an, Sie können sie jedoch ändern. Schlüsselrichtlinien sind keine gemeinsame Eigenschaft von multiregionalen Schlüsseln. Sie können Ihrem multiregionalen Primärschlüssel und seinen Replikaten die gleiche Schlüsselrichtlinie oder andere Schlüsselrichtlinien zuweisen. Von AWS KMS werden Schlüsselrichtlinien nicht synchronisiert. Sie können die Schlüsselrichtlinie eines jeden KMS-Schlüssels jederzeit ändern.
- Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter der Statement-ID wichtige Administratoren hinzu`"Allow access for Key Administrators"`. Eine Änderung dieser Anweisungs-ID kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Anweisung vornehmen.

12. (Optional) Um zu verhindern, dass die ausgewählten IAM-Benutzer und -Rollen diesen KMS-Schlüssel löschen, deaktivieren Sie unten auf der Seite im Abschnitt Key deletion (Schlüssellöschung) das Kontrollkästchen Allow key administrators to delete this key (Administratoren erlauben, diesen Schlüssel zu löschen).
13. Wählen Sie Weiter aus.
14. Wählen Sie die IAM-Benutzer und -Rollen aus, die den KMS-Schlüssel für kryptographische Operationen verwenden können.

Hinweis

Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter den Anweisungskennungen `"Allow use of the key"` und `"Allow attachment of persistent resources"` wichtige Benutzer hinzu. Das Ändern dieser Anweisungskennungen kann

sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Aussage vornehmen.

15. (Optional) Sie können anderen erlauben, diesen KMS-Schlüssel für kryptografische Operationen AWS-Konten zu verwenden. Wählen Sie dazu im Abschnitt Other AWS-Konten(Andere Konten) unten auf der Seite die Option Add another AWS-Konto(Weiteres Konto hinzufügen) und geben Sie die AWS-Konto -ID eines externen Kontos ein. Wiederholen Sie diesen Schritt, um weitere externe Konten hinzufügen.

> **Note**
>
> Um auch Prinzipalen aus den externen Konten Zugriff auf den KMS-Schlüssel zu erlauben, müssen die Administratoren der externen Konten IAM-Richtlinien erstellen, die diese Berechtigungen bereitstellen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

16. Wählen Sie Weiter aus.
17. Lesen Sie die wichtigsten Richtlinienerklärungen für den Schlüssel. Um Änderungen an der wichtigsten Richtlinie vorzunehmen, wählen Sie Bearbeiten aus.
18. Wählen Sie Weiter aus.
19. Überprüfen Sie die gewählten Einstellungen. Sie können immer noch zurückgehen und alle Einstellungen ändern.
20. Wählen Sie Fertig stellen, um den Replikatschlüssel für mehrere Regionen zu erstellen.

Verwenden der API AWS KMS

Verwenden Sie den Vorgang, um einen Replikatschlüssel für mehrere Regionen zu erstellen. ReplicateKey Sie können den CreateKeyVorgang nicht verwenden, um einen Replikatschlüssel zu erstellen. Bei dieser Produktion wird jeweils ein Replikatschlüssel erstellt. Die Region, die Sie angeben, muss den Anforderungen der Region für Replikatschlüssel entsprechen.

Wenn Sie die `ReplicateKey`-Produktion verwenden, müssen Sie keine Werte für gemeinsam genutzte Eigenschaften von multiregionalen Schlüsseln angeben. Gemeinsam genutzte Eigenschaftswerte werden aus dem Primärschlüssel kopiert und synchronisiert. Sie können jedoch Werte für nicht freigegebene Eigenschaften angeben. Andernfalls werden die Standardstandardwerte für KMS-Schlüssel AWS KMS angewendet, nicht die Werte des Primärschlüssels.

> **Note**
>
> Wenn Sie keine Werte für die `Tags` Parameter`Description`, oder angeben`KeyPolicy`, AWS KMS wird der Replikatschlüssel mit einer leeren Zeichenkettenbeschreibung, der Standardschlüsselrichtlinie und ohne Tags erstellt.
> Geben Sie keine vertraulichen oder sensiblen Informationen in die Felder `Description` oder `Tags` ein. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

Beispielsweise erstellt der folgende Befehl einen multiregionalen Replikatschlüssel in der Region Asien-Pazifik (Sydney) (ap-southeast-2). Dieser Replikatschlüssel wird nach dem Primärschlüssel in der Region USA Ost (Nord-Virginia) (us-east-1) modelliert, der durch den Wert des `KeyId`-Parameters identifiziert wird. Dieses Beispiel akzeptiert Standardwerte für alle anderen Eigenschaften, einschließlich der Schlüsselrichtlinie.

Die Antwort beschreibt den neuen Replikatschlüssel. Sie enthält Felder für gemeinsam genutzte Eigenschaften, z. B. `KeyId`, `KeySpec` und `KeyUsage`, und den Ursprung des Schlüsselmaterials (`Origin`). Sie enthält auch Eigenschaften, die unabhängig vom Primärschlüssel sind, z. B. die `Description`, Schlüsselrichtlinie (`ReplicaKeyPolicy`) und Tags (`ReplicaTags`).

Die Antwort enthält auch den Schlüssel-ARN und die Region des Primärschlüssels und aller seiner Replikatschlüssel, einschließlich des Schlüssels, der gerade in der Region ap-southeast-2 erstellt wurde. In diesem Beispiel zeigt das `ReplicaKey`-Element an, dass dieser Primärschlüssel bereits in der Region Europa (Irland) (eu-west-1) repliziert wurde.

```
$ aws kms replicate-key \
    --key-id arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \
    --replica-region ap-southeast-2
{
    "ReplicaKeyMetadata": {
        "MultiRegion": true,
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "REPLICA",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "us-east-1"
            },
```

```
            "ReplicaKeys": [
                {
                    "Arn": "arn:aws:kms:ap-southeast-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "ap-southeast-2"
                },
                {
                    "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "eu-west-1"
                }
            ]
        },
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:ap-southeast-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "CreationDate": 1607472987.918,
        "Description": "",
        "Enabled": true,
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "Origin": "AWS_KMS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    },
    "ReplicaKeyPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Id\" : \"key-
default-1\",...,
    "ReplicaTags": []
}
```

Erstellen Sie einen KMS-Schlüssel mit importiertem Schlüsselmaterial

Mit importiertem Schlüsselmaterial können Sie Ihre AWS Ressourcen mit von Ihnen generierten kryptografischen Schlüsseln schützen. Die folgende Übersicht erläutert den Prozess zum Importieren

Ihres Schlüsselmaterials in AWS KMS. Weitere Informationen zu den einzelnen Schritten des Prozesses finden Sie in den entsprechenden Themen.

1. Erstellen eines KMS-Schlüssels ohne Schlüsselmaterial – Der Ursprung muss `EXTERNAL` sein. Ein Schlüsselursprung von `EXTERNAL` weist darauf hin, dass der Schlüssel für importiertes Schlüsselmaterial konzipiert ist und AWS KMS verhindert, dass Schlüsselmaterial für den KMS-Schlüssel generiert wird. In einem späteren Schritt importieren Sie Ihr eigenes Schlüsselmaterial in diesen KMS-Schlüssel.

   Das Schlüsselmaterial, das Sie importieren, muss mit der Schlüsselspezifikation des zugehörigen AWS KMS Schlüssels kompatibel sein. Weitere Informationen zur Kompatibilität finden Sie unterthe section called “Anforderungen an importiertes Schlüsselmaterial”.
2. Herunterladen des öffentlichen Verpackungsschlüssels und Import-Tokens – Nachdem Sie Schritt 1 abgeschlossen haben, laden Sie einen öffentlichen Verpackungsschlüssel und einen Import-Token herunter. Diese Artikel schützen Ihre wichtigsten Materialien, während sie importiert werden AWS KMS.

   In diesem Schritt wählen Sie den Typ („Schlüsselspezifikation“) des RSA-Verpackungsschlüssels und den Verpackungsalgorithmus, den Sie zur Verschlüsselung Ihrer Daten während der Übertragung an AWS KMS verwenden werden. Sie können jedes Mal, wenn Sie dasselbe Schlüsselmaterial importieren oder erneut importieren, eine andere Verpackungsschlüsselspezifikation und einen anderen Verpackungsschlüsselalgorithmus wählen.
3. Verschlüsseln des Schlüsselmaterials – Verwenden Sie den öffentlichen Verpackungsschlüssel, den Sie in Schritt 2 heruntergeladen haben, um das Schlüsselmaterial zu verschlüsseln, das Sie in Ihrem eigenen System erstellt haben.
4. Importieren von Schlüsselmaterial – Laden Sie das verschlüsselte Schlüsselmaterial, das Sie in Schritt 3 erstellt haben, und das in Schritt 2 heruntergeladene Import-Token hoch.

   In diesem Stadium können Sie optional eine Ablaufzeit festlegen. Wenn importiertes Schlüsselmaterial abläuft, wird es AWS KMS gelöscht und der KMS-Schlüssel wird unbrauchbar. Wenn Sie den KMS-Schlüssel erneut nutzen möchten, müssen Sie das gleiche Schlüsselmaterial erneut importieren.

   Wenn der Importvorgang erfolgreich abgeschlossen wurde, ändert sich der Schlüsselstatus des KMS-Schlüssels von `PendingImport` zu `Enabled`. Sie können Ihren KMS-Schlüssel jetzt in kryptografischen Operationen verwenden.

AWS KMS zeichnet einen Eintrag in Ihrem AWS CloudTrail Protokoll auf, wenn Sie den KMS-Schlüssel erstellen, den öffentlichen Schlüssel und das Importtoken herunterladen und das Schlüsselmaterial importieren. AWS KMS zeichnet auch einen Eintrag auf, wenn Sie importiertes Schlüsselmaterial löschen oder wenn abgelaufenes Schlüsselmaterial AWS KMS gelöscht wird.

Berechtigungen zum Importieren von Schlüsselmaterial

Um KMS-Schlüssel mit importiertem Schlüsselmaterial zu erstellen und zu verwalten, benötigt der Benutzer die Berechtigung für die Operationen in diesem Prozess. Sie können die `kms:GetParametersForImport`-, `kms:ImportKeyMaterial`-, und`kms:DeleteImportedKeyMaterial`-Berechtigungen in der Schlüsselrichtlinie beim Erstellen des KMS-Schlüssels bereitstellen. In der AWS KMS Konsole werden diese Berechtigungen automatisch für Schlüsseladministratoren hinzugefügt, wenn Sie einen Schlüssel mit einem externen Schlüsselmaterial-Ursprung erstellen.

Um KMS-Schlüssel mit importiertem Schlüsselmaterial zu erstellen, benötigt der Prinzipal die folgenden Berechtigungen.

- kms: CreateKey (IAM-Richtlinie)
  - Um diese Berechtigung auf KMS-Schlüssel mit importiertem Schlüsselmaterial zu beschränken, verwenden Sie die KeyOrigin Richtlinienbedingung kms: mit dem Wert. `EXTERNAL`

```
{
  "Sid": "CreateKMSKeysWithoutKeyMaterial",
  "Effect": "Allow",
  "Resource": "*",
  "Action": "kms:CreateKey",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "EXTERNAL"
    }
  }
}
```

- kms: GetParametersForImport (Schlüsselrichtlinie oder IAM-Richtlinie)
  - Um diese Berechtigung auf Anfragen zu beschränken, die einen bestimmten Wrapping-Algorithmus und eine bestimmte Wrapping-Schlüsselspezifikation verwenden, verwenden Sie die WrappingKeySpec Richtlinienbedingungen kms: WrappingAlgorithm und kms:.
- kms: ImportKeyMaterial (Schlüsselrichtlinie oder IAM-Richtlinie)

- Verwenden Sie die ValidTo Richtlinienbedingungen kms: und kms:, um Schlüsselmaterial, das abläuft, zuzulassen oder zu verbieten ExpirationModel und das Ablaufdatum zu kontrollieren.

Um importiertes Schlüsselmaterial erneut zu importieren, benötigt der Principal die Berechtigungen kms: GetParametersForImport und kms: ImportKeyMaterial.

Um importiertes Schlüsselmaterial zu löschen, benötigt der Principal die kms: DeleteImportedKeyMaterial -Berechtigung.

Um beispielsweise der Beispiel-`KMSAdminRole` die Berechtigung zu erteilen, alle Aspekte eines KMS-Schlüssels mit importiertem Schlüsselmaterial zu verwalten, fügen Sie eine Schlüsselrichtlinienanweisung wie die folgende in die Schlüsselrichtlinie des KMS-Schlüssels ein.

```
{
  "Sid": "Manage KMS keys with imported key material",
  "Effect": "Allow",
  "Resource": "*",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/KMSAdminRole"
  },
  "Action": [
    "kms:GetParametersForImport",
    "kms:ImportKeyMaterial",
    "kms:DeleteImportedKeyMaterial"
  ]
}
```

Anforderungen an importiertes Schlüsselmaterial

Das Schlüsselmaterial, das Sie importieren, muss mit der Schlüsselspezifikation des zugehörigen KMS-Schlüssels kompatibel sein. Importieren Sie bei asymmetrischen Schlüsselpaaren nur den privaten Schlüssel des Paares. AWS KMS leitet den öffentlichen Schlüssel vom privaten Schlüssel ab.

AWS KMS unterstützt die folgenden Schlüsselspezifikationen für KMS-Schlüssel mit importiertem Schlüsselmaterial.

- Schlüssel zur symmetrischen Verschlüsselung
  - Wichtige Spezifikationen:

    - SYMMETRIC_DEFAULT.
  - Voraussetzungen:
    - 256 Bit (32 Byte) an Binärdaten.
    - In chinesischen Regionen muss es sich um 128 Bit (16 Byte) an Binärdaten handeln.
- HMAC-Schlüssel
  - Die wichtigsten technischen Daten:
    - HMAC_224
    - HMAC_256
    - HMAC_384
    - HMAC_512
  - Voraussetzungen:
    - Das HMAC-Schlüsselmaterial muss RFC 2104 entsprechen.
    - Die Schlüssellänge muss mindestens der in der Schlüsselspezifikation angegebenen Länge entsprechen. Die maximale Schlüssellänge beträgt 1024 Bit.
    - Wenn Ihr Schlüsselmaterial 1024 Bit überschreitet, können Sie das Schlüsselmaterial hashen und die Hash-Ausgabe importieren. Der Hash-Algorithmus muss mit der Schlüsselspezifikation des HMAC-KMS-Schlüssels übereinstimmen, den Sie erstellen.
  - Beispiel:
    - Um 2048 Bit Schlüsselmaterial in einen HMAC_256-Schlüssel zu importieren, berechnen Sie zuerst den SHA-256-Hash des 2048-Bit-Schlüsselmaterials und importieren Sie dann die resultierende 256-Bit-Hashausgabe in den KMS-Schlüssel.
  - Gültige Schlüssellängen:
    - HMAC_224:224—1024 Bit
    - HMAC_256:256—1024 Bit
    - HMAC_384:384—1024 Bit
    - HMAC_512:512—1024 Bit
- Asymmetrischer privater RSA-Schlüssel
  - Die wichtigsten Spezifikationen:
    - RSA_2048
    - RSA_3072
    - RSA_4096

- Voraussetzungen:
  - Der asymmetrische private RSA-Schlüssel, den Sie importieren, muss Teil eines Schlüsselpaares sein, das RFC 3447 entspricht.
  - Modul: 2 048 Bit, 3 072 Bit oder 4 096 Bit
  - Anzahl der Primzahlen: 2 (RSA-Schlüssel mit mehreren Primzahlen werden nicht unterstützt)
  - Asymmetrisches Schlüsselmaterial muss im Format Public-Key Cryptography Standards (PKCS) #8, das RFC 5208 entspricht, BER- oder DER-codiert sein.
- Asymmetrischer privater Schlüssel mit elliptischer Kurve
  - Die wichtigsten Spezifikationen:
    - ECC_NIST_P256 (secp256r1)
    - ECC_NIST_P384 (secp384r1)
    - ECC_NIST_P521 (secp521r1)
    - ECC_SECG_P256K1 (secp256k1)
  - Voraussetzungen:
    - Der asymmetrische private ECC-Schlüssel, den Sie importieren, muss Teil eines Schlüsselpaares sein, das RFC 5915 entspricht.
    - Kurve: NIST P-256, NIST P-384, NIST P-521 oder secP256K1.
    - Parameter: Nur benannte Kurven (ECC-Schlüssel mit expliziten Parametern werden zurückgewiesen).
    - Öffentliche Punktkoordinaten: Können komprimiert, unkomprimiert oder projektiv sein.
    - Asymmetrisches Schlüsselmaterial muss im Format Public-Key Cryptography Standards (PKCS) #8, das RFC 5208 entspricht, BER- oder DER-codiert sein.
- ML-DSA-Schlüssel
  - Die wichtigsten Spezifikationen:
    - ML_DSA_44
    - ML_DSA_65
    - ML_DSA_87

> **Important**
>
> Das Importieren von ML-DSA-Schlüsseln wird nicht unterstützt.

- SM2 asymmetrischer privater Schlüssel (nur Regionen China)

- Voraussetzungen:
  - Der SM2 asymmetrische private Schlüssel, den Sie importieren, muss Teil eines key pair sein, das 0003 entspricht GM/T .
  - Kurve:. SM2
  - Parameter: Nur benannte Kurve (SM2 Schlüssel mit expliziten Parametern werden abgelehnt).
  - Öffentliche Punktkoordinaten: Können komprimiert, unkomprimiert oder projektiv sein.
  - Asymmetrisches Schlüsselmaterial muss im Format Public-Key Cryptography Standards (PKCS) #8, das RFC 5208 entspricht, BER- oder DER-codiert sein.

Schritt AWS KMS key 1: Erstellen Sie eine ohne Schlüsselmaterial

AWS KMS Erstellt standardmäßig Schlüsselmaterial für Sie, wenn Sie einen KMS-Schlüssel erstellen. Wenn Sie Ihr eigenes Schlüsselmaterial importieren möchten, können Sie beginnen, indem Sie zunächst einen KMS-Schlüssel ohne Schlüsselmaterial erstellen. Importieren Sie dann das Schlüsselmaterial. Um einen KMS-Schlüssel ohne Schlüsselmaterial zu erstellen, verwenden Sie die AWS KMS Konsole oder den CreateKeyVorgang.

Um einen Schlüssel ohne Schlüsselmaterial zu erstellen, geben Sie einen Ursprung von `EXTERNAL` an. Die Herkunftseigenschaft eines KMS-Schlüssels ist unveränderlich. Sobald Sie ihn erstellt haben, können Sie einen KMS-Schlüssel, der für importiertes Schlüsselmaterial entworfen wurde, nicht in einen KMS-Schlüssel mit Schlüsselmaterial aus AWS KMS oder einer anderen Quelle konvertieren.

Der Schlüsselstatus eines KMS-Schlüssels mit einem `EXTERNAL`-Ursprung und ohne Schlüsselmaterial ist `PendingImport`. Ein KMS-Schlüssel kann auf unbestimmte Zeit im Status `PendingImport`bleiben. Sie können jedoch einen KMS-Schlüssel im `PendingImport`-Status nicht für kryptografische Operationen verwenden. Wenn Sie Schlüsselmaterial importieren, ändert sich der Schlüsselstatus des KMS-Schlüssels zu `Enabled`, und Sie können ihn für kryptografische Operationen verwenden.

AWS KMS zeichnet ein Ereignis in Ihrem AWS CloudTrail Protokoll auf, wenn Sie den KMS-Schlüssel erstellen, den öffentlichen Schlüssel und das Importtoken herunterladen und das Schlüsselmaterial importieren. AWS KMS zeichnet auch ein CloudTrail Ereignis auf, wenn Sie importiertes Schlüsselmaterial löschen oder wenn abgelaufenes Schlüsselmaterial AWS KMS gelöscht wird.

Themen

- Erstellen eines KMS-Schlüssels ohne Schlüsselmaterial (Konsole)

- Einen KMS-Schlüssel ohne Schlüsselmaterial (API) erstellen AWS KMS

Erstellen eines KMS-Schlüssels ohne Schlüsselmaterial (Konsole)

Sie müssen nur einmal einen KMS-Schlüssel für das importierte Schlüsselmaterial anlegen. Sie können dasselbe Schlüsselmaterial so oft wie nötig in den bestehenden KMS-Schlüssel importieren und wieder importieren, aber Sie können kein anderes Schlüsselmaterial in einen KMS-Schlüssel importieren. Details hierzu finden Sie unter Schritt 2: Herunterladen des öffentlichen Verpackungsschlüssels und des Import-Tokens.

Um vorhandene KMS-Schlüssel mit importiertem Schlüsselmaterial in Ihrer Tabelle mit den vom Kunden verwalteten Schlüsseln zu finden, verwenden Sie das Zahnradsymbol in der oberen rechten Ecke, um die Spalte Origin (Herkunft) in der Liste der KMS-Schlüssel anzuzeigen. Importierte Schlüssel haben den Herkunft-Wert Extern (Schlüsselmaterial importieren) erkennbar.

Um einen KMS-Schlüssel mit importiertem Schlüsselmaterial zu erstellen, folgen Sie zunächst den Anweisungen zum Erstellen eines KMS-Schlüssels Ihres bevorzugten Schlüsseltyps, mit der folgenden Ausnahme.

Nachdem Sie die Schlüsselnutzung ausgewählt haben, gehen Sie wie folgt vor:

1. Erweitern Sie Advanced options (Erweiterte Optionen).
2. Stellen Sie die Key material origin (Herkunft des Schlüsselmaterials) auf External (Import key material) (Extern (Schlüsselmaterial importieren)) ein.
3. Aktivieren Sie dann das Kontrollkästchen neben Mir ist bekannt, dass die Verwendung eines importieren Schlüssel sich auf die Sicherheit und Lebensdauer auswirken kann um zu bestätigen, dass Sie die Auswirkungen der Verwendung von importiertem Schlüsselmaterial kennen. Weitere Informationen zu diesen Implikationen finden Sie unter Schützen von importiertem Schlüsselmaterial.
4. Optional: Um einen KMS-Schlüssel für mehrere Regionen mit importiertem Schlüsselmaterial zu erstellen, wählen Sie unter Regionalität die Option Multi-Region-Schlüssel aus.
5. Kehren Sie zu den grundlegenden Anweisungen zurück. Die verbleibenden Schritte des grundlegenden Verfahrens sind für alle KMS-Schlüssel dieses Typs identisch.

Wenn Sie Fertig stellen wählen, haben Sie einen KMS-Schlüssel ohne Schlüsselmaterial und mit dem Status (Schlüsselstatus) Ausstehender Import erstellt.

Anstatt jedoch zur Tabelle mit vom Kunden verwalteten Schlüsseln zurückzukehren, zeigt die Konsole eine Seite an, auf der Sie den öffentlichen Schlüssel herunterladen und das Token importieren können, das Sie für den Import Ihres Schlüsselmaterials benötigen. Sie können jetzt mit dem Download-Schritt fortfahren oder auf Abbrechen klicken, um den Vorgang an dieser Stelle zu beenden. Sie können jederzeit zu diesem Download-Schritt zurückkehren.

Danach: Schritt 2: Herunterladen des öffentlichen Verpackungsschlüssels und des Import-Tokens.

Einen KMS-Schlüssel ohne Schlüsselmaterial (API) erstellen AWS KMS

Um mithilfe der AWS KMS API einen KMS-Schlüssel mit symmetrischer Verschlüsselung ohne Schlüsselmaterial zu erstellen, senden Sie eine CreateKeyAnfrage, bei der der `Origin` Parameter auf `EXTERNAL` gesetzt ist. Im folgenden Beispiel wird gezeigt, wie Sie dies mit dem AWS Command Line Interface (AWS CLI) durchführen.

```
$ aws kms create-key --origin EXTERNAL
```

Wenn der Befehl erfolgreich ausgeführt wurde, wird eine Ausgabe ähnlich der Folgenden angezeigt. Der AWS KMS Schlüssel `Origin` ist `EXTERNAL` und `KeyState` ist`PendingImport`.

> **Tip**
>
> Wenn der Befehl nicht erfolgreich ist, wird möglicherweise eine `KMSInvalidStateException` oder eine `NotFoundException` angezeigt. Sie können die Anfrage wiederholen.

```
{
    "KeyMetadata": {
        "Origin": "EXTERNAL",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "Enabled": false,
        "MultiRegion": false,
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "PendingImport",
        "CreationDate": 1568289600.0,
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
```

```
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

Kopieren Sie den Wert `KeyId` für die spätere Verwendung aus der Befehlsausgabe und fahren Sie dann mit Schritt 2: Herunterladen des öffentlichen Verpackungsschlüssels und des Import-Tokens fort.

> **Note**
>
> Dieser Befehl erstellt einen KMS-Schlüssel für die symmetrische Verschlüsselung mit `KeySpec` von `SYMMETRIC_DEFAULT` und `KeyUsage` von `ENCRYPT_DECRYPT`. Sie können die optionalen Parameter `--key-spec` und `--key-usage` verwenden, um einen asymmetrischen oder HMAC-KMS-Schlüssel zu erstellen. Weitere Informationen finden Sie unter dem Vorgang CreateKey.

Schritt 2: Herunterladen des öffentlichen Verpackungsschlüssels und des Import-Tokens

Nachdem Sie ein AWS KMS key Material ohne Schlüssel erstellt haben, laden Sie mithilfe der AWS KMS Konsole oder der GetParametersForImportAPI einen öffentlichen Wrapping-Schlüssel und ein Import-Token für diesen KMS-Schlüssel herunter. Der öffentliche Verpackungsschlüssel und das Import-Token bilden einen untrennbaren Satz, der zusammen verwendet werden muss.

Sie werden den öffentlichen Verpackungsschlüssel verwenden, um Ihr Schlüsselmaterial für den Transport zu verschlüsseln. Bevor Sie ein RSA-Wrapping-Schlüsselpaar herunterladen, wählen Sie die Länge (Schlüsselspezifikation) des RSA-Wrapping-Schlüsselpaars und den Wrapping-Algorithmus aus, mit dem Sie Ihr importiertes Schlüsselmaterial für den Transport in Schritt 3 verschlüsseln werden. AWS KMS unterstützt auch die SM2 Wrapping-Schlüsselspezifikation (nur Regionen China).

Jeder öffentliche Verpackungsschlüssel und Import-Token-Satz ist 24 Stunden gültig. Wenn Sie sie nicht innerhalb von 24 Stunden nach dem Download verwenden, um Schlüsselmaterial zu

importieren, müssen Sie einen neuen Satz herunterladen. Sie können jederzeit neue öffentliche Verpackungsschlüssel herunterladen und Tokensätze importieren. Auf diese Weise können Sie die Länge Ihres RSA-Verpackungsschlüssels („Schlüsselspezifikation“) ändern oder einen verlorenen Satz ersetzen.

Sie können auch einen öffentlichen Verpackungsschlüssel und ein Import-Token-Set herunterladen, um dasselbe Schlüsselmaterial erneut in einen KMS-Schlüssel zu importieren. Sie können dies tun, um die Ablaufzeit für das Schlüsselmaterial festzulegen oder zu ändern, oder um abgelaufenes oder gelöschtes Schlüsselmaterial wiederherzustellen. Sie müssen Ihr Schlüsselmaterial bei jedem Import in herunterladen und erneut verschlüsseln. AWS KMS

Verwendung des öffentlichen Verpackungsschlüssels

Der Download beinhaltet einen öffentlichen Schlüssel, der nur für Sie bestimmt ist. AWS-Konto Er wird auch als öffentlicher Schlüssel bezeichnet.

Bevor Sie Schlüsselmaterial importieren, verschlüsseln Sie das Schlüsselmaterial mit dem öffentlichen Wrapping-Schlüssel und laden das verschlüsselte Schlüsselmaterial anschließend in hoch. AWS KMS Beim AWS KMS Empfang Ihres verschlüsselten Schlüsselmaterials entschlüsselt es das Schlüsselmaterial mit dem entsprechenden privaten Schlüssel und verschlüsselt das Schlüsselmaterial anschließend erneut unter einem symmetrischen AES-Schlüssel, alles innerhalb eines AWS KMS Hardware-Sicherheitsmoduls (HSM).

Nutzen des Import-Tokens

Der Download enthält ein Import-Token mit Metadaten, das sicherstellt, dass Ihr Schlüsselmaterial korrekt importiert wird. Wenn Sie Ihr verschlüsseltes Schlüsselmaterial hochladen AWS KMS, müssen Sie dasselbe Import-Token hochladen, das Sie in diesem Schritt heruntergeladen haben.

Wählen Sie eine Spezifikation für den öffentlichen Verpackungsschlüssel

Um Ihr Schlüsselmaterial beim Import zu schützen, verschlüsseln Sie es mit einem öffentlichen Wrapping-Schlüssel, von dem Sie herunterladen AWS KMS, und einem unterstützten Wrapping-Algorithmus. Sie wählen eine Schlüsselspezifikation aus, bevor Sie Ihren öffentlichen Verpackungsschlüssel und das Import-Token herunterladen. Alle Wrapping-Schlüsselpaare werden in AWS KMS Hardware-Sicherheitsmodulen (HSMs) generiert. Der private Schlüssel verlässt das HSM niemals im Klartext.

Die wichtigsten Spezifikationen für das RSA-Wrapping

Die Schlüsselspezifikation des öffentlichen Verpackungsschlüssels bestimmt die Länge der Schlüssel in dem RSA-Schlüsselpaar, das Ihr Schlüsselmaterial während des Transports zu AWS KMS schützt. Im Allgemeinen empfehlen wir die Verwendung des am längsten umschließenden öffentlichen Verpackungsschlüssels, der praktisch ist. Wir bieten verschiedene öffentliche Schlüsselspezifikationen für die Verpackung an, um eine Vielzahl von HSMs Schlüsselmanagern zu unterstützen.

AWS KMS unterstützt die folgenden Schlüsselspezifikationen für die RSA-Wrapping-Schlüssel, die für den Import von Schlüsselmaterial aller Art verwendet werden, sofern nicht anders angegeben.

- RSA_4096 (bevorzugt)
- RSA_3072
- RSA_2048

> **Note**
>
> Die folgende Kombination wird NICHT unterstützt: ECC_NIST_P521-Schlüsselmaterial, die RSA_2048-Spezifikation für öffentliche Verpackungsschlüssel und ein RSAES_OAEP_SHA_*-Verpackungsalgorithmus.
> Sie können das Schlüsselmaterial von ECC_NIST_P521 nicht direkt mit einem öffentlichen RSA_2048-Verpackungsschlüssel verpacken. Verwenden Sie einen größeren Verpackungsschlüssel oder einen RSA_AES_KEY_WRAP_SHA_*-WRAP_*-Verpackungsalgorithmus.

SM2 Spezifikation für Verpackungsschlüssel (nur Regionen in China)

AWS KMS unterstützt die folgenden Schlüsselspezifikationen für Wrapping-Schlüssel, die für SM2 den Import von asymmetrischem Schlüsselmaterial verwendet werden.

- SM2

Auswählen des Verpackungsalgorithmus

Um Ihr Schlüsselmaterial während des Imports zu schützen, verschlüsseln Sie es mit dem heruntergeladenen öffentlichen Verpackungsschlüssel und einem unterstützten Verpackungsalgorithmus.

AWS KMS unterstützt mehrere standardmäßige RSA-Wrapping-Algorithmen und einen zweistufigen Hybrid-Wrapping-Algorithmus. Im Allgemeinen empfehlen wir, den sichersten Verpackungsalgorithmus zu verwenden, der mit Ihrem importierten Schlüsselmaterial und den Verpackungsschlüsselspezifikationen kompatibel ist. In der Regel wählen Sie einen Algorithmus aus, der von dem Hardwaresicherheitsmodul (HSM) oder dem Schlüsselverwaltungssystem unterstützt wird, mit dem Ihr Schlüsselmaterial geschützt werden.

In der folgenden Tabelle sind die Verpackungsalgorithmen aufgeführt, die für jeden Typ von Schlüsselmaterial und KMS-Schlüssel unterstützt werden. Die Algorithmen werden in Präferenzreihenfolge aufgeführt.

Schlüsselmaterial	Unterstützter Verpackungsalgorithmus und -Spezifikation
Schlüssel zur symmetrischen Verschlüsselung 256-Bit-AES-Schlüssel SM4 128-Bit-Schlüssel (nur Regionen China)	Verpackungsalgorithmen: RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 Veraltete Verpackungsalgorithmen: PKCS1RSAES_ _V1 **Note** Unterstützt seit dem 10. Oktober 2023 den PKCS1 RSAES_ _V1_5-Wrapping-Algorithmus AWS KMS nicht. Verpackungsschlüsselspezifikationen: RSA_2048 RSA_3072 RSA_4096

Schlüsselmaterial	Unterstützter Verpackungsalgorithmus und -Spezifikation
Asymmetrischer privater RSA-Schlüssel	Verpackungsalgorithmen: RSA_AES_KEY_WRAP_SHA_256 RSA_AES_KEY_WRAP_SHA_1 SM2PKE (nur Regionen China) Verpackungsschlüsselspezifikationen: RSA_2048 RSA_3072 RSA_4096 SM2 (nur Regionen China)
Asymmetrischer privater Schlüssel mit elliptischer Kurve (ECC) Sie können die RSAES_OAEP_SHA_*-Verpackungsalgorithmen nicht mit der RSA_2048-Verpackungsschlüsselspezifikation verwenden, um ECC_NIST_P521-Schlüsselmaterial zu verpacken.	Verpackungsalgorithmen: RSA_AES_KEY_WRAP_SHA_256 RSA_AES_KEY_WRAP_SHA_1 RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 SM2PKE (nur Regionen China) Verpackungsschlüsselspezifikationen: RSA_2048 RSA_3072 RSA_4096 SM2 (nur Regionen China)

Schlüsselmaterial	Unterstützter Verpackungsalgorithmus und -Spezifikation
Asymmetrischer SM2 privater Schlüssel (nur Regionen China)	Verpackungsalgorithmen: RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 SM2PKE (nur Regionen China) Verpackungsschlüsselspezifikationen: RSA_2048 RSA_3072 RSA_4096 SM2 (nur Regionen China)
HMAC-Schlüssel	Verpackungsalgorithmen: RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 Verpackungsschlüsselspezifikationen: RSA_2048 RSA_3072 RSA_4096

> **Note**
>
> Die Algorithmen `RSA_AES_KEY_WRAP_SHA_256` und `RSA_AES_KEY_WRAP_SHA_1` Wrapping werden in China Regionen nicht unterstützt.

- `RSA_AES_KEY_WRAP_SHA_256` – Ein zweistufiger Hybrid-Verpackungsalgorithmus, der die Verschlüsselung Ihres Schlüsselmaterials mit einem von Ihnen generierten symmetrischen AES-Schlüssel und die anschließende Verschlüsselung des symmetrischen AES-Schlüssels mit dem heruntergeladenen öffentlichen RSA-Verpackungsschlüssel und dem RSAES_OAEP_SHA_256-Verpackungsalgorithmus kombiniert.

  Für das `RSA_AES_KEY_WRAP_SHA_*` Verpacken von privatem RSA-Schlüsselmaterial ist ein Wrapping-Algorithmus erforderlich, außer in China Regionen, wo Sie den `SM2PKE` Wrapping-Algorithmus verwenden müssen.
- `RSA_AES_KEY_WRAP_SHA_1` – Ein zweistufiger Hybrid-Verpackungsalgorithmus, der die Verschlüsselung Ihres Schlüsselmaterials mit einem von Ihnen generierten symmetrischen AES-Schlüssel und die anschließende Verschlüsselung des symmetrischen AES-Schlüssels mit dem heruntergeladenen öffentlichen RSA-Verpackungsschlüssel und dem RSAES_OAEP_SHA_1-Verpackungsalgorithmus kombiniert.

  Für das `RSA_AES_KEY_WRAP_SHA_*` Verpacken von privatem RSA-Schlüsselmaterial ist ein Wrapping-Algorithmus erforderlich, außer in China Regionen, wo Sie den `SM2PKE` Wrapping-Algorithmus verwenden müssen.
- `RSAES_OAEP_SHA_256` – Der RSA-Verschlüsselungsalgorithmus mit Optimal Asymmetric Encryption Padding (OAEP) mit der SHA-256-Hash-Funktion.
- `RSAES_OAEP_SHA_1` – Der RSA-Verschlüsselungsalgorithmus mit Optimal Asymmetric Encryption Padding (OAEP) mit der SHA-1-Hash-Funktion.
- `RSAES_PKCS1_V1_5`(Veraltet; unterstützt seit dem 10. Oktober 2023 den Wrapping-Algorithmus RSAES_ PKCS1 _V1_5 AWS KMS nicht) — Der RSA-Verschlüsselungsalgorithmus mit dem in PKCS #1 Version 1.5 definierten Auffüllformat.
- `SM2PKE`(Nur Regionen China) — Ein auf elliptischen Kurven basierender Verschlüsselungsalgorithmus, der von OSCCA in GM/T 0003.4-2012 definiert wurde.

Themen

- Herunterladen des öffentlichen Verpackungsschlüssels und Import-Tokens (Konsole)
- Der öffentliche Schlüssel für die Verpackung und das Import-Token (AWS KMS API) werden heruntergeladen

Herunterladen des öffentlichen Verpackungsschlüssels und Import-Tokens (Konsole)

Sie können die AWS KMS Konsole verwenden, um den öffentlichen Schlüssel für die Verpackung herunterzuladen und das Token zu importieren.

1. Wenn Sie gerade die Schritte zum Erstellen eines KMS-Schlüssel ohne Schlüsselmaterial abgeschlossen haben und sich auf der Seite Download wrapping key and import token (Umhüllungsschlüssel und Import-Token herunterladen) befinden, fahren Sie gleich mit Step 9 fort.
2. Melden Sie sich bei der Konsole AWS Key Management Service (AWS KMS) an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
3. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
4. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.

   > **Tip**
   >
   > Sie können Schlüsselmaterial nur in einen KMS-Schlüssel mit dem Ursprung Extern (Schlüsselmaterial importieren). Dies bedeutet, dass der KMS-Schlüssel ohne Schlüsselmaterial erstellt wurde. Um die Spalte Origin (Ursprung) zu Ihrer Tabelle hinzuzufügen, klicken Sie auf das Symbol „Settings (Einstellungen)“ in der rechten oberen Ecke der Seite ( ). Aktivieren Sie Origin (Ursprung) und wählen Sie dann Confirm (Bestätigen) aus.

5. Wählen Sie den Alias oder die Schlüssel-ID des KMS-Schlüssels aus, dessen Import ausstehend ist.
6. Erweitern Sie den Bereich Cryptographic configuration (kryptografische Konfiguration) und zeigen Sie dessen Werte an. Die Registerkarte wird unter dem Abschnitt General Configuration (allgemeine Konfiguration) angezeigt.

   Sie können Schlüsselmaterial nur in KMS-Schlüssel mit dem Ursprung Extern (Schlüsselmaterial importieren). Weitere Informationen zum Erstellen von KMS-Schlüsseln mit importiertem Schlüsselmaterial finden Sie unter Schlüsselmaterial für AWS KMS Schlüssel importieren.
7. Wählen Sie die Registerkarte Schlüsselmaterial und dann Schlüsselmaterial importieren.

Die Registerkarte Schlüsselmaterial wird nur für KMS-Schlüssel angezeigt, bei denen der Wert für Ursprung Extern (Schlüsselmaterial importieren) ist.

8. Wählen Sie unter Verpackungsschlüssel-Spezifikation auswählen die Konfiguration für Ihren KMS-Schlüssel aus. Nachdem Sie diesen Schlüssel erstellt haben, können Sie die Schlüsselspezifikationen nicht ändern.
9. Wählen Sie für Select wrapping algorithm die Option aus, die Sie zum Verschlüsseln Ihres Schlüsselmaterials verwenden werden. Weitere Informationen zu den Optionen finden Sie unter Verpackungsalgorithmus auswählen.
10. Klicken Sie auf Herunterladen des öffentlichen Verpackungsschlüssels und Import-Tokens und speichern Sie dann die Datei.

    Wenn die Option Next (Weiter) vorhanden ist, wählen Sie Next (Weiter), um den Vorgang fortzusetzen. Um später fortzufahren, wählen Sie Cancel (Abbrechen).
11. Dekomprimieren Sie die `.zip`-Datei, die Sie im vorherigen Schritt gespeichert haben (`Import_Parameters_<key_id>_<timestamp>`).

    Der Ordner enthält die folgenden Dateien:

    - Ein umschließender öffentlicher Schlüssel in eine Datei mit dem Namen. `WrappingPublicKey.bin`
    - Ein Import-Token in einer Datei mit dem Namen `ImportToken.bin`.
    - Eine Textdatei mit dem Namen README.txt. Diese Datei enthält Informationen über den öffentlichen Verpackungsschlüssel, den zu verwendenden Verpackungsschlüssel zum Verschlüsseln Ihres Schlüsselmaterials und das Datum und die Uhrzeit des Ablaufens des öffentlichen Verpackungsschlüssels und des Import-Tokens.
12. Um mit dem Vorgang fortzufahren, nehmen Sie ihn mit dem Schritt Verschlüsseln Ihres Schlüsselmaterials wieder auf.

Der öffentliche Schlüssel für die Verpackung und das Import-Token (AWS KMS API) werden heruntergeladen

Verwenden Sie die GetParametersForImportAPI, um den öffentlichen Schlüssel herunterzuladen und das Token zu importieren. Geben Sie den KMS-Schlüssel an, der mit dem importierten Schlüsselmaterial verknüpft werden soll. Dieser KMS-Schlüssel muss einen Urspungs-Wert von`EXTERNAL` haben.

> **Note**
>
> Sie können kein Schlüsselmaterial für ML-DSA-KMS-Schlüssel importieren.

In diesem Beispiel werden der `RSA_AES_KEY_WRAP_SHA_256`-Verpackungsalgorithmus, die RSA_3072-Spezifikation zum Verpacken öffentlicher Schlüssel und ein Beispiel für eine Schlüssel-ID angegeben. Ersetzen Sie diese Beispielwerte durch gültige Werte für Ihren Download. Für die Schlüssel-ID können Sie eine Schlüssel-ID oder eine ARN des Schlüssels verwenden, aber keinen Aliasnamen oder Alias-ARN.

```
$ aws kms get-parameters-for-import \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --wrapping-algorithm RSA_AES_KEY_WRAP_SHA_256 \
    --wrapping-key-spec RSA_3072
```

Wenn der Befehl erfolgreich ausgeführt wurde, wird eine Ausgabe ähnlich der Folgenden angezeigt:

```
{
    "ParametersValidTo": 1568290320.0,
    "PublicKey": "public key (base64 encoded)",
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "ImportToken": "import token (base64 encoded)"
}
```

Um die Daten für den nächsten Schritt vorzubereiten, dekodiert base64 den öffentlichen Schlüssel und das Import-Token und speichert die dekodierten Werte in Dateien.

So dekodieren Sie den öffentlichen Schlüssel und das Import-Token von base64:

1. Kopieren Sie den base64-codierten öffentlichen Schlüssel (*`public key (base64 encoded)`*in der Beispielausgabe durch), fügen Sie ihn in eine neue Datei ein, und speichern Sie die Datei dann. Geben Sie der Datei einen aussagekräftigen Namen wie `PublicKey.b64`.
2. Verwenden Sie OpenSSL für die base64-Entschlüsselung der Inhalte der Datei und speichern Sie die entschlüsselten Daten in einer neuen Datei. Das folgende Beispiel decodiert die Daten in die Datei, die Sie im vorherigen Schritt gespeichert haben (`PublicKey.b64`), und speichert die Ausgabe in einer neuen Datei mit dem Namen `WrappingPublicKey.bin`.

```
$ openssl enc -d -base64 -A -in PublicKey.b64 -out WrappingPublicKey.bin
```

3. Kopieren Sie das Base64-kodierte Import-Token (*import token (base64 encoded)*in der Beispielausgabe durch), fügen Sie es in eine neue Datei ein, und speichern Sie dann die Datei. Geben Sie der Datei einen aussagekräftigen Namen, wie z. B. `importtoken.b64`.
4. Verwenden Sie OpenSSL für die base64-Entschlüsselung der Inhalte der Datei und speichern Sie die entschlüsselten Daten in einer neuen Datei. Das folgende Beispiel decodiert die Daten in die Datei, die Sie im vorherigen Schritt gespeichert haben (`ImportToken.b64`), und speichert die Ausgabe in einer neuen Datei mit dem Namen `ImportToken.bin`.

```
$ openssl enc -d -base64 -A -in importtoken.b64 -out ImportToken.bin
```

Fahren Sie mit Schritt 3: Verschlüsselung des Schlüsselmaterials fort.

Schritt 3: Verschlüsselung des Schlüsselmaterials

Nachdem Sie den öffentlichen Schlüssel und das Import-Token heruntergeladen haben, verschlüsseln Sie Ihr Schlüsselmaterial mit dem öffentlichen Schlüssel, den Sie heruntergeladen haben und dem von Ihnen angegebenen Verpackungsalgorithmus. Wenn Sie den öffentlichen Schlüssel oder das Import-Token ersetzen oder den Verpackungsalgorithmus ändern müssen, müssen Sie einen neuen öffentlichen Schlüssel und Import-Token herunterladen. Hinweise zu den AWS KMS unterstützten öffentlichen Schlüsseln und Wrapping-Algorithmen finden Sie unter Wählen Sie eine Spezifikation für den öffentlichen Verpackungsschlüssel undAuswählen des Verpackungsalgorithmus.

Das Schlüsselmaterial muss im Binärformat vorliegen. Weitere Informationen hierzu finden Sie unter Anforderungen an importiertes Schlüsselmaterial.

> **Note**
>
> Verschlüsseln und importieren Sie bei asymmetrischen Schlüsselpaaren nur den privaten Schlüssel. AWS KMS leitet den öffentlichen Schlüssel vom privaten Schlüssel ab.
> Die folgende Kombination wird NICHT unterstützt: ECC_NIST_P521-Schlüsselmaterial, die RSA_2048-Spezifikation für öffentliche Verpackungsschlüssel und ein RSAES_OAEP_SHA_*-Verpackungsalgorithmus.
> Sie können das Schlüsselmaterial von ECC_NIST_P521 nicht direkt mit einem öffentlichen RSA_2048-Verpackungsschlüssel verpacken. Verwenden Sie einen

größeren Verpackungsschlüssel oder einen RSA_AES_KEY_WRAP_SHA_*-WRAP_*-Verpackungsalgorithmus.
Die Wrapping-Algorithmen RSA_AES_KEY_WRAP_SHA_256 und RSA_AES_KEY_WRAP_SHA_1 werden in China Regionen nicht unterstützt.

In der Regel verschlüsseln Sie das Schlüsselmaterial, wenn Sie es aus dem Hardwaresicherheitsmodul (HSM) oder Schlüsselverwaltungssystem exportieren. Weitere Informationen zum Exportieren von Schlüsselmaterial im Binärformat finden Sie in der Dokumentation zum HSM oder Schlüsselverwaltungssystem. Sie können sich auch den folgenden Abschnitt ansehen, der eine Machbarkeitsnachweis-Demonstration unter Verwendung von OpenSSL enthält.

Verwenden Sie beim Verschlüsseln Ihres Schlüsselmaterials den gleichen Verpackungsalgorithmus, den Sie beim Herunterladen des öffentlichen Schlüssels und Import-Tokens angegeben haben. Den von Ihnen angegebenen Wrapping-Algorithmus finden Sie im Protokollereignis für die zugehörige Anfrage. CloudTrail GetParametersForImport

Generieren Sie Schlüsselmaterial für Tests

Die folgenden OpenSSL-Befehle generieren Schlüsselmaterial für jeden unterstützten Typ zum Testen. Diese Beispiele dienen nur zu Test- und proof-of-concept Demonstrationszwecken. Verwenden Sie für Produktionssysteme eine sicherere Methode zur Generierung Ihres Schlüsselmaterials, z. B. ein Hardware-Sicherheitsmodul oder ein Schlüsselverwaltungssystem.

Um die privaten Schlüssel von asymmetrischen Schlüsselpaaren in das DER-kodierte Format zu konvertieren, leiten Sie den Befehl zur Erzeugung von Schlüsselmaterial an den folgenden `openssl pkcs8`-Befehl weiter. Der `topk8`-Parameter weist OpenSSL an, einen privaten Schlüssel als Eingabe zu verwenden und einen PKCS-#8-formatierten Schlüssel zurückzugeben. (Das Standardverhalten ist das Gegenteil.)

```
openssl pkcs8 -topk8 -outform der -nocrypt
```

Die folgenden Befehle erzeugen Testschlüsselmaterial für jeden der unterstützten Schlüsseltypen.

- Symmetrischer Verschlüsselungsschlüssel (32 Bytes)

  Dieser Befehl erzeugt einen symmetrischen 256-Bit-Schlüssel (32-Byte-Zufallszeichenfolge) und speichert ihn in der Datei `PlaintextKeyMaterial.bin`. Sie müssen dieses Schlüsselmaterial nicht verschlüsseln.

```
openssl rand -out PlaintextKeyMaterial.bin 32
```

Nur in chinesischen Regionen müssen Sie einen symmetrischen 128-Bit-Schlüssel (16-Byte-Zufallszeichenfolge) generieren.

```
openssl rand -out PlaintextKeyMaterial.bin 16
```

- HMAC-Schlüssel

  Dieser Befehl generiert eine zufällige Byte-Zeichenfolge der angegebenen Größe. Sie müssen dieses Schlüsselmaterial nicht verschlüsseln.

  Die Länge Ihres HMAC-Schlüssels muss der Länge entsprechen, die in der Schlüsselspezifikation des KMS-Schlüssels definiert ist. Wenn der KMS-Schlüssel beispielsweise HMAC_384 lautet, müssen Sie einen 384-Bit-Schlüssel (48 Byte) importieren.

```
openssl rand -out HMAC_224_PlaintextKey.bin 28

openssl rand -out HMAC_256_PlaintextKey.bin 32

openssl rand -out HMAC_384_PlaintextKey.bin 48

openssl rand -out HMAC_512_PlaintextKey.bin 64
```

- RSA-Privatschlüssel

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:2048 | openssl pkcs8 -topk8 -
outform der -nocrypt > RSA_2048_PrivateKey.der

openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:3072 | openssl pkcs8 -topk8 -
outform der -nocrypt > RSA_3072_PrivateKey.der

openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 | openssl pkcs8 -topk8 -
outform der -nocrypt > RSA_4096_PrivateKey.der
```

- ECC-Privatschlüssel

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 | openssl pkcs8 -topk8
 -outform der -nocrypt > ECC_NIST_P256_PrivateKey.der
```

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-384 | openssl pkcs8 -topk8
 -outform der -nocrypt > ECC_NIST_P384_PrivateKey.der

openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-521 | openssl pkcs8 -topk8
 -outform der -nocrypt > ECC_NIST_P521_PrivateKey.der

openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:secp256k1 | openssl pkcs8 -
topk8 -outform der -nocrypt > ECC_SECG_P256K1_PrivateKey.der
```

- SM2 private Schlüssel (nur Regionen China)

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:sm2 | openssl pkcs8 -topk8 -
outform der -nocrypt > SM2_PrivateKey.der
```

Beispiele für die Verschlüsselung von Schlüsselmaterial mit OpenSSL

Die folgenden Beispiele zeigen, wie Sie OpenSSL verwenden, um Ihr Schlüsselmaterial mit dem öffentlichen Schlüssel zu verschlüsseln, den Sie heruntergeladen haben. Um Ihr Schlüsselmaterial mit einem SM2 öffentlichen Schlüssel zu verschlüsseln (nur Regionen China), verwenden Sie die `SM2OfflineOperationHelper`Klasse. Weitere Informationen zu den wichtigsten Materialtypen, die jeder Wrapping-Algorithmus unterstützt, finden Sie unterthe section called “Auswählen des Verpackungsalgorithmus”.

> **Important**
>
> Diese Beispiele dienen nur zur Demonstration des Konzepts. Verwenden Sie bei Produktionssystemen eine sicherere Methode (beispielsweise ein kommerzielles HSM oder Schlüsselverwaltungssystem), um Ihr Schlüsselmaterial zu generieren und zu speichern.
> Die folgende Kombination wird NICHT unterstützt: ECC_NIST_P521-Schlüsselmaterial, die RSA_2048-Spezifikation für öffentliche Verpackungsschlüssel und ein RSAES_OAEP_SHA_*-Verpackungsalgorithmus.
> Sie können das Schlüsselmaterial von ECC_NIST_P521 nicht direkt mit einem öffentlichen RSA_2048-Verpackungsschlüssel verpacken. Verwenden Sie einen größeren Verpackungsschlüssel oder einen RSA_AES_KEY_WRAP_SHA_*-WRAP_*-Verpackungsalgorithmus.

RSAES_OAEP_SHA_1

AWS KMS unterstützt RSAES_OAEP_SHA_1 für symmetrische Verschlüsselungsschlüssel (SYMMETRIC_DEFAULT), private Schlüssel mit elliptischen Kurven (ECC), private Schlüssel und HMAC-Schlüssel. SM2

RSAES_OAEP_SHA_1 wird für private RSA-Schlüssel nicht unterstützt. Außerdem können Sie einen öffentlichen RSA_2048-Verpackungsschlüssel nicht mit einem RSAES_OAEP_SHA_*-Verpackungsalgorithmus verwenden, um einen privaten ECC_NIST_P521 (secp521r1)-Schlüssel zu verpacken. Sie müssen einen größeren öffentlichen Verpackungsschlüssel oder einen RSA_AES_KEY_WRAP-Verpackungsalgorithmus verwenden.

Im folgenden Beispiel wird Ihr Schlüsselmaterial mit dem öffentlichen Schlüssel, den Sie heruntergeladen haben, und dem Verpackungsalgorithmus RSAES_OAEP_SHA_1 verschlüsselt und in der Datei `EncryptedKeyMaterial.bin` gespeichert.

In diesem Beispiel:

- *`WrappingPublicKey.bin`* ist die Datei, die den heruntergeladenen öffentlichen Verpackungsschlüssel enthält.
- *`PlaintextKeyMaterial.bin`* ist die Datei, die das Schlüsselmaterial enthält, das Sie verschlüsseln, z. B. `PlaintextKeyMaterial.bin`, `HMAC_384_PlaintextKey.bin` oder `ECC_NIST_P521_PrivateKey.der`.

```
$ openssl pkeyutl \
    -encrypt \
    -in PlaintextKeyMaterial.bin \
    -out EncryptedKeyMaterial.bin \
    -inkey WrappingPublicKey.bin \
    -keyform DER \
    -pubin \
    -pkeyopt rsa_padding_mode:oaep \
    -pkeyopt rsa_oaep_md:sha1
```

RSAES_OAEP_SHA_256

AWS KMS unterstützt RSAES_OAEP_SHA_256 für symmetrische Verschlüsselungsschlüssel (SYMMETRIC_DEFAULT), private Schlüssel mit elliptischen Kurven (ECC), private Schlüssel und HMAC-Schlüssel. SM2

RSAES_OAEP_SHA_256 wird für private RSA-Schlüssel nicht unterstützt. Außerdem können Sie einen öffentlichen RSA_2048-Verpackungsschlüssel nicht mit einem RSAES_OAEP_SHA_*-Verpackungsalgorithmus verwenden, um einen privaten ECC_NIST_P521 (secp521r1)-Schlüssel zu verpacken. Sie müssen einen größeren öffentlichen Schlüssel oder einen RSA_AES_KEY_WRAP-Verpackungsalgorithmus verwenden.

Im folgenden Beispiel wird Schlüsselmaterial mit dem öffentlichen Schlüssel, den Sie heruntergeladen haben, und dem Verpackungsalgorithmus RSAES_OAEP_SHA_256 verschlüsselt und in der Datei `EncryptedKeyMaterial.bin` gespeichert.

In diesem Beispiel:

- *`WrappingPublicKey.bin`* ist die Datei, die den heruntergeladenen öffentlichen Verpackungsschlüssel enthält. Wenn Sie den öffentlichen Schlüssel über die Konsole heruntergeladen haben, hat diese Datei den Namen `wrappingKey_`*`KMS key_key_ID_timestamp`* (z. B. `wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909`).
- *`PlaintextKeyMaterial.bin`* ist die Datei, die das Schlüsselmaterial enthält, das Sie verschlüsseln, z. B. `PlaintextKeyMaterial.bin`, `HMAC_384_PlaintextKey.bin` oder `ECC_NIST_P521_PrivateKey.der`.

```
$ openssl pkeyutl \
    -encrypt \
    -in PlaintextKeyMaterial.bin \
    -out EncryptedKeyMaterial.bin \
    -inkey WrappingPublicKey.bin \
    -keyform DER \
    -pubin \
    -pkeyopt rsa_padding_mode:oaep \
    -pkeyopt rsa_oaep_md:sha256 \
    -pkeyopt rsa_mgf1_md:sha256
```

RSA_AES_KEY_WRAP_SHA_1

Der Verpackungsalgorithmus RSA_AES_KEY_WRAP_SHA_1 umfasst zwei Verschlüsselungsoperationen.

1. Verpacken Sie Ihr Schlüsselmaterial mit einem von Ihnen generierten symmetrischen AES-Schlüssel und einem symmetrischen AES-Verpackungsalgorithmus.

2. Verschlüsseln Sie den symmetrischen AES-Schlüssel, den Sie verwendet haben, mit dem öffentlichen Schlüssel, den Sie heruntergeladen haben, und dem RSAES_OAEP_SHA_1-Verpackungsalgorithmus.

Der Verpackungsalgorithmus RSA_AES_KEY_WRAP_SHA_1 erfordert OpenSSL Version 3. x oder später.

1. Generieren Sie einen symmetrischen 256-Bit-AES-Schlüssel für die Verschlüsselung

   Dieser Befehl generiert einen symmetrischen AES-Verpackungsschlüssel, der aus 256 zufälligen Bits besteht, und speichert ihn in der Datei `aes-key.bin`

   ```
   # Generate a 32-byte AES symmetric encryption key
   $ openssl rand -out aes-key.bin 32
   ```

2. Verschlüsseln Sie Ihr Schlüsselmaterial mit dem symmetrischen AES-Verschlüsselungsschlüssel

   Dieser Befehl verschlüsselt Ihr Schlüsselmaterial mit dem symmetrischen AES-Verschlüsselungsschlüssel und speichert das verschlüsselte Schlüsselmaterial in der Datei `key-material-wrapped.bin`.

   In diesem Beispielbefehl:

   - *`PlaintextKeyMaterial.bin`* ist die Datei, die das Schlüsselmaterial enthält, das Sie importieren, z. B. `PlaintextKeyMaterial.bin`, `HMAC_384_PlaintextKey.bin`, `RSA_3072_PrivateKey.der` oder `ECC_NIST_P521_PrivateKey.der`.
   - *`aes-key.bin`* ist die Datei, die den symmetrischen 256-Bit-AES-Verschlüsselungsschlüssel enthält, den Sie mit dem vorherigen Befehl erzeugt haben.

   ```
   # Encrypt your key material with the AES symmetric encryption key
   $ openssl enc -id-aes256-wrap-pad \
           -K "$(xxd -p < aes-key.bin | tr -d '\n')" \
           -iv A65959A6 \
           -in PlaintextKeyMaterial.bin\
           -out key-material-wrapped.bin
   ```

3. Verschlüsseln Sie Ihren symmetrischen AES-Verschlüsselungsschlüssel mit dem öffentlichen Schlüssel

   Dieser Befehl verschlüsselt Ihren symmetrischen AES-Verschlüsselungsschlüssel mit dem öffentlichen Schlüssel, den Sie heruntergeladen haben, und dem RSAES_OAEP_SHA_1-Verpackungsalgorithmus, DER-kodiert ihn und speichert ihn in der Datei `aes-key-wrapped.bin`.

   In diesem Beispielbefehl:

   - *`WrappingPublicKey.bin`* ist die Datei, die den heruntergeladenen öffentlichen Verpackungsschlüssel enthält. Wenn Sie den öffentlichen Schlüssel über die Konsole heruntergeladen haben, hat diese Datei den Namen `wrappingKey_`*`KMS key_key_ID_timestamp`* (z. B. `wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909`
   - *`aes-key.bin`* ist die Datei, die den symmetrischen 256-Bit-AES-Verschlüsselungsschlüssel enthält, den Sie im ersten Befehl in dieser Beispielsequenz generiert haben.

```
# Encrypt your AES symmetric encryption key with the downloaded public key
$ openssl pkeyutl \
    -encrypt \
    -in aes-key.bin \
    -out aes-key-wrapped.bin \
    -inkey WrappingPublicKey.bin \
    -keyform DER \
    -pubin \
    -pkeyopt rsa_padding_mode:oaep \
    -pkeyopt rsa_oaep_md:sha1 \
    -pkeyopt rsa_mgf1_md:sha1
```

4. Generieren Sie die zu importierende Datei

   Verketten Sie die Datei mit dem verschlüsselten Schlüsselmaterial und die Datei mit dem verschlüsselten AES-Schlüssel. Speichern Sie sie in der `EncryptedKeyMaterial.bin`-Datei, die Sie in Schritt 4: Importieren des Schlüsselmaterials importieren werden.

   In diesem Beispielbefehl:

- *key-material-wrapped.bin* ist die Datei, die das verschlüsselte Schlüsselmaterial enthält.
- *aes-key-wrapped.bin* ist die Datei, die den verschlüsselten AES-Verschlüsselungsschlüssel enthält.

```
# Combine the encrypted AES key and encrypted key material in a file
$ cat aes-key-wrapped.bin key-material-wrapped.bin > EncryptedKeyMaterial.bin
```

RSA_AES_KEY_WRAP_SHA_256

Der Verpackungsalgorithmus RSA_AES_KEY_WRAP_SHA_256 umfasst zwei Verschlüsselungsoperationen.

1. Verpacken Sie Ihr Schlüsselmaterial mit einem von Ihnen generierten symmetrischen AES-Schlüssel und einem symmetrischen AES-Verpackungsalgorithmus.
2. Verschlüsseln Sie den symmetrischen AES-Schlüssel, den Sie verwendet haben, mit dem öffentlichen Schlüssel, den Sie heruntergeladen haben, und dem RSAES_OAEP_SHA_256-Verpackungsalgorithmus.

Der Verpackungsalgorithmus RSA_AES_KEY_WRAP_SHA_256 erfordert OpenSSL Version 3. x oder später.

1. Generieren Sie einen symmetrischen 256-Bit-AES-Schlüssel für die Verschlüsselung

   Dieser Befehl generiert einen symmetrischen AES-Verpackungsschlüssel, der aus 256 zufälligen Bits besteht, und speichert ihn in der Datei `aes-key.bin`

   ```
   # Generate a 32-byte AES symmetric encryption key
   $ openssl rand -out aes-key.bin 32
   ```

2. Verschlüsseln Sie Ihr Schlüsselmaterial mit dem symmetrischen AES-Verschlüsselungsschlüssel

   Dieser Befehl verschlüsselt Ihr Schlüsselmaterial mit dem symmetrischen AES-Verschlüsselungsschlüssel und speichert das verschlüsselte Schlüsselmaterial in der Datei `key-material-wrapped.bin`.

In diesem Beispielbefehl:

- *PlaintextKeyMaterial.bin* ist die Datei, die das Schlüsselmaterial enthält, das Sie importieren, z. B. `PlaintextKeyMaterial.bin`, `HMAC_384_PlaintextKey.bin`, `RSA_3072_PrivateKey.der` oder `ECC_NIST_P521_PrivateKey.der`.
- *aes-key.bin* ist die Datei, die den symmetrischen 256-Bit-AES-Verschlüsselungsschlüssel enthält, den Sie mit dem vorherigen Befehl erzeugt haben.

```
# Encrypt your key material with the AES symmetric encryption key
$ openssl enc -id-aes256-wrap-pad \
        -K "$(xxd -p < aes-key.bin | tr -d '\n')" \
        -iv A65959A6 \
        -in PlaintextKeyMaterial.bin\
        -out key-material-wrapped.bin
```

3. Verschlüsseln Sie Ihren symmetrischen AES-Verschlüsselungsschlüssel mit dem öffentlichen Schlüssel

   Dieser Befehl verschlüsselt Ihren symmetrischen AES-Verschlüsselungsschlüssel mit dem öffentlichen Schlüssel, den Sie heruntergeladen haben, und dem RSAES_OAEP_SHA_256-Verpackungsalgorithmus, DER-kodiert ihn und speichert ihn in der Datei `aes-key-wrapped.bin`.

   In diesem Beispielbefehl:

   - *WrappingPublicKey.bin* ist die Datei, die den heruntergeladenen öffentlichen Verpackungsschlüssel enthält. Wenn Sie den öffentlichen Schlüssel über die Konsole heruntergeladen haben, hat diese Datei den Namen `wrappingKey_`*KMS key_key_ID_timestamp* (z. B. `wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909`
   - *aes-key.bin* ist die Datei, die den symmetrischen 256-Bit-AES-Verschlüsselungsschlüssel enthält, den Sie im ersten Befehl in dieser Beispielsequenz generiert haben.

```
# Encrypt your AES symmetric encryption key with the downloaded public key
$ openssl pkeyutl \
    -encrypt \
```

```
    -in aes-key.bin \
    -out aes-key-wrapped.bin \
    -inkey WrappingPublicKey.bin \
    -keyform DER \
    -pubin \
    -pkeyopt rsa_padding_mode:oaep \
    -pkeyopt rsa_oaep_md:sha256 \
    -pkeyopt rsa_mgf1_md:sha256
```

4. Generieren Sie die zu importierende Datei

   Verketten Sie die Datei mit dem verschlüsselten Schlüsselmaterial und die Datei mit dem verschlüsselten AES-Schlüssel. Speichern Sie sie in der `EncryptedKeyMaterial.bin`-Datei, die Sie in Schritt 4: Importieren des Schlüsselmaterials importieren werden.

   In diesem Beispielbefehl:

   - *`key-material-wrapped.bin`* ist die Datei, die das verschlüsselte Schlüsselmaterial enthält.
   - *`aes-key-wrapped.bin`* ist die Datei, die den verschlüsselten AES-Verschlüsselungsschlüssel enthält.

```
# Combine the encrypted AES key and encrypted key material in a file
$ cat aes-key-wrapped.bin key-material-wrapped.bin > EncryptedKeyMaterial.bin
```

Fahren Sie mit Schritt 4: Importieren des Schlüsselmaterials fort.

Schritt 4: Importieren des Schlüsselmaterials

Nachdem Sie Ihr Schlüsselmaterial verschlüsselt haben, können Sie das Schlüsselmaterial importieren, um es mit einem AWS KMS key zu verwenden. Um Schlüsselmaterial zu importieren, laden Sie das verschlüsselte Schlüsselmaterial aus Schritt 3: Verschlüsselung des Schlüsselmaterials und den Import-Token, den Sie von Schritt 2: Herunterladen des öffentlichen Verpackungsschlüssels und des Import-Tokens heruntergeladen haben, hoch. Sie müssen das Schlüsselmaterial in den gleichen KMS-Schlüssel importieren, den Sie auch beim Herunterladen des öffentlichen Schlüssels und Import-Tokens angegeben haben. Wenn Schlüsselmaterial erfolgreich importiert wird, ändert sich der Schlüsselstatus des KMS-Schlüssels auf `Enabled`, und Sie können den KMS-Schlüssel in kryptografischen Operationen verwenden.

Wenn Sie Schlüsselmaterial importieren, können Sie ein optionales Ablaufdatum für das Schlüsselmaterial festlegen. Wenn das Schlüsselmaterial abgelaufen ist, löscht AWS KMS das Schlüsselmaterial und der KMS-Schlüssel kann nicht mehr verwendet werden. Nachdem Sie Ihr Schlüsselmaterial importiert haben, können Sie das Ablaufdatum für den aktuellen Import nicht mehr festlegen, ändern oder stornieren. Um diese Werte zu ändern, müssen Sie dasselbe Schlüsselmaterial erneut importieren.

Bei allen KMS-Schlüsseln mit `EXTERNAL` Ursprung wird das erste Schlüsselmaterial, das in sie importiert wurde, aktuell und dauerhaft mit ihnen verknüpft. Symmetrische Verschlüsselungsschlüssel mit `EXTERNAL` Ursprung in einer Region unterstützen die Rotation nach Bedarf. Sie können importierten Schlüsseln, die eine Rotation nach Bedarf unterstützen, mehrere Schlüsselmaterialien zuordnen. Sie müssen den `importType` Parameter für `NEW_KEY_MATERIAL` die ImportKeyMaterialAktion auf festlegen, um neues Schlüsselmaterial einem KMS-Schlüssel zuzuordnen. Dieses Schlüsselmaterial ist dem Schlüssel erst dauerhaft zugeordnet, wenn Sie die RotateKeyOnDemandAktion ausführen. Bis dahin ist dieses Schlüsselmaterial in `PENDING_ROTATION` Ordnung. Der Standardwert des optionalen `ImportType` Parameters ist`EXISTING_KEY_MATERIAL`. Wenn Sie den `ImportType` Parameter weglassen oder als angeben`EXISTING_KEY_MATERIAL`, müssen Sie ein Schlüsselmaterial importieren, das zuvor dem KMS-Schlüssel zugeordnet war.

Bei asymmetrischen, HMAC- oder regionsübergreifenden KMS-Schlüsseln mit `EXTERNAL` Ursprung kann dem Schlüssel immer nur ein Schlüsselmaterial zugeordnet werden. AWS KMS lehnt ImportKeyMaterialAPI-Anfragen mit dem Parameter ab. `ImportType`

Wenn alle Schlüsselmaterialien importiert werden, die dauerhaft mit einem KMS-Schlüssel verknüpft sind, steht der KMS-Schlüssel für kryptografische Operationen zur Verfügung. Wenn eines dieser Schlüsselmaterialien gelöscht wird oder abläuft, ändert sich der Status des KMS-Schlüssels `PendingImport` und der Schlüssel ist für kryptografische Operationen unbrauchbar.

Um Schlüsselmaterial zu importieren, können Sie die AWS KMS Konsole oder die ImportKeyMaterialAPI verwenden. Sie können die API direkt verwenden, indem Sie HTTP-Anfragen stellen oder ein AWS SDKs, AWS Command Line Interfaceoder verwenden AWS -Tools für PowerShell.

Wenn Sie das Schlüsselmaterial importieren, wird Ihrem AWS CloudTrail Protokoll ein ImportKeyMaterialEintrag hinzugefügt, um den `ImportKeyMaterial` Vorgang aufzuzeichnen. Der CloudTrail Eintrag ist derselbe, unabhängig davon, ob Sie die AWS KMS Konsole oder die AWS KMS API verwenden.

Festlegen einer Ablaufzeit (optional)

Wenn Sie das Schlüsselmaterial für Ihren KMS-Schlüssel importieren, können Sie optional ein Ablaufdatum und eine Ablaufzeit für das Schlüsselmaterial von bis zu 365 Tagen ab dem Importdatum festlegen. Wenn importiertes Schlüsselmaterial abläuft, AWS KMS wird es gelöscht. Diese Aktion ändert den Status des KMS-Schlüssels zu `PendingImport`, wodurch verhindert wird, dass er in kryptografischen Operationen verwendet wird. Wenn Sie den KMS-Schlüssel verwenden möchten, müssen Sie das Schlüsselmaterial erneut importieren.

Wenn Sie sicherstellen, dass importiertes Schlüsselmaterial häufig abläuft, können Sie zwar die gesetzlichen Anforderungen erfüllen, jedoch birgt dies ein zusätzliches Risiko für Daten, die unter dem KMS-Schlüssel verschlüsselt sind. Solange Sie keine Kopie des ursprünglichen Schlüsselmaterials neu importieren, ist ein KMS-Schlüssel mit abgelaufenem Schlüsselmaterial unbrauchbar, und alle unter dem KMS-Schlüssel verschlüsselten Daten sind unzugänglich. Wenn Sie das Schlüsselmaterial aus irgendeinem Grund, einschließlich des Verlusts Ihrer Kopie des ursprünglichen Schlüsselmaterials, nicht erneut importieren, ist der KMS-Schlüssel dauerhaft unbrauchbar, und die unter dem KMS-Schlüssel verschlüsselten Daten sind nicht wiederherstellbar.

Um dieses Risiko zu minimieren, sollten Sie sicherstellen, dass auf Ihre Kopie des importierten Schlüsselmaterials zugegriffen werden kann, und ein System einrichten, das das Schlüsselmaterial löscht und erneut importiert, bevor es abläuft und Ihre Arbeitslast unterbricht. AWS Wir empfehlen Ihnen, einen Alarm für den Ablauf Ihres importierten Schlüsselmaterials einzustellen, der Ihnen genügend Zeit gibt, das Schlüsselmaterial vor Ablauf erneut zu importieren. Sie können Ihre CloudTrail Protokolle auch verwenden, um Vorgänge zu überprüfen, bei denen Schlüsselmaterial importiert (und erneut importiert) und importiertes Schlüsselmaterial gelöscht wird, sowie den AWS KMS Vorgang zum Löschen abgelaufener Schlüsselmaterialien.

AWS KMS Das gelöschte Schlüsselmaterial kann nicht wiederhergestellt, wiederhergestellt oder reproduziert werden. Anstatt eine Ablaufzeit festzulegen, können Sie das importierte Schlüsselmaterial programmgesteuert löschen und in regelmäßigen Abständen erneut importieren. Die Anforderungen für die Beibehaltung einer Kopie des ursprünglichen Schlüsselmaterials sind jedoch dieselben.

Sie legen fest, ob und wann importiertes Schlüsselmaterial abläuft, wenn Sie das Schlüsselmaterial importieren. Sie können das Ablaufdatum jedoch ein- und ausschalten oder eine neue Ablaufzeit festlegen, indem Sie das Schlüsselmaterial erneut importieren. Verwenden Sie den `ExpirationModel` Parameter von ImportKeyMaterial, um den Ablauf ein- (`KEY_MATERIAL_EXPIRES`) und auszuschalten (`KEY_MATERIAL_DOES_NOT_EXPIRE`), und den

`ValidTo` Parameter, um die Ablaufzeit festzulegen. Der maximale Zeitraum beträgt 365 Tage ab dem Import der Daten. Es gibt kein Minimum, aber der Zeitpunkt muss in der Zukunft liegen.

Geben Sie die Beschreibung des wichtigsten Materials ein

Symmetrischen Verschlüsselungsschlüsseln mit `EXTERNAL` Ursprung in einer Region können mehrere Schlüsselmaterialien zugeordnet sein. Sie können beim Import von Schlüsselmaterial in solche Schlüssel eine optionale Beschreibung des Schlüsselmaterials angeben. Anhand der Beschreibung kann nachverfolgt werden, wo das entsprechende Schlüsselmaterial dauerhaft draußen AWS KMS aufbewahrt wird.

Schlüsselmaterial erneut importieren

Wenn Sie einen KMS-Schlüssel mit importiertem Schlüsselmaterial verwalten, müssen Sie das Schlüsselmaterial möglicherweise erneut importieren. Sie können das Schlüsselmaterial erneut importieren, um ablaufendes odergelöschtes Schlüsselmaterial zu ersetzen oder um das Ablaufmodell oder das Ablaufdatum des Schlüsselmaterials zu ändern.

Sie können Schlüsselmaterial jederzeit nach einem beliebigen Zeitplan, der Ihren Sicherheitsanforderungen entspricht, wieder importieren. Sie müssen nicht warten, bis das Schlüsselmaterial seine Ablaufzeit erreicht oder fast erreicht hat.

Das Verfahren zum erneuten Importieren von Schlüsselmaterial entspricht dem Verfahren, das Sie für den ersten Import des Schlüsselmaterials verwenden, mit den folgenden Ausnahmen.

- Verwenden Sie einen vorhandenen KMS-Schlüssel, anstatt einen neuen KMS-Schlüssel zu erstellen. Sie können Schritt 1 des Importvorgangs überspringen.
- Wenn Sie Schlüsselmaterial erneut importieren, können Sie das Ablaufmodell und das Ablaufdatum ändern.

Jedes Mal, wenn Sie Schlüsselmaterial in einen KMS-Schlüssel importieren, müssen Sie einen neuen Verpackungsschlüssel und ein Import-Token für den KMS-Schlüssel herunterladen und verwenden. Das Verpackungsverfahren hat keine Auswirkungen auf den Inhalt des Schlüsselmaterials. Sie können also unterschiedliche öffentliche Verpackungsschlüssel und unterschiedliche Verpackungsalgorithmen verwenden, um das gleiche Schlüsselmaterial zu importieren.

Neues Schlüsselmaterial importieren

Um bei Bedarf einen KMS-Schlüssel mit symmetrischer Verschlüsselung und importiertem Schlüsselmaterial zu rotieren, müssen Sie zunächst neues Schlüsselmaterial importieren, das dem Schlüssel noch nicht zugeordnet war. Verwenden Sie die ImportKeyMaterialOperation mit dem `ImportType` Parametersatz auf, `NEW_KEY_MATERIAL` um diese Aufgabe auszuführen. Auf diese Weise importiertes Schlüsselmaterial bleibt so `PENDING_ROTATION` lange erhalten, bis Sie den RotateKeyOnDemandVorgang ausführen oder den Schlüssel in der drehen AWS Management Console. Ein KMS-Schlüssel kann zu jedem Zeitpunkt höchstens ein Schlüsselmaterial im `PENDING_ROTATION` Status haben.

Importieren von Schlüsselmaterial (Konsole)

Sie können den verwenden AWS Management Console , um Schlüsselmaterial zu importieren.

1. Wenn Sie sich auf der Seite Ihr verpacktes Schlüsselmaterial hochladen befinden, fahren Sie mit Step 8 fort.
2. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
3. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
4. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
5. Wählen Sie die Schlüssel-ID oder den Alias des KMS-Schlüssels, für den Sie den öffentlichen Schlüssel und das Import-Token heruntergeladen haben.
6. Erweitern Sie den Bereich Cryptographic configuration (kryptografische Konfiguration) und zeigen Sie dessen Werte an. Die Registerkarten befinden sich auf der Detailseite für einen KMS-Schlüssel unter dem Abschnitt General configuration (allgemeine Konfiguration).

   Sie können Schlüsselmaterial nur in KMS-Schlüssel mit dem Ursprung Extern (Schlüsselmaterial importieren). Weitere Informationen zum Erstellen eines KMS-Schlüssels mit importiertem Schlüsselmaterial finden Sie unter Schlüsselmaterial für AWS KMS Schlüssel importieren.
7. Wählen Sie für asymmetrische Schlüssel, HMAC-Schlüssel und Schlüssel mit mehreren Regionen die Registerkarte Schlüsselmaterial und dann Schlüsselmaterial importieren. Wählen Sie für symmetrische Verschlüsselungsschlüssel mit einer Region die Registerkarte Schlüsselmaterial und Rotationen. Wählen Sie dann entweder Ursprüngliches Schlüsselmaterial importieren oder Neues Schlüsselmaterial importieren oder Schlüsselmaterial erneut importieren.

Die Option Schlüsselmaterial erneut importieren ist im `Actions` Menü in der Tabelle Schlüsselmaterialien verfügbar.

Wenn Sie das Schlüsselmaterial und Import-Token heruntergeladen haben und das Schlüsselmaterial verschlüsselt haben, wählen Sie Weiter.

8. Gehen Sie im Abschnitt Verschlüsseltes Schlüsselmaterial und Import-Token wie folgt vor.

   a. Wählen Sie unter Verpacktes Schlüsselmaterial Datei auswählen aus. Laden Sie dann die Datei hoch, die das durch den Umhüllungsschlüssel geschützte Schlüsselmaterial enthält.

   b. Wählen Sie unter Token importieren die Option Datei wählen. Laden Sie die heruntergeladene Datei mit dem Import-Token hoch.

9. Wählen Sie im Bereich Expiration option (Ablaufoption) aus, ob das Schlüsselmaterial abläuft. Um ein Ablaufdatum und eine Uhrzeit festzulegen, wählen Sie Key material expires (Schlüsselmaterial läuft ab) und wählen Sie dann Datum und Uhrzeit über das Kalendersteuerelement aus. Sie können ein Datum angeben, das bis zu 365 Tage nach dem aktuellen Datum und der aktuellen Uhrzeit liegt.

10. Bei symmetrischen Verschlüsselungsschlüsseln können Sie optional eine Beschreibung für das zu importierende Schlüsselmaterial angeben.

11. Wählen Sie Schlüsselmaterial importieren aus.

Schlüsselmaterial importieren (AWS KMS API)

Verwenden Sie den ImportKeyMaterialVorgang, um Schlüsselmaterial zu importieren. Für die folgenden Beispiele wird die AWS CLIverwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Zur Verwendung dieses Beispiels gehen Sie wie folgt vor:

1. Ersetzen Sie *`1234abcd-12ab-34cd-56ef-1234567890ab`* durch eine Schlüssel-ID des KMS-Schlüssels, den Sie beim Herunterladen des öffentlichen Schlüssels und des Import-Tokens angegeben haben. Verwenden Sie zum Identifizieren des KMS-Schlüssels seine Schlüssel-ID oder den Schlüssel-ARN. Sie können für diese Operation keinen Alias-Namen oder Alias-ARN verwenden.

2. Ersetzen Sie *`EncryptedKeyMaterial.bin`* durch den Namen der Datei, die das verschlüsselten Schlüsselmaterial enthält.

3. Ersetzen Sie *`ImportToken.bin`* durch den Namen der Datei, die den Import-Token enthält.

4. Wenn das importierte Schlüsselmaterial ablaufen soll, setzen Sie den Wert für den `expiration-model`-Parameter auf den Standardwert zurück, `KEY_MATERIAL_EXPIRES`, oder lassen Sie den `expiration-model`-Parameter weg. Ersetzen Sie dann den Wert des `valid-to`-Parameters mit dem Datum und der Uhrzeit, zu der das Schlüsselmaterial ablaufen soll. Datum und Uhrzeit können bis zu 365 Tage ab dem Zeitpunkt der Anfrage betragen.

```
$ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
    --import-token fileb://ImportToken.bin \
    --expiration-model KEY_MATERIAL_EXPIRES \
    --valid-to 2023-06-17T12:00:00-08:00
```

   Wenn das importierte Schlüsselmaterial nicht ablaufen soll, setzen Sie den Wert für den `expiration-model`-Parameter auf`KEY_MATERIAL_DOES_NOT_EXPIRE` und lassen Sie den `valid-to`-Parameter aus dem Befehl weg.

```
$ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
    --import-token fileb://ImportToken.bin \
    --expiration-model KEY_MATERIAL_DOES_NOT_EXPIRE
```

5. Wenn Sie neues Schlüsselmaterial importieren möchten, das zuvor nicht mit dem KMS-Schlüssel verknüpft war, legen Sie den `ImportType` Parameter auf fest`NEW_KEY_MATERIAL`. Diese Option kann nur mit symmetrischen Verschlüsselungsschlüsseln aus einer Region verwendet werden. Für diese Schlüssel können Sie im folgenden Befehlszeilenbeispiel auch den optionalen `KeyMaterialDescription` Parameter verwenden, um eine Beschreibung für das importierte Schlüsselmaterial festzulegen:

```
$ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
    --import-token fileb://ImportToken.bin \
    --expiration-model KEY_MATERIAL_EXPIRES \
    --valid-to 2023-06-17T12:00:00-08:00 \
    --import-type NEW_KEY_MATERIAL \
    --key-material-description "Q2 2025 Rotation"
```

> **Tip**
>
> Wenn der Befehl nicht erfolgreich ist, wird möglicherweise eine `KMSInvalidStateException` oder eine `NotFoundException` angezeigt. Sie können die Anfrage wiederholen.

Erstellen Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher

Nachdem Sie einen AWS CloudHSM Schlüsselspeicher erstellt haben, können Sie ihn AWS KMS keys in Ihrem Schlüsselspeicher erstellen. Es muss sich um KMS-Schlüssel mit symmetrischer Verschlüsselung und AWS KMS generiertem Schlüsselmaterial handeln. Sie können asymmetrische KMS-Schlüssel, HMAC-KMS-Schlüssel oder KMS-Schlüssel mit importiertem Schlüsselmaterial nicht in einem benutzerdefinierten Schlüsselspeicher erstellen. Außerdem können Sie auch keine KMS-Schlüssel mit symmetrischer Verschlüsselung in einem benutzerdefinierten Schlüsselspeicher verwenden, um asymmetrische Datenschlüsselpaare zu generieren. KMS kann nicht IPv6 mit AWS CloudHSM Schlüsselspeichern kommunizieren.

Um einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher zu erstellen, muss der AWS CloudHSM Schlüsselspeicher mit dem zugehörigen AWS CloudHSM Cluster verbunden sein und der Cluster muss mindestens zwei aktive Mitglieder HSMs in unterschiedlichen Availability Zones enthalten. Den Verbindungsstatus und die Anzahl der HSMs Verbindungen finden Sie auf der Seite mit den AWS CloudHSM Schlüsselspeichern in der AWS Management Console. Wenn Sie die API-Operationen verwenden, verwenden Sie den DescribeCustomKeyStoresVorgang, um zu überprüfen, ob der AWS CloudHSM Schlüsselspeicher verbunden ist. Verwenden Sie den AWS CloudHSM DescribeClustersVorgang, um die Anzahl der aktiven Personen HSMs im Cluster und deren Availability Zones zu überprüfen.

Wenn Sie einen KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher erstellen, AWS KMS wird der KMS-Schlüssel in erstellt AWS KMS. Es erstellt jedoch das Schlüsselmaterial für den KMS-Schlüssel im zugehörigen AWS CloudHSM Cluster. Insbesondere AWS KMS meldet es sich als die von Ihnen erstellte `kmsuser` CU beim Cluster an. Anschließend erstellt AWS KMS einen persistenten, nicht extrahierbaren, symmetrischen 256-Bit-AES-Schlüssel (Advanced Encryption Standard) in dem Cluster und legt den Wert für das Schlüsselbezeichnungsattribut, das nur innerhalb des Clusters sichtbar ist, auf den Amazon-Ressourcenname (ARN) des KMS-Schlüssel fest.

Wenn der Befehl erfolgreich ausgeführt wurde, lautet der Schlüsselstatus des neuen KMS-Schlüssels `Enabled` und der Ursprung `AWS_CLOUDHSM`. Der Ursprung eines KMS-Schlüssels kann nicht mehr geändert werden, nachdem Sie ihn erstellt haben. Wenn Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher in der AWS KMS Konsole oder mithilfe des DescribeKeyVorgangs anzeigen, können Sie typische Eigenschaften wie die Schlüssel-ID, den Schlüsselstatus und das Erstellungsdatum sehen. Daneben können Sie aber auch die ID des benutzerdefinierten Schlüsselspeichers sowie (optional) die AWS CloudHSM -Cluster-ID.

Wenn Ihr Versuch, einen KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher zu erstellen, fehlschlägt, können Sie anhand der Fehlermeldung die Ursache ermitteln. Dies kann darauf hindeuten, dass der AWS CloudHSM Schlüsselspeicher nicht verbunden ist (`CustomKeyStoreInvalidStateException`) oder HSMs dass der zugehörige AWS CloudHSM Cluster nicht über die beiden aktiven Elemente verfügt, die für diesen Vorgang erforderlich sind (`CloudHsmClusterInvalidConfigurationException`). Weitere Informationen dazu finden Sie unter Fehlerbehebung für einen Custom Key Store.

Ein Beispiel für das AWS CloudTrail Protokoll des Vorgangs, der einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher erstellt, finden Sie unterCreateKey.

Erstellen Sie einen neuen KMS-Schlüssel in Ihrem CloudHSM-Schlüsselspeicher

Sie können einen KMS-Schlüssel mit symmetrischer Verschlüsselung in Ihrem AWS CloudHSM Schlüsselspeicher in der AWS KMS Konsole oder mithilfe des Vorgangs erstellen. CreateKey

Mithilfe der Konsole AWS KMS

Gehen Sie wie folgt vor, um einen KMS-Schlüssel mit symmetrischer Verschlüsselung in einem AWS CloudHSM Schlüsselspeicher zu erstellen.

> **Note**
>
> Nehmen Sie keine vertraulichen oder sensiblen Informationen in den Alias, in der Beschreibung oder in den Tags auf. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.

2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Klicken Sie auf Create key.
5. Wählen Sie Symmetric (Symmetrisch).
6. Unter Key usage (Schlüsselverwendung) ist die Option Encrypt and decrypt (Verschlüsseln und Entschlüsseln) für Sie ausgewählt. Ändern Sie dies nicht.
7. Wählen Sie Advanced options (Erweiterte Optionen) aus.
8. Wählen Sie unter Ursprung des Schlüsselmaterials die Option AWS CloudHSM - Schlüsselspeicher aus.

   Sie können keinen Schlüssel für mehrere Regionen in einem Schlüsselspeicher erstellen. AWS CloudHSM
9. Wählen Sie Weiter aus.
10. Wählen Sie einen AWS CloudHSM Schlüsselspeicher für Ihren neuen KMS-Schlüssel aus. Um einen neuen AWS CloudHSM Schlüsselspeicher zu erstellen, wählen Sie Benutzerdefinierten Schlüsselspeicher erstellen.

    Der AWS CloudHSM Schlüsselspeicher, den Sie auswählen, muss den Status Verbunden haben. Der zugehörige AWS CloudHSM Cluster muss aktiv sein und mindestens zwei aktive Cluster HSMs in unterschiedlichen Availability Zones enthalten.

    Hilfe zum Herstellen einer Verbindung mit einem AWS CloudHSM Schlüsselspeicher finden Sie unterVerbindung zu einem AWS CloudHSM Schlüsselspeicher trennen. Hilfe zum Hinzufügen HSMs finden Sie unter Hinzufügen eines HSM im AWS CloudHSM Benutzerhandbuch.
11. Wählen Sie Weiter aus.
12. Geben Sie einen Alias und eine optionale Beschreibung für den KMS-Schlüssel ein.
13. (Optional). Fügen Sie auf der Seite Add Tags (Tags hinzufügen) Tags hinzu, um Ihre KMS-Schlüssel identifizieren zu können und sie zu kategorisieren.

    Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenverteilungsbericht generiert, in dem die Nutzung und die Kosten nach Stichwörtern zusammengefasst sind. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Schlagworte in AWS KMS und ABAC für AWS KMS.

14. Wählen Sie Weiter aus.
15. Sie können im Abschnitt Key administrators (Schlüsseladministratoren) die IAM-Benutzer und -Rollen auswählen, die den KMS-Schlüssel verwalten dürfen. Weitere Informationen finden Sie unter Erlaubt Schlüsseladministratoren die Verwaltung des KMS-Schlüssels.

> **Hinweise**
>
> Daneben können IAM-Benutzer und -Rollen die erforderlichen Berechtigungen zur Verwendung des KMS-Schlüssel auch über IAM-Richtlinien erhalten.
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter der Anweisungs-ID `"Allow access for Key Administrators"` wichtige Administratoren hinzu. Eine Änderung dieser Anweisungs-ID kann sich darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Anweisung vornehmen.

16. (Optional) Um zu verhindern, dass diese Schlüsseladministratoren diesen KMS-Schlüssel löschen, deaktivieren Sie das Kontrollkästchen unten auf der Seite neben Allow key administrators to delete this key (Administratoren erlauben, diesen Schlüssel zu löschen).
17. Wählen Sie Weiter aus.
18. Wählen Sie im Abschnitt Dieses Konto die IAM-Benutzer und -Rollen aus AWS-Konto , die den KMS-Schlüssel für kryptografische Operationen verwenden können. Weitere Informationen finden Sie unter Erlaubt Schlüsselbenutzern die Verwendung des KMS-Schlüssels.

> **Hinweise**
>
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
> Die AWS KMS Konsole fügt der Schlüsselrichtlinie unter den Anweisungskennungen `"Allow use of the key"` und Schlüsselbenutzer hinzu. `"Allow attachment of persistent resources"` Das Ändern dieser Anweisungskennungen kann sich

darauf auswirken, wie die Konsole Aktualisierungen anzeigt, die Sie an der Aussage vornehmen.

19. (Optional) Sie können anderen erlauben, diesen KMS-Schlüssel für kryptografische Operationen AWS-Konten zu verwenden. Wählen Sie dazu unten auf der Seite im AWS-Konten Abschnitt Andere die Option Weiteres hinzufügen aus AWS-Konto und geben Sie die AWS-Konto ID eines externen Kontos ein. Wiederholen Sie diesen Schritt, um weitere externe Konten hinzufügen.

> **Note**
>
> Administratoren der anderen AWS-Konten müssen ebenfalls Zugriff auf den KMS-Schlüssel gewähren, indem sie IAM-Richtlinien für ihre Benutzer erstellen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

20. Wählen Sie Weiter aus.
21. Lesen Sie die wichtigsten Richtlinienerklärungen für den Schlüssel. Um Änderungen an der wichtigsten Richtlinie vorzunehmen, wählen Sie Bearbeiten aus.
22. Wählen Sie Weiter aus.
23. Überprüfen Sie die gewählten Einstellungen. Sie können immer noch zurückgehen und alle Einstellungen ändern.
24. Wählen Sie danach Finish (Fertigstellen) aus.

Wenn der Vorgang erfolgreich ist, wird auf dem Display der neue KMS-Schlüssel in dem von Ihnen ausgewählten AWS CloudHSM Schlüsselspeicher angezeigt. Wenn Sie den Namen oder Alias des neuen KMS-Schlüssels auswählen, werden auf der zugehörigen Detailseite auf der Registerkarte Kryptografische Konfiguration der Ursprung des KMS-Schlüssels (AWS CloudHSM), der Name, die ID und der Typ des benutzerdefinierten Schlüsselspeichers sowie die ID des AWS CloudHSM Clusters angezeigt. Wenn der Vorgang fehlschlägt, wird unter eine Fehlermeldung mit einer Beschreibung des Fehlers angezeigt.

> **Tip**
>
> Sie können die KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher leichter identifizieren, wenn Sie auf der Seite Customer managed keys (kundenverwaltete Schlüssel) festlegen, dass die Spalte Custom key store ID (ID des benutzerdefinierten

Schlüsselspeichers) angezeigt wird. Klicken Sie auf das Zahnradsymbol rechts oben und wählen Sie dann Custom key store ID (ID des benutzerdefinierten Schlüsselspeichers) aus. Details hierzu finden Sie unter Passen Sie Ihre Konsolenansicht an.

Verwenden der API AWS KMS

Verwenden Sie den CreateKeyVorgang, um einen neuen AWS KMS key (AWS CloudHSM KMS-Schlüssel) in Ihrem Schlüsselspeicher zu erstellen. Verwenden Sie den Parameter `CustomKeyStoreId`, um den benutzerdefinierten Schlüsselspeicher zu identifizieren und den `Origin`-Wert auf `AWS_CLOUDHSM` festzulegen.

Gegebenenfalls können Sie auch mit dem Parameter `Policy` eine Schlüsselrichtlinie angeben. Sie können die Schlüsselrichtlinie (PutKeyPolicy) jederzeit ändern und optionale Elemente wie eine Beschreibung und Tags hinzufügen.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Das folgende Beispiel beginnt mit einem Aufruf des DescribeCustomKeyStoresVorgangs, um zu überprüfen, ob der AWS CloudHSM Schlüsselspeicher mit dem zugehörigen AWS CloudHSM Cluster verbunden ist. Diese Produktion gibt standardmäßig alle benutzerdefinierten Schlüsselspeicher innerhalb des Kontos und der Region zurück. Um nur einen bestimmten AWS CloudHSM Schlüsselspeicher zu beschreiben, verwenden Sie dessen `CustomKeyStoreName` Parameter `CustomKeyStoreId` oder (aber nicht beide).

Wenn Sie diesen Befehl ausführen, denken Sie daran, die ID des benutzerdefinierten Schlüsselspeichers in dem Beispiel durch eine gültige ID zu ersetzen.

> **Note**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in die Felder `Description` oder `Tags` ein. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben im Klartext vorkommen.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
   "CustomKeyStores": [
```

```
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "CustomKeyStoreType": "AWS CloudHSM key store",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "CONNECTED"
   ],
}
```

Der nächste Beispielbefehl verwendet den DescribeClustersVorgang, um zu überprüfen, ob der AWS CloudHSM Cluster, der dem `ExampleKeyStore` (cluster-1a23b4cdefg) zugeordnet ist, mindestens zwei aktive Cluster hat. HSMs Wenn der Cluster weniger als zwei hat, schlägt der Vorgang fehl. HSMs `CreateKey`

```
$ aws cloudhsmv2 describe-clusters
{
    "Clusters": [
        {
            "SubnetMapping": {
               ...
            },
            "CreateTimestamp": 1507133412.351,
            "ClusterId": "cluster-1a23b4cdefg",
            "SecurityGroup": "sg-865af2fb",
            "HsmType": "hsm1.medium",
            "VpcId": "vpc-1a2b3c4d",
            "BackupPolicy": "DEFAULT",
            "Certificates": {
                "ClusterCertificate": "-----BEGIN CERTIFICATE-----\...\n-----END
 CERTIFICATE-----\n"
            },
            "Hsms": [
                {
                    "AvailabilityZone": "us-west-2a",
                    "EniIp": "10.0.1.11",
                    "ClusterId": "cluster-1a23b4cdefg",
                    "EniId": "eni-ea8647e1",
                    "StateMessage": "HSM created.",
                    "SubnetId": "subnet-a6b10bd1",
                    "HsmId": "hsm-abcdefghijk",
                    "State": "ACTIVE"
                },
```

```
                {
                    "AvailabilityZone": "us-west-2b",
                    "EniIp": "10.0.0.2",
                    "ClusterId": "cluster-1a23b4cdefg",
                    "EniId": "eni-ea8647e1",
                    "StateMessage": "HSM created.",
                    "SubnetId": "subnet-b6b10bd2",
                    "HsmId": "hsm-zyxwvutsrqp",
                    "State": "ACTIVE"
                },
            ],
            "State": "ACTIVE"
        }
    ]
}
```

In diesem Beispielbefehl wird der CreateKeyVorgang verwendet, um einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher zu erstellen. Um einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher zu erstellen, müssen Sie die benutzerdefinierte Schlüsselspeicher-ID des AWS CloudHSM Schlüsselspeichers und den `Origin` Wert angeben`AWS_CLOUDHSM`.

Die Antwort umfasst die IDs des benutzerdefinierten Schlüsselspeichers und des AWS CloudHSM Clusters.

Wenn Sie diesen Befehl ausführen, denken Sie daran, die ID des benutzerdefinierten Schlüsselspeichers in dem Beispiel durch eine gültige ID zu ersetzen.

```
$ aws kms create-key --origin AWS_CLOUDHSM --custom-key-store-id cks-1234567890abcdef0
{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": 1.499288695918E9,
    "Description": "Example key",
    "Enabled": true,
    "MultiRegion": false,
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "Origin": "AWS_CLOUDHSM"
    "CloudHsmClusterId": "cluster-1a23b4cdefg",
```

```
    "CustomKeyStoreId": "cks-1234567890abcdef0"
    "KeySpec": "SYMMETRIC_DEFAULT",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
    ]
  }
}
```

Erstellen Sie einen KMS-Schlüssel in externen Schlüsselspeichern

Nachdem Sie Ihren externen Schlüsselspeicher erstellt und verbunden haben, können Sie ihn AWS KMS keys in Ihrem Schlüsselspeicher erstellen. Dabei muss es sich um KMS-Schlüssel mit symmetrischer Verschlüsselung mit dem Ursprungswert External key store (Externer Schlüsselspeicher) (`EXTERNAL_KEY_STORE`) handeln. Sie können asymmetrische KMS-Schlüssel, HMAC-KMS-Schlüssel oder KMS-Schlüssel mit importiertem Schlüsselmaterial nicht in einem benutzerdefinierten Schlüsselspeicher erstellen. Außerdem können Sie auch keine KMS-Schlüssel mit symmetrischer Verschlüsselung in einem benutzerdefinierten Schlüsselspeicher verwenden, um asymmetrische Datenschlüsselpaare zu generieren.

Ein KMS-Schlüssel in einem externen Schlüsselspeicher hat möglicherweise eine schlechtere Latenz, Haltbarkeit und Verfügbarkeit als ein Standard-KMS-Schlüssel, da er von Komponenten abhängt, die sich außerhalb von AWS befinden. Bevor Sie einen KMS-Schlüssel in einem externen Schlüsselspeicher erstellen oder verwenden, prüfen Sie, ob Sie einen Schlüssel mit den Eigenschaften eines externen Schlüsselspeichers benötigen.

> **Note**
>
> Einige externe Schlüsselmanager bieten eine einfachere Methode zum Erstellen von KMS-Schlüsseln in einem externen Schlüsselspeicher. Weitere Informationen finden Sie in der Dokumentation zum externen Schlüsselmanager.

Zum Erstellen eines KMS-Schlüssels in Ihrem externen Schlüsselspeicher müssen Sie Folgendes angeben:

- Die ID Ihres externen Schlüsselspeichers.
- Externer Schlüsselspeicher (`EXTERNAL_KEY_STORE`) als Herkunft des Schlüsselmaterials.

- Die ID eines vorhandenen externen Schlüssels im externen Schlüsselmanager, der Ihrem externen Schlüsselspeicher zugeordnet ist. Dieser externe Schlüssel dient als Schlüsselmaterial für den KMS-Schlüssel. Sie können die ID des externen Schlüssels nicht mehr ändern, nachdem Sie den KMS-Schlüssel erstellt haben.

  AWS KMS stellt Ihrem externen Schlüsselspeicher-Proxy bei Anfragen für Verschlüsselungs- und Entschlüsselungsvorgänge die externe Schlüssel-ID zur Verfügung. AWS KMS kann nicht direkt auf Ihren externen Schlüsselmanager oder einen seiner kryptografischen Schlüssel zugreifen.

Neben dem externen Schlüssel enthält ein KMS-Schlüssel in einem externen Schlüsselspeicher auch AWS KMS Schlüsselmaterial. Alle mit dem KMS-Schlüssel verschlüsselten Daten werden zuerst AWS KMS mit dem Schlüsselmaterial des AWS KMS Schlüssels und dann von Ihrem externen Schlüsselmanager mit Ihrem externen Schlüssel verschlüsselt. Dieser Prozess der doppelten Verschlüsselung sorgt dafür, dass der durch einen KMS-Schlüssel in einem externen Schlüsselspeicher geschützte Geheimtext mindestens so stark ist wie nur durch AWS KMS geschützter Geheimtext. Details hierzu finden Sie unter Funktionsweise externer Schlüsselspeicher.

Wenn der `CreateKey`-Vorgang erfolgreich ist, lautet der Schlüsselstatus des neuen KMS-Schlüssels `Enabled`. Wenn Sie einen KMS-Schlüssel in einem externen Schlüsselspeicher anzeigen, können Sie typische Eigenschaften wie die Schlüssel-ID, die Schlüsselspezifikation, die Schlüsselnutzung, den Schlüsselstatus und das Erstellungsdatum sehen. Sie können aber auch die ID und den Verbindungsstatus des externen Schlüsselspeichers sowie die ID des externen Schlüssels sehen.

Wenn Ihr Versuch, einen KMS-Schlüssel in Ihrem externen Schlüsselspeicher zu erstellen, fehlschlägt, ermitteln Sie die Ursache anhand der Fehlermeldung. Sie kann darauf hinweisen, dass der externe Schlüsselspeicher nicht verbunden ist (`CustomKeyStoreInvalidStateException`), dass der Proxy Ihres externen Schlüsselspeichers keinen externen Schlüssel mit der angegebenen externen Schlüssel-ID finden kann (`XksKeyNotFoundException`) oder dass der externe Schlüssel bereits mit einem KMS-Schlüssel im selben externen Schlüsselspeicher verknüpft ist (`XksKeyAlreadyInUseException`).

Ein Beispiel für das AWS CloudTrail Protokoll des Vorgangs, bei dem ein KMS-Schlüssel in einem externen Schlüsselspeicher erstellt wird, finden Sie unterCreateKey.

Themen

- Anforderungen an einen KMS-Schlüssel in einem externen Schlüsselspeicher
- Erstellen Sie einen neuen KMS-Schlüssel in Ihrem externen Schlüsselspeicher

Anforderungen an einen KMS-Schlüssel in einem externen Schlüsselspeicher

Zum Erstellen eines KMS-Schlüssels in einem externen Schlüsselspeicher sind die folgenden Eigenschaften des externen Schlüsselspeichers, des KMS-Schlüssels und des externen Schlüssels, der als externes kryptografisches Schlüsselmaterial für den KMS-Schlüssel dient, erforderlich.

Anforderungen an einen externen Schlüsselspeicher

- Muss mit seinem externen Schlüsselspeicher-Proxy verbunden sein.

  Informationen zum Verbindungsstatus Ihres externen Schlüsselspeichers finden Sie unter Externe Schlüsselspeicher anzeigen. Informationen zum Verbinden Ihres externen Schlüsselspeichers finden Sie unter Externe Schlüsselspeicher Connect und trennen.

Anforderungen an KMS-Schlüssel

Diese Eigenschaften können nach dem Erstellen des KMS-Schlüssels nicht mehr geändert werden.

- Schlüsselspezifikation: SYMMMETRIC_DEFAULT
- Schlüsselnutzung: ENCRYPT_DECRYPT
- Schlüsselmaterialursprung: EXTERNAL_KEY_STORE
- Multi-Region: FALSE

Anforderungen an externe Schlüssel

- Kryptografischer 256-Bit-AES-Schlüssel (256 zufällige Bits). Die `KeySpec` des externen Schlüssels muss `AES_256` sein.
- Aktiviert und zur Verwendung verfügbar. Die `Status` des externen Schlüssels muss `ENABLED` sein.
- Konfiguriert für Verschlüsselung und Entschlüsselung. Die `KeyUsage` des externen Schlüssels muss `ENCRYPT` und `DECRYPT` enthalten.
- Wird nur mit diesem KMS-Schlüssel verwendet. Jeder `KMS key` in einem externen Schlüsselspeicher muss mit einem anderen externen Schlüssel verbunden sein.

AWS KMS empfiehlt außerdem, den externen Schlüssel ausschließlich für den externen Schlüsselspeicher zu verwenden. Diese Einschränkung macht es einfacher, Probleme mit dem Schlüssel zu erkennen und zu beheben.

- Zugriff durch den Proxy des externen Schlüsselspeichers für den externen Schlüsselspeicher möglich.

  Wenn der Proxy des externen Schlüsselspeichers den Schlüssel mit der angegebenen externen Schlüssel-ID nicht finden kann, schlägt der Vorgang `CreateKey` fehl.
- Kann den zu erwartenden Datenverkehr verarbeiten, den Ihre Verwendung von AWS-Services generiert. AWS KMS empfiehlt, externe Schlüssel für die Verarbeitung von bis zu 1800 Anfragen pro Sekunde vorzubereiten.

Erstellen Sie einen neuen KMS-Schlüssel in Ihrem externen Schlüsselspeicher

Sie können einen neuen KMS-Schlüssel in Ihrem externen Schlüsselspeicher in der AWS KMS Konsole oder mithilfe des CreateKeyVorgangs erstellen.

Mithilfe der AWS KMS Konsole

Es gibt zwei Möglichkeiten, einen KMS-Schlüssel in einem externen Schlüsselspeicher zu erstellen.

- Methode 1 (empfohlen): Wählen Sie einen externen Schlüsselspeicher und erstellen Sie dann einen KMS-Schlüssel in diesem externen Schlüsselspeicher.
- Methode 2: Erstellen Sie einen KMS-Schlüssel und geben Sie dann an, dass er sich in einem externen Schlüsselspeicher befindet.

Wenn Sie Methode 1 verwenden, bei der Sie Ihren externen Schlüsselspeicher auswählen, bevor Sie Ihren Schlüssel erstellen, AWS KMS wählt er alle erforderlichen KMS-Schlüsseleigenschaften für Sie aus und gibt die ID Ihres externen Schlüsselspeichers ein. Diese Methode vermeidet Fehler, die Ihnen bei der Erstellung Ihres KMS-Schlüssels unterlaufen könnten.

> **Note**
>
> Nehmen Sie keine vertraulichen oder sensiblen Informationen in den Alias, in der Beschreibung oder in den Tags auf. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

Methode 1 (empfohlen): Starten in Ihrem externen Schlüsselspeicher

Um diese Methode zu verwenden, wählen Sie Ihren externen Schlüsselspeicher und erstellen dann einen KMS-Schlüssel. Die AWS KMS Konsole wählt alle erforderlichen Eigenschaften für Sie aus und gibt die ID Ihres externen Schlüsselspeichers ein. Diese Methode vermeidet viele Fehler, die Ihnen bei der Erstellung Ihres KMS-Schlüssels unterlaufen könnten.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (Benutzerdefinierte Schlüsselspeicher), External key stores (Externe Schlüsselspeicher) aus.
4. Wählen Sie den Namen Ihres externen Schlüsselspeichers aus.
5. Wählen Sie oben rechts die Option Create a KMS key in this key store (Erstellen eines KMS-Schlüssels in diesem Schlüsselspeicher) aus.

   Wenn der externe Schlüsselspeicher nicht verbunden ist, werden Sie aufgefordert, ihn zu verbinden. Wenn der Verbindungsversuch fehlschlägt, müssen Sie das Problem lösen und den externen Schlüsselspeicher verbinden, bevor Sie darin einen neuen KMS-Schlüssel erstellen können.

   Wenn der externe Schlüsselspeicher verbunden ist, werden Sie zur Seite Customer managed keys (Kundenverwaltete Schlüssel) weitergeleitet, auf der Sie einen Schlüssel erstellen können. Die erforderlichen Schlüsselkonfigurationswerte wurden bereits für Sie ausgewählt. Außerdem ist die benutzerdefinierte Schlüsselspeicher-ID Ihres externen Schlüsselspeichers ausgefüllt, Sie können diese aber ändern.
6. Geben Sie die Schlüssel-ID eines externen Schlüssels in Ihren externen Schlüsselmanager ein. Dieser externe Schlüssel muss die Anforderungen für die Verwendung mit einem KMS-Schlüssel erfüllen. Sie können diesen Wert nach der Erstellung des KMS-Schlüssels nicht mehr ändern.

Wenn der externe Schlüssel mehrere hat IDs, geben Sie die Schlüssel-ID ein, die der externe Schlüsselspeicher-Proxy zur Identifizierung des externen Schlüssels verwendet.

7. Bestätigen Sie, dass Sie beabsichtigen, einen KMS-Schlüssel im angegebenen externen Schlüsselspeicher zu erstellen.
8. Wählen Sie Weiter aus.

   Der Rest dieses Verfahrens entspricht dem Erstellen eines Standard-KMS-Schlüssels.
9. Geben Sie einen Alias (erforderlich) und eine Beschreibung (optional) für den KMS-Schlüssel ein.
10. (Optional). Fügen Sie auf der Seite Add Tags (Tags hinzufügen) Tags hinzu, um Ihre KMS-Schlüssel identifizieren zu können und sie zu kategorisieren.

    Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenverteilungsbericht generiert, in dem die Nutzung und die Kosten nach Stichwörtern zusammengefasst sind. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Schlagworte in AWS KMS und ABAC für AWS KMS.
11. Wählen Sie Weiter aus.
12. Sie können im Abschnitt Key administrators (Schlüsseladministratoren) die IAM-Benutzer und -Rollen auswählen, die den KMS-Schlüssel verwalten dürfen. Weitere Informationen finden Sie unter Erlaubt Schlüsseladministratoren die Verwaltung des KMS-Schlüssels.

    > **Note**
    >
    > Daneben können IAM-Benutzer und -Rollen die erforderlichen Berechtigungen zur Verwendung des KMS-Schlüssel auch über IAM-Richtlinien erhalten.
    > Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
13. (Optional) Um zu verhindern, dass diese Schlüsseladministratoren diesen KMS-Schlüssel löschen, deaktivieren Sie das Kontrollkästchen Allow key administrators to delete this key (Ermöglicht Schlüsseladministratoren das Löschen dieses Schlüssels).

Das Löschen eines KMS-Schlüssels ist ein endgültiger und irreversibler Vorgang, der dazu führen kann, dass Geheimtext nicht wiederhergestellt werden kann. Sie können einen symmetrischen KMS-Schlüssel in einem externen Schlüsselspeicher nicht wiederherstellen, selbst wenn Sie über das externe Schlüsselmaterial verfügen. Das Löschen eines KMS-Schlüssels hat jedoch keine Auswirkungen auf den zugehörigen externen Schlüssel. Informationen zum Löschen eines KMS-Schlüssels aus einem externen Schlüsselspeicher finden Sie unter Besondere Überlegungen zum Löschen von Schlüsseln.

14. Wählen Sie Weiter aus.
15. Wählen Sie im Abschnitt Dieses Konto die IAM-Benutzer und Rollen aus, AWS-Konto die den KMS-Schlüssel für kryptografische Operationen verwenden können. Weitere Informationen finden Sie unter Erlaubt Schlüsselbenutzern die Verwendung des KMS-Schlüssels.

> **Note**
>
> Daneben können IAM-Benutzer und -Rollen die erforderlichen Berechtigungen zur Verwendung des KMS-Schlüssels auch über IAM-Richtlinien erhalten.
> Bewährte IAM-Methoden raten von der Verwendung von IAM-Benutzern mit langfristigen Anmeldeinformationen ab. Verwenden Sie nach Möglichkeit IAM-Rollen, die temporäre Anmeldeinformationen bereitstellen. Weitere Informationen finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

16. (Optional) Sie können anderen erlauben, diesen KMS-Schlüssel für kryptografische Operationen AWS-Konten zu verwenden. Wählen Sie dazu unten auf der Seite im AWS-Konten Abschnitt Andere die Option Weiteres hinzufügen aus AWS-Konto und geben Sie die AWS-Konto ID eines externen Kontos ein. Wiederholen Sie diesen Schritt, um weitere externe Konten hinzufügen.

> **Note**
>
> Administratoren der anderen AWS-Konten müssen ebenfalls Zugriff auf den KMS-Schlüssel gewähren, indem sie IAM-Richtlinien für ihre Benutzer erstellen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

17. Wählen Sie Weiter aus.
18. Überprüfen Sie die gewählten Einstellungen. Sie können immer noch zurückgehen und alle Einstellungen ändern.

19. Wählen Sie danach Finish (Fertigstellen) aus.

Methode 2: Starten in kundenverwalteten Schlüsseln

Dieses Verfahren entspricht dem Verfahren zur Erstellung eines symmetrischen Verschlüsselungsschlüssels mit AWS KMS Schlüsselmaterial. In diesem Verfahren geben Sie jedoch die benutzerdefinierte Schlüsselspeicher-ID des externen Schlüsselspeichers und die Schlüssel-ID des externen Schlüssels an. Sie müssen auch die erforderlichen Eigenschaftswerte für einen KMS-Schlüssel in einem externen Schlüsselspeicher angeben, z. B. die Schlüsselspezifikation und die Schlüsselnutzung.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Klicken Sie auf Create key.
5. Wählen Sie Symmetric (Symmetrisch).
6. Unter Key usage (Schlüsselverwendung) ist die Option Encrypt and decrypt (Verschlüsseln und Entschlüsseln) für Sie ausgewählt. Ändern Sie dies nicht.
7. Wählen Sie Advanced options (Erweiterte Optionen) aus.
8. Wählen Sie unter Key material origin (Schlüsselmaterialursprung) die Option External key store (Externer Schlüsselspeicher) aus.
9. Bestätigen Sie, dass Sie beabsichtigen, einen KMS-Schlüssel im angegebenen externen Schlüsselspeicher zu erstellen.
10. Wählen Sie Weiter aus.
11. Wählen Sie die Zeile, die den externen Schlüsselspeicher für den neuen KMS-Schlüssel darstellt.

    Sie können keinen getrennten externen Schlüsselspeicher auswählen. Zum Verbinden eines nicht verbundenen Schlüsselspeichers wählen Sie den Namen des Schlüsselspeichers und dann unter Key store actions (Schlüsselspeicheraktionen) die Option Connect (Verbinden) aus. Details hierzu finden Sie unter Mithilfe der Konsole AWS KMS.

12. Geben Sie die Schlüssel-ID eines externen Schlüssels in Ihren externen Schlüsselmanager ein. Dieser externe Schlüssel muss die Anforderungen für die Verwendung mit einem KMS-Schlüssel erfüllen. Sie können diesen Wert nach der Erstellung des KMS-Schlüssels nicht mehr ändern.

    Wenn der externe Schlüssel mehrere hat IDs, geben Sie die Schlüssel-ID ein, die der externe Schlüsselspeicher-Proxy zur Identifizierung des externen Schlüssels verwendet.

13. Wählen Sie Weiter aus.

    Der Rest dieses Verfahrens entspricht dem Erstellen eines Standard-KMS-Schlüssels.

14. Geben Sie einen Alias und eine optionale Beschreibung für den KMS-Schlüssel ein.

15. (Optional). Fügen Sie auf der Seite Add Tags (Tags hinzufügen) Tags hinzu, um Ihre KMS-Schlüssel identifizieren zu können und sie zu kategorisieren.

    Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenverteilungsbericht generiert, in dem die Nutzung und die Kosten nach Stichwörtern zusammengefasst sind. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Schlagworte in AWS KMS und ABAC für AWS KMS.

16. Wählen Sie Weiter aus.

17. Sie können im Abschnitt Key administrators (Schlüsseladministratoren) die IAM-Benutzer und -Rollen auswählen, die den KMS-Schlüssel verwalten dürfen. Weitere Informationen finden Sie unter Erlaubt Schlüsseladministratoren die Verwaltung des KMS-Schlüssels.

    > **Note**
    >
    > Daneben können IAM-Benutzer und -Rollen die erforderlichen Berechtigungen zur Verwendung des KMS-Schlüssel auch über IAM-Richtlinien erhalten.

18. (Optional) Um zu verhindern, dass diese Schlüsseladministratoren diesen KMS-Schlüssel löschen, deaktivieren Sie das Kontrollkästchen Allow key administrators to delete this key (Ermöglicht Schlüsseladministratoren das Löschen dieses Schlüssels).

    Das Löschen eines KMS-Schlüssels ist ein endgültiger und irreversibler Vorgang, der dazu führen kann, dass Geheimtext nicht wiederhergestellt werden kann. Sie können einen symmetrischen KMS-Schlüssel in einem externen Schlüsselspeicher nicht wiederherstellen, selbst wenn Sie über das externe Schlüsselmaterial verfügen. Das Löschen eines KMS-Schlüssels hat jedoch keine Auswirkungen auf den zugehörigen externen Schlüssel.

Informationen zum Löschen eines KMS-Schlüssels aus einem externen Schlüsselspeicher finden Sie unter Lösche eine AWS KMS key.

19. Wählen Sie Weiter aus.
20. Wählen Sie im Abschnitt Dieses Konto die IAM-Benutzer und Rollen aus, AWS-Konto die den KMS-Schlüssel für kryptografische Operationen verwenden können. Weitere Informationen finden Sie unter Erlaubt Schlüsselbenutzern die Verwendung des KMS-Schlüssels.

> **Note**
>
> Daneben können IAM-Benutzer und -Rollen die erforderlichen Berechtigungen zur Verwendung des KMS-Schlüssels auch über IAM-Richtlinien erhalten.

21. (Optional) Sie können anderen erlauben, diesen KMS-Schlüssel für kryptografische Operationen AWS-Konten zu verwenden. Wählen Sie dazu unten auf der Seite im AWS-Konten Abschnitt Andere die Option Weiteres hinzufügen aus AWS-Konto und geben Sie die AWS-Konto ID eines externen Kontos ein. Wiederholen Sie diesen Schritt, um weitere externe Konten hinzufügen.

> **Note**
>
> Administratoren der anderen AWS-Konten müssen ebenfalls Zugriff auf den KMS-Schlüssel gewähren, indem sie IAM-Richtlinien für ihre Benutzer erstellen. Weitere Informationen finden Sie unter Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben.

22. Wählen Sie Weiter aus.
23. Überprüfen Sie die gewählten Einstellungen. Sie können immer noch zurückgehen und alle Einstellungen ändern.
24. Wählen Sie danach Finish (Fertigstellen) aus.

Wenn der Vorgang erfolgreich abgeschlossen wird, wird der neue KMS-Schlüssel in dem ausgewählten externen Schlüsselspeicher angezeigt. Wenn Sie den Namen oder Alias des neuen KMS-Schlüssels auswählen, werden auf der Registerkarte Cryptographic configuration (Kryptografische Konfiguration) auf der Detailseite der Ursprung des KMS-Schlüssels (Externer Schlüsselspeicher), der Name, die ID und der Typ des benutzerdefinierten Schlüsselspeichers sowie die ID, die Schlüsselnutzung und der Status des externen Schlüssels angezeigt. Wenn der

Vorgang fehlschlägt, wird unter eine Fehlermeldung mit einer Beschreibung des Fehlers angezeigt. Informationen finden Sie unter Fehlerbehebung bei externen Schlüsselspeichern.

> **Tip**
>
> Um die Identifizierung von KMS-Schlüsseln in einem benutzerdefinierten Schlüsselspeicher zu erleichtern, fügen Sie auf der Seite Customer managed keys (Kundenverwaltete Schlüssel) die Spalten Origin (Urspring) und Custom key store ID (ID des benutzerdefinierten Schlüsselspeichers) zur Anzeige hinzu. Zum Ändern der Tabellenfelder wählen Sie das Zahnradsymbol rechts oben auf der Seite aus. Details hierzu finden Sie unter Passen Sie Ihre Konsolenansicht an.

Verwenden der API AWS KMS

Verwenden Sie den CreateKeyVorgang, um einen neuen KMS-Schlüssel in einem externen Schlüsselspeicher zu erstellen. Die folgenden Parameter sind erforderlich:

- Der `Origin`-Wert muss `EXTERNAL_KEY_STORE` lauten.
- Der `CustomKeyStoreId`-Parameter identifiziert Ihren externen Schlüsselspeicher. Der `ConnectionState` des angegebenen externen Schlüsselspeichers muss `CONNECTED` sein. Verwenden Sie den `DescribeCustomKeyStores`-Vorgang um die `CustomKeyStoreId` und den `ConnectionState` zu ermitteln.
- Der `XksKeyId`-Parameter identifiziert den externen Schlüssel. Dieser externe Schlüssel muss die Anforderungen für die Zuordnung zu einem KMS-Schlüssel erfüllen.

Sie können auch jeden der optionalen Parameter des `CreateKey`-Vorgangs verwenden, z. B. die `Policy`- oder Tags-Parameter.

> **Note**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in die Felder `Description` oder `Tags` ein. Diese Felder können in CloudTrail Protokollen und anderen Ausgaben als Klartext erscheinen.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

In diesem Beispielbefehl wird der CreateKeyVorgang verwendet, um einen KMS-Schlüssel in einem externen Schlüsselspeicher zu erstellen. Die Antwort umfasst die Eigenschaften der KMS-Schlüssel, die ID des externen Schlüsselspeichers sowie ID, Nutzung und Status des externen Schlüssels.

Wenn Sie diesen Befehl ausführen, denken Sie daran, die ID des benutzerdefinierten Schlüsselspeichers in dem Beispiel durch eine gültige ID zu ersetzen.

```
$ aws kms create-key --origin EXTERNAL_KEY_STORE --custom-key-store-
id cks-1234567890abcdef0 --xks-key-id bb8562717f809024
{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2022-12-02T07:48:55-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "CustomKeyStoreId": "cks-1234567890abcdef0",
    "Description": "",
    "Enabled": true,
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "EXTERNAL_KEY_STORE",
    "XksKeyConfiguration": {
      "Id": "bb8562717f809024"
    }
  }
}
```

Schlüssel identifizieren und anzeigen

Sie können die API AWS Management Consoleoder AWS Key Management Service (AWS KMS) verwenden, um Informationen für jedes Konto und jede Region anzuzeigen AWS KMS keys , einschließlich KMS-Schlüssel, die Sie verwalten, und KMS-Schlüssel, die von verwaltet werden AWS.

Themen

- Finden Sie die Schlüssel-ID und den Schlüssel-ARN
- KMS-Schlüsseldetails aufrufen und auflisten
- Identifizieren Sie verschiedene Schlüsseltypen
- Passen Sie Ihre Konsolenansicht an
- Suchen Sie KMS-Schlüssel und Schlüsselmaterial in einem AWS CloudHSM Schlüsselspeicher

Finden Sie die Schlüssel-ID und den Schlüssel-ARN

Um einen zu identifizieren AWS KMS key, können Sie die Schlüssel-ID oder den Amazon-Ressourcennamen (Schlüssel-ARN) verwenden. Bei kryptografischen Produktionen können Sie auch den Aliasnamen oder den Alias-ARNverwenden.

Sie können die AWS KMS Konsole oder den ListKeysVorgang verwenden, um die Schlüssel-ID und den Schlüssel-ARN jedes KMS-Schlüssels in Ihrem Konto und Ihrer Region zu identifizieren.

Ausführliche Informationen zu den KMS-Schlüsselkennungen, die von unterstützt werden AWS KMS, finden Sie unterSchlüsselkennungen () KeyId. Weitere Informationen zur Suche nach Aliasnamen und Alias-ARN finden Sie unter Suchen Sie den Aliasnamen und den Alias-ARN für einen KMS-Schlüssel.

Verwenden der Konsole AWS KMS

1. Öffnen Sie die AWS KMS Konsole unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Zum Anzeigen der Schlüssel in Ihrem Konto, die Sie erstellen und verwalten, wählen Sie im Navigationsbereich Customer managed keys (Vom Kunden verwaltete Schlüssel) aus. Um die Schlüssel in Ihrem Konto anzuzeigen, das für Sie AWS erstellt und verwaltet wird, wählen AWS Sie im Navigationsbereich die Option Verwaltete Schlüssel aus.

4. Um die Schlüssel-ID für einen KMS-Schlüssel zu finden, suchen Sie nach der Zeile, die mit dem KMS-Schlüssel-Alias beginnt.

   Die Spalte Key ID (Schlüssel-ID) wird standardmäßig in den Tabellen angezeigt. Wenn die Schlüssel-ID-Spalte nicht in der Tabelle angezeigt wird, verwenden Sie das unter the section called “Passen Sie Ihre Konsolenansicht an” beschriebene Verfahren, um sie wiederherzustellen. Sie können die Schlüssel-ID eines KMS-Schlüssels auch auf der Detailseite anzeigen.

5. Um den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels zu suchen, wählen Sie die Schlüssel-ID oder den Alias aus. Der Schlüssel-ARN wird im Bereich General Configuration (Allgemeine Konfiguration) angezeigt.

Verwenden der AWS KMS API

Verwenden Sie die ListKeysOperation, um die Schlüssel-ID und den Schlüssel-ARN eines AWS KMS key zu ermitteln.

Der `ListKeys` Vorgang gibt die Schlüssel-ID und den Amazon-Ressourcennamen (ARN) aller KMS-Schlüssel im Konto und in der Region des Anrufers zurück.

Dieser Aufruf der `ListKeys`-Produktion gibt beispielsweise die ID und den ARN jedes KMS-Schlüssels in diesem fiktiven Konto zurück. Beispiele in verschiedenen Programmiersprachen finden Sie unter Verwendung `ListKeys` mit einem AWS SDK oder CLI.

```
$ aws kms list-keys
{
    "Keys": [
        {
            "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "KeyArn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ]
}
```

KMS-Schlüsseldetails aufrufen und auflisten

Sie können die AWS KMS Konsole oder den DescribeKeyVorgang verwenden, um auf detaillierte Informationen zu den KMS-Schlüsseln im Konto und in der Region zuzugreifen und diese aufzulisten.

Die folgenden Verfahren zeigen, wie Sie auf KMS-Schlüsseldetails wie Schlüssel-ID, Schlüsselspezifikation, Schlüsselverwendung und mehr zugreifen können.

Verwenden der Konsole AWS KMS

Die Detailseite für jeden KMS-Schlüssel zeigt die Eigenschaften des KMS-Schlüssels an. Sie unterscheidet sich geringfügig für die verschiedenen Arten von KMS-Schlüsseln.

Um detaillierte Informationen zu einem KMS-Schlüssel anzuzeigen, wählen Sie auf der Seite Von AWS verwaltete Schlüssel oder Vom Kunden verwaltete Schlüssel den Alias oder die Schlüssel-ID des KMS-Schlüssels aus.

Die Detailseite eines KMS-Schlüssels enthält den Abschnitt General Configuration (allgemeine Konfiguration), der die grundlegenden Eigenschaften des KMS-Schlüssels anzeigt. Sie enthält auch Registerkarten, auf denen Sie Eigenschaften des KMS-Schlüssels anzeigen und bearbeiten können, z. B. Schlüsselrichtlinie, kryptografische Konfiguration, Tags, Schlüsselmaterial und Rotationen (für KMS-Schlüssel, die automatische oder On-Demand-Rotation unterstützen), Regionalität (für Schlüssel mit mehreren Regionen) und Öffentlicher Schlüssel (für asymmetrische KMS-Schlüssel).

> **Note**
>
> In der AWS KMS Konsole werden die KMS-Schlüssel angezeigt, zu deren Anzeige Sie in Ihrem Konto und Ihrer Region berechtigt sind. KMS-Schlüssel unter „Andere“ werden AWS-Konten nicht in der Konsole angezeigt, auch wenn Sie berechtigt sind, sie anzuzeigen, zu verwalten und zu verwenden. Verwenden Sie den DescribeKeyVorgang, um KMS-Schlüssel in anderen Konten anzuzeigen.

So navigieren Sie zur Detailseite für einen KMS-Schlüssel.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.

3. Zum Anzeigen der Schlüssel in Ihrem Konto, die Sie erstellen und verwalten, wählen Sie im Navigationsbereich Customer managed keys (Vom Kunden verwaltete Schlüssel) aus. Um die Schlüssel in Ihrem Konto anzuzeigen, das für Sie AWS erstellt und verwaltet wird, wählen AWS Sie im Navigationsbereich die Option Verwaltete Schlüssel aus.
4. Um die Seite mit den Schlüsseldetails zu öffnen, wählen Sie in der Schlüsseltabelle die Schlüssel-ID oder den Alias des KMS-Schlüssels aus.

   Wenn der KMS-Schlüssel mehrere Aliase enthält, wird eine Aliasübersicht (+n mehr) neben dem Namen des Aliases angezeigt. Wenn Sie die Aliasübersicht auswählen, gelangen Sie direkt zur Aliases (Aliase)-Registerkarte auf der Seite mit den Schlüsseldetails.

In der folgenden Liste werden die Felder in der Detailanzeige beschrieben, einschließlich der Felder in den Registerkarten. Einige dieser Felder sind auch als Spalten in der Tabellenanzeige verfügbar.

Aliasnamen

Wo: Registerkarte für Aliase

Ein Anzeigename für den KMS-Schlüssel Sie können einen Alias verwenden, um den KMS-Schlüssel in der Konsole und in einigen Konsolen zu identifizieren AWS KMS APIs. Details hierzu finden Sie unter Aliase in AWS KMS.

Auf der Registerkarte Aliase werden alle Aliase angezeigt, die dem KMS-Schlüssel in der Region AWS-Konto und zugeordnet sind.

ARN

Wo: Abschnitt über allgemeine Konfiguration

Der Amazon-Ressourcenname (ARN) eines KMS-Schlüssels. Dieser Wert identifiziert den KMS-Schlüssel eindeutig. Sie können damit den KMS-Schlüssel in AWS KMS -API-Operationen identifizieren.

Verbindungsstatus

Gibt an, ob ein benutzerdefinierter Schlüsselspeicher mit seinem Unterstützungsschlüsselspeicher verbunden ist. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher erstellt wird.

Informationen zu den Werten in diesem Feld finden Sie ConnectionStatein der AWS KMS API-Referenz.

Erstellungsdatum

Wo: Abschnitt über allgemeine Konfiguration

Datum und Uhrzeit, wann der KMS-Schlüssel erstellt wurde. Dieser Wert wird in Ortszeit für das Gerät angezeigt. Die Zeitzone hängt nicht von der Region ab.

Anders als bei Ablauf bezieht sich die Erstellung nur auf den KMS-Schlüssel, nicht auf sein Schlüsselmaterial.

CloudHSM-Cluster-ID

Wo: Registerkarte für kryptografische Konfiguration

Die Cluster-ID des AWS CloudHSM Clusters, der das Schlüsselmaterial für den KMS-Schlüssel enthält. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher erstellt wird.

Wenn Sie die CloudHSM-Cluster-ID wählen, wird die Cluster-Seite in der AWS CloudHSM Konsole geöffnet.

Aktuelles Schlüsselmaterial

Wo: Abschnitt über allgemeine Konfiguration

Symmetrische Verschlüsselungsschlüssel mit `AWS_KMS` Ursprung unterstützen sowohl die automatische Rotation als auch die Rotation nach Bedarf. Symmetrische Verschlüsselungsschlüssel mit `EXTERNAL` Herkunft für einzelne Regionen unterstützen die Rotation nach Bedarf. Diesen Schlüsseln können mehrere Schlüsselmaterialien zugeordnet sein. Das zuletzt rotierte Schlüsselmaterial kann sowohl für die Verschlüsselung als auch für die Entschlüsselung verwendet werden. Dieses Schlüsselmaterial wird als das aktuelle Schlüsselmaterial identifiziert. Andere Schlüsselmaterialien können nur zur Entschlüsselung verwendet werden. Durch die automatische oder bedarfsgesteuerte Schlüsselrotation eines KMS-Schlüssels wird das aktuelle Schlüsselmaterial geändert.

ID des benutzerdefinierten Schlüsselspeichers

Wo: Registerkarte für kryptografische Konfiguration

Die ID des benutzerdefinierten Schlüsselspeichers, der den KMS-Schlüssel enthält. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher erstellt wird.

Wenn Sie die benutzerdefinierte Schlüsselspeicher-ID wählen, wird die Seite Benutzerdefinierte Schlüsselspeicher in der AWS KMS Konsole geöffnet.

Name des benutzerdefinierten Schlüsselspeichers

Wo: Registerkarte für kryptografische Konfiguration

Der Name des benutzerdefinierten Schlüsselspeichers, der den KMS-Schlüssel enthält. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher erstellt wird.

Typ des benutzerdefinierten Schlüsselspeichers

Wo: Registerkarte für kryptografische Konfiguration

Gibt an, ob der benutzerdefinierte Schlüsselspeicher ein AWS CloudHSM Schlüsselspeicher oder ein externer Schlüsselspeicher ist. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher erstellt wird.

Beschreibung

Wo: Abschnitt über allgemeine Konfiguration

Eine kurze, optionale Beschreibung des KMS-Schlüssels, die Sie schreiben und bearbeiten können. Um die Beschreibung eines kundenverwalteten Schlüssels hinzuzufügen oder zu aktualisieren, wählen Sie über General Configuration (allgemeine Konfiguration) Edit (Bearbeiten) aus.

Verschlüsselungsalgorithmen

Wo: Registerkarte für kryptografische Konfiguration

Listet die Verschlüsselungsalgorithmen auf, die mit dem KMS-Schlüssel in AWS KMS verwendet werden können. Dieses Feld wird nur angezeigt, wenn Key type (Schlüsseltyp) auf Asymmetric (Asymmetrisch) und Key usage (Schlüsselnutzung) auf Encrypt and decrypt (Verschlüsseln und Entschlüsseln) eingestellt ist. Informationen zu den AWS KMS unterstützten Verschlüsselungsalgorithmen finden Sie unter Schlüsselspezifikation SYMMMETRIC_DEFAULT undRSA-Schlüsselspezifikationen für Verschlüsselung und Entschlüsselung.

Ablaufdatum

Wo: Registerkarte für Schlüsselmaterial

Datum und Uhrzeit des Zeitpunkts, an dem das Schlüsselmaterial für den KMS-Schlüssel abläuft. Dieses Feld wird nur bei KMS-Schlüsseln mit importiertem Schlüsselmaterial angezeigt, das heißt, wenn der Ursprung extern ist und der KMS-Schlüssel Schlüsselmaterial enthält, das abläuft. Symmetrischen Verschlüsselungsschlüsseln mit einer Region können mehrere Schlüsselmaterialien zugeordnet sein. Für solche Schlüssel gibt dieses Feld das früheste Datum und die Uhrzeit an, zu der eines der zugehörigen Schlüsselmaterialien abläuft.

ID des externen Schlüssels

Wo: Registerkarte für kryptografische Konfiguration

Die ID des externen Schlüssels, der einem KMS-Schlüssel in einem externen Schlüsselspeicher zugeordnet ist. Dieses Feld wird nur für KMS-Schlüssel in einem externen Schlüsselspeicher angezeigt.

Status des externen Schlüssels

Wo: Registerkarte für kryptografische Konfiguration

Der letzte Status, den der externe Schlüsselspeicher-Proxy für den externen Schlüssel gemeldet hat, der dem KMS-Schlüssel zugeordnet ist. Dieses Feld wird nur für KMS-Schlüssel in einem externen Schlüsselspeicher angezeigt.

Nutzung des externen Schlüssels

Wo: Registerkarte für kryptografische Konfiguration

Die kryptografischen Vorgänge, die für den externen Schlüssel aktiviert sind, der dem KMS-Schlüssel zugeordnet ist. Dieses Feld wird nur für KMS-Schlüssel in einem externen Schlüsselspeicher angezeigt.

Schlüsselrichtlinie

Wo: Registerkarte für Schlüsselrichtlinie

Steuert den Zugriff auf den KMS-Schlüssel zusammen mit IAM-Richtlinien und Erteilungen. Jeder KMS-Schlüssel besitzt eine Schlüsselrichtlinie. Es ist das einzige obligatorische Berechtigungselement. Um die Schlüsselrichtlinie eines kundenverwalteten KMS-Schlüssels zu ändern, wählen Sie auf der Registerkarte Key policy (Schlüsselrichtlinie) die Option Edit (Bearbeiten). Details hierzu finden Sie unter the section called “Schlüsselrichtlinien”.

Schlüsselmaterial und Rotationen

Wo: Registerkarte „Schlüsselmaterial und Rotationen“

Diese Registerkarte wird nur für symmetrische Verschlüsselungsschlüssel mit `AWS_KMS` Ursprung (die sowohl automatische Rotation als auch On-Demand-Rotation unterstützen) sowie für symmetrische Verschlüsselungsschlüssel mit `EXTERNAL` Ursprung mit einer Region (die Rotation nach Bedarf unterstützen) angezeigt.

Die Registerkarte besteht aus drei Bereichen:

Automatische Rotation: Aktiviert und deaktiviert die automatische Rotation des Schlüsselmaterials in einem vom Kunden verwalteten KMS-Schlüssel. Verwenden Sie das Kontrollkästchen, um den Status der Schlüsselrotation eines vom Kunden verwalteten Schlüssels zu ändern. Sie können die Drehung von Schlüsselmaterial in einem Von AWS verwalteter Schlüssel nicht aktivieren oder deaktivieren. Von AWS verwaltete Schlüssel werden automatisch jedes Jahr gedreht.

Rotation nach Bedarf: Initiieren Sie eine On-Demand-Rotation des Schlüsselmaterials in einem vom Kunden verwalteten Schlüssel. Bei importierten Schlüsseln muss bereits ein importiertes Schlüsselmaterial im `PENDING_ROTATION` Status vorhanden sein, damit die Option „Jetzt drehen“ verfügbar ist.

Schlüsselmaterialien: Listet alle Schlüsselmaterialien auf, die dem KMS-Schlüssel zugeordnet sind. Jedes Schlüsselmaterial hat eine eindeutige Kennung, und in seiner Zeile werden zusätzliche Informationen zum Schlüsselmaterial angezeigt, z. B. das Rotationsdatum, an dem das Schlüsselmaterial für die Verwendung in KMS verfügbar wurde. Für importierte Schlüssel verfügt jede Zeile auch über ein Aktionsmenü, mit dem Sie ein bestimmtes Schlüsselmaterial löschen oder es erneut in den KMS-Schlüssel importieren können.

Schlüsselspezifikation

Wo: Registerkarte für kryptografische Konfiguration

Der Typ des Schlüsselmaterials im KMS-Schlüssel. AWS KMS unterstützt KMS-Schlüssel mit symmetrischer Verschlüsselung (SYMMETRIC_DEFAULT), HMAC-KMS-Schlüssel unterschiedlicher Länge, KMS-Schlüssel für RSA-Schlüssel unterschiedlicher Länge und elliptische Kurvenschlüssel mit unterschiedlichen Kurven. Details hierzu finden Sie unter Key spec.

Schlüsseltyp

Wo: Registerkarte für kryptografische Konfiguration

Gibt an, ob der KMS-Schlüssel symmetrisch oder asymmetrisch ist.

Schlüsselnutzung

Wo: Registerkarte für kryptografische Konfiguration

Gibt an, ob ein KMS-Schlüssel für Encrypt and decrypt (Verschlüsseln und Entschlüsseln), Sign and verify (Signieren und Überprüfen) und Generate and verify MAC (MAC generieren und überprüfen) verwendet werden kann. Details hierzu finden Sie unter Key usage.

Urspung

Wo: Registerkarte für kryptografische Konfiguration

Die Quelle des Schlüsselmaterials für den KMS-Schlüssel. Gültige Werte für sind:

- AWS KMS für Schlüsselmaterial, das AWS KMS generiert
- AWS CloudHSM für KMS-Schlüssel in einem AWS CloudHSM -Schlüsselspeicher
- Extern für importiertes Schlüsselmaterial (BYOK)
- Externer Schlüsselspeicher für KMS-Schlüssel in einem externen Schlüsselspeicher

MAC-Algorithmen

Wo: Registerkarte für kryptografische Konfiguration

Listet die MAC-Algorithmen auf, die mit einem HMAC-KMS-Schlüssel in AWS KMS verwendet werden können. Dieses Feld wird nur angezeigt, wenn Key spec (Schlüsselspezifikation) eine HMAC-Schlüsselspezifikation (HMAC_*) ist. Weitere Informationen zu den von AWS KMS unterstützten MAC-Algorithmen finden Sie unter Schlüsselspezifikationen für HMAC-KMS-Schlüssel.

Primärschlüssel

Wo: Registerkarte für Regionalität

Gibt an, dass dieser KMS-Schlüssel ein multiregionaler Primärschlüssel ist. Autorisierte Benutzer können diesen Abschnitt zum Ändern des Primärschlüssels in einen anderen verwandten multiregionalen Schlüssel verwenden. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel ein multiregionaler Primärschlüssel ist.

Öffentlicher Schlüssel

Wo: Registerkarte für öffentliche Schlüssel

Zeigt den öffentlichen Schlüssel eines asymmetrischen KMS-Schlüssels an. Autorisierte Benutzer können diese Registerkarte verwenden, um den öffentlichen Schlüssel zu kopieren und herunterzuladen.

Regionalität

Wo: Abschnitt über allgemeine Konfiguration und Registerkarten für Regionalität

Gibt an, ob ein KMS-Schlüssel ein einzelregionaler Schlüssel, ein multiregionaler Primärschlüssel oder ein multiregionaler Replikatschlüssel ist. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel ein multiregionaler Schlüssel ist.

Verwandte multiregionale Schlüssel

Wo: Registerkarte für Regionalität

Zeigt alle verwandte multiregionale Primär- und Replikatschlüssel an, mit Ausnahme des aktuellen KMS-Schlüssels. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel ein multiregionaler Schlüssel ist.

Im Abschnitt über verwandte multiregionale Schlüssel eines Primärschlüssels können autorisierte Benutzer neue Replikatschlüssel erstellen.

Replikat-Schlüssel

Wo: Registerkarte für Regionalität

Gibt an, dass dieser KMS-Schlüssel ein multiregionaler Replikatschlüssel ist. Dieses Feld wird nur angezeigt, wenn der KMS-Schlüssel ein multiregionaler Replikatschlüssel ist.

Signaturalgorithmen

Wo: Registerkarte für kryptografische Konfiguration

Listet die Signaturalgorithmen auf, die mit dem KMS-Schlüssel in AWS KMS verwendet werden können. Dieses Feld wird nur angezeigt, wenn Key type (Schlüsseltyp) auf Asymmetric (Asymmetrisch) und Key usage (Schlüsselnutzung) auf Sign and verify (Signieren und Überprüfen) eingestellt ist. Hinweise zu den unterstützten Signaturalgorithmen finden Sie unter und. AWS KMS RSA-Schlüsselspezifikationen für Signatur und Verifizierung Elliptic Curve(EC)-Schlüsselspezifikationen

Status

Wo: Abschnitt über allgemeine Konfiguration

Der Schlüsselstatus des KMS-Schlüssels. Sie können den KMS-Schlüssel nur in kryptografischen Operationen verwenden, wenn der Status Enabled (Aktiviert) lautet. Eine detaillierte Beschreibung eines jeden KMS-Schlüssel-Status und seiner Auswirkungen darauf, welche Operationen Sie für den KMS-Schlüssel ausführen können, finden Sie unter Wichtige Zustände von AWS KMS Schlüsseln.

Tags

Wo: Registerkarte für Tags

Optionale Schlüssel-Wert-Paare, die den KMS-Schlüssel beschreiben. Um die Tags für einen KMS-Schlüssel hinzuzufügen oder zu ändern, wählen Sie auf der Tags-Registerkarte die Option Edit (Bearbeiten).

Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenzuordnungsbericht generiert, der die Nutzung und die Kosten nach Stichwörtern zusammenfasst. Markierungen können auch verwendet werden, um den Zugriff auf einen KMS-Schlüssel zu steuern. Weitere Informationen über das Markieren von KMS-Schlüsseln finden Sie unter Schlagworte in AWS KMS und ABAC für AWS KMS.

Mithilfe der API AWS KMS

Der DescribeKeyVorgang gibt Details zum angegebenen KMS-Schlüssel zurück. Verwenden Sie zum Identifizieren eines KMS-Schlüssels seine Schlüssel-ID, den Schlüssel-ARN, den Aliasnamen oder den Alias-ARN.

Im Gegensatz zu dem ListKeysVorgang, bei dem nur KMS-Schlüssel im Konto und in der Region des Anrufers angezeigt werden, können autorisierte Benutzer den `DescribeKey` Vorgang verwenden, um Details zu KMS-Schlüsseln in anderen Konten abzurufen.

> **Note**
>
> Die `DescribeKey`-Antwort enthält sowohl `KeySpec` und `CustomerMasterKeySpec`-Elemente mit den gleichen Werten. Das `CustomerMasterKeySpec`-Element ist veraltet.

Dieser Aufruf an `DescribeKey` gibt beispielsweise Informationen zu einem KMS-Schlüssel mit symmetrischer Verschlüsselung zurück. Die Felder in der Antwort variieren je nach AWS KMS key -Spezifikation, Schlüsselstatus und Ursprung des Schlüsselmaterials. Beispiele in verschiedenen Programmiersprachen finden Sie unter Verwendung `DescribeKey` mit einem AWS SDK oder CLI.

```
$ aws kms describe-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
        "Enabled": true,
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "CreationDate": 1499988169.234,
        "MultiRegion": false,
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "CurrentKeyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    }
}
```

In diesem Beispiel wird die `DescribeKey`-Produktion für einen asymmetrischen KMS-Schlüssel aufgerufen, der für Signatur und Verifizierung verwendet wird. Die Antwort enthält die Signaturalgorithmen, die AWS KMS für diesen KMS-Schlüssel unterstützt.

```
$ aws kms describe-key --key-id 0987dcba-09fe-87dc-65ba-ab0987654321

{
    "KeyMetadata": {
        "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "Origin": "AWS_KMS",
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "CreationDate": 1569973196.214,
        "Description": "",
```

```
        "KeySpec": "ECC_NIST_P521",
        "CustomerMasterKeySpec": "ECC_NIST_P521",
        "AWSAccountId": "111122223333",
        "Enabled": true,
        "MultiRegion": false,
        "KeyManager": "CUSTOMER",
        "SigningAlgorithms": [
            "ECDSA_SHA_512"
        ]
    }
}
```

Identifizieren Sie verschiedene Schlüsseltypen

In den folgenden Themen wird erklärt, wie Sie verschiedene Schlüsseltypen in der AWS KMS Konsole und in den DescribeKeyAntworten identifizieren können.

Hilfe zur Navigation zur Registerkarte Kryptografische Konfiguration auf der Detailseite für einen KMS-Schlüssel finden Sie unter. the section called “KMS-Schlüsseldetails aufrufen und auflisten”

Themen

- Identifizieren Sie asymmetrische KMS-Schlüssel
- Identifizieren Sie HMAC-KMS-Schlüssel
- Identifizieren Sie KMS-Schlüssel für mehrere Regionen
- Identifizieren Sie KMS-Schlüssel mit importiertem Schlüsselmaterial
- Identifizieren Sie KMS-Schlüssel in AWS CloudHSM Schlüsselspeichern
- Identifizieren Sie KMS-Schlüssel in externen Schlüsselspeichern

Identifizieren Sie asymmetrische KMS-Schlüssel

In der Konsole AWS KMS

In der Spalte Schlüsseltyp der Tabelle Vom Kunden verwaltete Schlüssel wird angezeigt, ob jeder KMS-Schlüssel symmetrisch oder asymmetrisch ist. Sie können die Tabelle nach dem Wert Schlüsseltyp filtern, um nur asymmetrische KMS-Schlüssel anzuzeigen. Weitere Informationen finden Sie unter the section called “Sortieren und filtern Sie Ihre KMS-Schlüssel”.

Auf der Registerkarte Kryptografische Konfiguration auf der Detailseite für einen KMS-Schlüssel wird der Schlüsseltyp angezeigt, der angibt, ob der Schlüssel symmetrisch oder asymmetrisch ist. Außerdem wird die Schlüsselverwendung angezeigt, die angibt, ob Ihr asymmetrischer KMS-Schlüssel zur Verschlüsselung und Entschlüsselung, Signierung und Überprüfung oder zum Ableiten von gemeinsamen Geheimnissen verwendet wird.

In DescribeKey Antworten

Wenn Sie den `DescribeKey` Vorgang für einen asymmetrischen KMS-Schlüssel aufrufen, enthält die Antwort die `KeyUsage` Werte `KeySpec` und, anhand derer bestimmt werden kann, ob ein KMS-Schlüssel symmetrisch oder asymmetrisch ist.

Wenn der `KeySpec` Wert ist`SYMMETRIC_DEFAULT`, handelt es sich bei dem Schlüssel um einen KMS-Schlüssel mit symmetrischer Verschlüsselung. Einzelheiten zu den Spezifikationen für asymmetrische Schlüssel finden Sie unter. Referenz zu wichtigen Spezifikationen

Wenn der `KeyUsage` Wert `SIGN_VERIFY` oder ist`KEY_AGREEMENT`, handelt es sich bei dem Schlüssel um einen asymmetrischen KMS-Schlüssel.

Der `DescribeKey` Vorgang gibt auch die folgenden Details für asymmetrische KMS-Schlüssel zurück.

- Bei asymmetrischen KMS-Schlüsseln mit dem `KeyUsage` Wert von `ENCRYPT_DECRYPT` gibt der Vorgang den zurück`EncryptionAlgorithms`, der die gültigen Verschlüsselungsalgorithmen für den Schlüssel auflistet.
- Bei asymmetrischen KMS-Schlüsseln mit dem `KeyUsage` Wert von `SIGN_VERIFY` gibt die Operation den zurück`SigningAlgorithms`, der die gültigen Signaturalgorithmen für den Schlüssel auflistet.
- Bei asymmetrischen KMS-Schlüsseln mit dem `KeyUsage` Wert von `KEY_AGREEMENT` gibt die Operation den zurück`KeyAgreementAlgorithms`, der die gültigen Schlüsselvereinbarungsalgorithmen für den Schlüssel auflistet.

Weitere Informationen zu asymmetrischen KMS-Schlüsseln finden Sie unter. the section called “Asymmetrische Schlüssel”

Identifizieren Sie HMAC-KMS-Schlüssel

In der Konsole AWS KMS

HMAC-KMS-Schlüssel sind in der Tabelle mit vom Kunden verwalteten Schlüsseln enthalten, aber Sie können diese Tabelle nicht nach den Schlüsselspezifikationen oder Schlüsselverwendungswerten sortieren oder filtern, die HMAC-Schlüssel identifizieren. Um das Auffinden Ihrer HMAC-Schlüssel zu erleichtern, weisen Sie ihnen einen unverwechselbaren Alias oder Tag zu. Dann können Sie nach dem Alias oder Tag sortieren oder filtern.

Auf der Registerkarte Kryptografische Konfiguration auf der Detailseite für einen KMS-Schlüssel wird der Schlüsseltyp angezeigt, der angibt, ob der Schlüssel symmetrisch oder asymmetrisch ist. HMAC-KMS-Schlüssel sind symmetrisch. Auf der Registerkarte Kryptografische Konfiguration wird auch die Schlüsselverwendung angezeigt. Für HMAC-KMS-Schlüssel lautet der Wert für die Schlüsselverwendung immer MAC generieren und verifizieren.

In Antworten DescribeKey

Wenn Sie den `DescribeKey` Vorgang mit einem HMAC-KMS-Schlüssel aufrufen, enthält die Antwort die `KeyUsage` Werte `KeySpec` und. Bei HMAC-KMS-Schlüsseln ist der Wert für die Schlüsselverwendung immer `GENERATE_VERIFY_MAC` und der Wert für die Schlüsselspezifikation beginnt immer mit. `HMAC_`

Weitere Informationen zu HMAC-KMS-Schlüsseln finden Sie unter. the section called “HMAC-Schlüssel”

Identifizieren Sie KMS-Schlüssel für mehrere Regionen

In der Konsole AWS KMS

In der Tabelle mit vom Kunden verwalteten Schlüsseln werden nur KMS-Schlüssel in der ausgewählten Region angezeigt. Sie können multiregionale Primär- und Replikatschlüssel in der ausgewählten Region anzeigen. Um die AWS Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Konsole.

Um die Identifizierung von Schlüsseln mit mehreren Regionen in der Tabelle „Vom Kunden verwaltete Schlüssel“ zu vereinfachen, fügen Sie Ihrer Tabelle die Spalte Regionalität hinzu. Weitere Informationen dazu finden Sie unter the section called “Passen Sie Ihre KMS-Schlüsseltabellen an”.

Die Detailseite für KMS-Schlüssel mit mehreren Regionen enthält die Registerkarte Regionalität. Die Registerkarte Regionality (Regionalität) eines Primärschlüssels enthält die Schaltflächen Primärbereich ändern und Neue Replikatschlüssel erstellen. (Die Regionalität-Registerkarte für einen Replikatschlüssel enthält keine dieser Schaltflächen.) Der Abschnitt Related multi-Region keys (verwandte multiregionale Schlüssel) listet alle multiregionale Schlüssel auf, die mit dem aktuellen verwandt sind. Wenn der aktuelle Schlüssel ein Replikatschlüssel ist, enthält diese Liste den Primärschlüssel.

Wenn Sie einen verwandten Schlüssel für mehrere Regionen aus der Tabelle Verwandte Schlüssel für mehrere Regionen auswählen, wechselt die AWS KMS Konsole zur Region des ausgewählten Schlüssels und die Detailseite für den Schlüssel wird geöffnet. Wenn Sie beispielsweise den Replikatschlüssel in der `sa-east-1` Region aus dem Abschnitt „Verwandte Schlüssel für mehrere Regionen“ unten auswählen, wechselt die AWS KMS Konsole zur **`sa-east-1`** Region, um die Detailseite für diesen Replikatschlüssel anzuzeigen. Sie könnten dies tun, um den Alias- oder die Schlüsselrichtlinie für den Replikatschlüssel anzuzeigen. Um die Region erneut zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.

In Antworten DescribeKey

Standardmäßig sind AWS KMS API-Operationen regional und geben nur die Ressourcen in der aktuellen oder angegebenen Region zurück. Wenn Sie den `DescribeKey` Vorgang jedoch für einen KMS-Schlüssel mit mehreren Regionen aufrufen, umfasst die Antwort alle zugehörigen Schlüssel für mehrere Regionen in anderen AWS Regionen des `MultiRegionConfiguration` Elements.

Weitere Informationen zu KMS-Schlüsseln für mehrere Regionen finden Sie unter. the section called “Multiregionale Schlüssel”

Identifizieren Sie KMS-Schlüssel mit importiertem Schlüsselmaterial

In der AWS KMS Konsole

Um die Identifizierung von KMS-Schlüsseln mit importiertem Schlüsselmaterial in der Tabelle mit vom Kunden verwalteten Schlüsseln zu vereinfachen, fügen Sie Ihrer Tabelle die Origin-Spalte hinzu. In der Ursprung-Spalte sind KMS-Schlüssel mit einem Externen (Schlüsselmaterial importieren)-Ursprung-Eigenschaftswert leicht erkennbar. Weitere Informationen dazu finden Sie unter the section called “Passen Sie Ihre KMS-Schlüsseltabellen an”.

Auf der Registerkarte Kryptografische Konfiguration auf der Detailseite für einen KMS-Schlüssel wird der Ursprung angezeigt, der die Quelle des Schlüsselmaterials für den KMS-Schlüssel identifiziert. Bei KMS-Schlüsseln mit importiertem Schlüsselmaterial ist der Ursprungswert immer Extern (Schlüsselmaterial importieren). Die Detailseite enthält auch eine Registerkarte Schlüsselmaterial, die detaillierte Informationen über das importierte Schlüsselmaterial enthält. Symmetrische Verschlüsselungsschlüssel mit `EXTERNAL` Ursprung in einer Region unterstützen Rotationen nach Bedarf und können mit mehreren Schlüsselmaterialien verknüpft werden. Bei solchen Schlüsseln trägt die Registerkarte die Bezeichnung Schlüsselmaterial und Rotationen.

In Antworten DescribeKey

Wenn Sie den `DescribeKey` Vorgang für einen KMS-Schlüssel mit importiertem Schlüsselmaterial aufrufen, enthält die Antwort die `ValidTo` Werte `OriginExpirationModel`, und. Bei KMS-Schlüsseln mit importiertem Schlüsselmaterial ist der Ursprungswert immer`EXTERNAL`. Der `ExpirationModel` Wert gibt an, ob das Schlüsselmaterial abläuft oder nicht, und der `ValidTo` Wert gibt an, wann das Schlüsselmaterial abläuft. Wenn einem Schlüssel mehrere Schlüsselmaterialien zugeordnet sind, gibt der `ValidTo` Wert die früheste Ablaufzeit für alle Schlüsselmaterialien an (mit Ausnahme der einen ausstehenden Rotation) und `ExpirationModel` wird nur dann auf „`DOES_NOT_EXPIRE`Nur“ gesetzt, wenn für keines dieser Schlüsselmaterialien ein Ablaufdatum festgelegt ist. Weitere Informationen finden Sie unter Festlegen einer Ablaufzeit (optional).

Weitere Informationen zu KMS-Schlüsseln mit importiertem Schlüsselmaterial finden Sie unterthe section called “Importiertes Schlüsselmaterial”.

Identifizieren Sie KMS-Schlüssel in AWS CloudHSM Schlüsselspeichern

In der AWS KMS Konsole

Um die Identifizierung von KMS-Schlüsseln in AWS CloudHSM Schlüsselspeichern in der Tabelle mit vom Kunden verwalteten Schlüsseln zu vereinfachen, fügen Sie Ihrer Tabelle die Origin-Spalte hinzu. In der Origin (Ursprung)-Spalte sind KMS-Schlüssel mit einem AWS CloudHSM-Ursprung-Eigenschaftswert leicht erkennbar. Weitere Informationen dazu finden Sie unter the section called “Passen Sie Ihre KMS-Schlüsseltabellen an”.

Auf der Registerkarte Kryptografische Konfiguration auf der Detailseite für einen KMS-Schlüssel wird der Ursprung angezeigt, der die Quelle des Schlüsselmaterials für den KMS-Schlüssel identifiziert. Bei KMS-Schlüsseln in AWS CloudHSM Schlüsselspeichern ist der Ursprungswert immer AWS CloudHSM.

Für einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher enthält die Registerkarte Kryptografische Konfiguration einen zusätzlichen Abschnitt, Benutzerdefinierter Schlüsselspeicher, der Informationen über den AWS CloudHSM Schlüsselspeicher und den AWS CloudHSM Cluster enthält, die dem KMS-Schlüssel zugeordnet sind.

In Antworten DescribeKey

Wenn Sie den `DescribeKey` Vorgang mit einem KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher aufrufen, enthält die Antwort den`Origin`, der die Quelle des Schlüsselmaterials identifiziert. Bei KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher ist der Ursprungswert immer`AWS_CLOUDHSM`. Die Operation gibt auch die folgenden speziellen Felder für KMS-Schlüssel in AWS CloudHSM Schlüsselspeichern zurück:

- `CloudHsmClusterId`
- `CustomKeyStoreId`

Weitere Informationen zu AWS CloudHSM Schlüsselspeichern finden Sie unterthe section called “AWS CloudHSM wichtige Geschäfte”.

Identifizieren Sie KMS-Schlüssel in externen Schlüsselspeichern

In der AWS KMS Konsole

Um die Identifizierung von KMS-Schlüsseln in externen Schlüsselspeichern in der Tabelle mit vom Kunden verwalteten Schlüsseln zu vereinfachen, fügen Sie Ihrer Tabelle die Origin-Spalte hinzu. Die Origin-Spalte erleichtert die Identifizierung von KMS-Schlüsseln mit dem Eigenschaftswert „Externer Schlüsselspeicher“. Weitere Informationen dazu finden Sie unter the section called “Passen Sie Ihre KMS-Schlüsseltabellen an”.

Auf der Registerkarte Kryptografische Konfiguration auf der Detailseite für einen KMS-Schlüssel wird der Ursprung angezeigt, der die Quelle des Schlüsselmaterials für den KMS-Schlüssel identifiziert. Für KMS-Schlüssel in externen Schlüsselspeichern lautet der Ursprungswert immer Externer Schlüsselspeicher.

Für einen KMS-Schlüssel in einem externen Schlüsselspeicher enthält die Registerkarte Kryptografische Konfiguration zwei zusätzliche Abschnitte: Benutzerdefinierter Schlüsselspeicher und Externer Schlüssel. Die Tabelle Benutzerdefinierter Schlüsselspeicher enthält Informationen über den externen Schlüsselspeicher, der dem KMS-Schlüssel zugeordnet ist. Die Tabelle mit externen Schlüsseln wird in der AWS KMS Konsole nur für KMS-Schlüssel in externen Schlüsselspeichern angezeigt. Er enthält Informationen über den externen Schlüssel, der dem

KMS-Schlüssel zugeordnet ist. Der externe Schlüssel ist ein kryptografischer Schlüssel AWS , der jedoch als Schlüsselmaterial für den KMS-Schlüssel im externen Schlüsselspeicher dient. Wenn Sie mit dem KMS-Schlüssel verschlüsseln oder entschlüsseln, wird der Vorgang von Ihrem externen Schlüsselmanager unter Verwendung des angegebenen externen Schlüssels ausgeführt.

Die folgenden Werte werden im Abschnitt External key (Externer Schlüssel) angezeigt.

ID des externen Schlüssels

Die Kennung für den externen Schlüssel in seinem externen Schlüsselmanager. Diesen Wert verwendet der Proxy des externen Schlüsselspeichers, um den externen Schlüssel zu identifizieren. Sie geben die ID des externen Schlüssels an, wenn Sie den KMS-Schlüssel erstellen. Sie kann danach nicht mehr geändert werden. Wenn sich der Wert der ID des externen Schlüssels, den Sie zur Erstellung des KMS-Schlüssels verwendet haben, ändert oder ungültig wird, müssen Sie die Löschung des KMS-Schlüssel planen und einen neuen KMS-Schlüssel mit dem korrekten Wert der ID des externen Schlüssels erstellen.

In Antworten DescribeKey

Wenn Sie den `DescribeKey` Vorgang für einen KMS-Schlüssel in einem externen Schlüsselspeicher aufrufen, enthält die Antwort den`Origin`, der die Quelle des Schlüsselmaterials identifiziert. Bei KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher ist der Ursprungswert immer`EXTERNAL_KEY_STORE`. Die Operation gibt auch das `CustomKeyStoreId` Element zurück, das den externen Schlüsselspeicher identifiziert, der den KMS-Schlüsseln zugeordnet ist.

Weitere Informationen zu externen Schlüsselspeichern finden Sie unterthe section called “Externe Schlüsselspeicher”.

Passen Sie Ihre Konsolenansicht an

Sie können die Ansicht der AWS KMS Konsole anpassen, um das Auffinden Ihrer KMS-Schlüssel zu erleichtern. Passen Sie die Tabellen auf den Seiten Von AWS verwaltete Schlüsselund Vom Kunden verwaltete Schlüssel angezeigten Tabellen so an, dass die Informationen angezeigt werden, die Sie am dringendsten benötigen, oder sortieren und filtern Sie die in den Tabellen zurückgegebenen KMS-Schlüssel.

Themen

- Sortieren und filtern Sie Ihre KMS-Schlüssel
- Passen Sie Ihre KMS-Schlüsseltabellen an

Sortieren und filtern Sie Ihre KMS-Schlüssel

Um das Auffinden Ihrer KMS-Schlüssel in der Konsole zu erleichtern, können Sie die Schlüsseltabellen sortieren und filtern.

Sortierung

Sie können kundenverwaltete KMS-Schlüssel in aufsteigender oder absteigender Reihenfolge nach ihren Spaltenwerten sortieren. Diese Funktion sortiert alle KMS-Schlüssel in der Tabelle, auch wenn sie nicht auf der aktuellen Tabellenseite angezeigt werden.

Sortierbare Spalten werden durch einen Pfeil neben dem Spaltennamen gekennzeichnet. Auf der Seite Von AWS verwaltete Schlüssel können Sie nach Aliases (Aliase) oder Key ID (Schlüssel-ID) sortieren. Auf der Seite Customer managed keys (kundenverwaltete Schlüssel) können Sie nach Alias, Schlüssel-ID oder Schlüsseltyp sortieren.

Um in aufsteigender Reihenfolge zu sortieren, wählen Sie die Spaltenüberschrift aus, bis der Pfeil nach oben zeigt. Um in absteigender Reihenfolge zu sortieren, wählen Sie die Spaltenüberschrift, bis der Pfeil nach unten zeigt. Es kann jeweils nur nach einer Spalte sortiert werden.

Beispielsweise können Sie KMS-Schlüssel in aufsteigender Reihenfolge nach Schlüssel-ID sortieren, anstatt standardmäßig nach Alias.

Wenn Sie KMS-Schlüssel auf der Seite Customer managed keys (kundenverwaltete Schlüssel) in aufsteigender Reihenfolge nach Schlüsseltyp sortieren, werden alle asymmetrischen Schlüssel vor allen symmetrischen Schlüsseln angezeigt.

Filter

Sie können KMS-Schlüssel nach ihren Eigenschaftswerten oder Tags filtern. Der Filter gilt für alle KMS-Schlüssel in der Tabelle, auch wenn sie nicht auf der aktuellen Tabellenseite angezeigt werden. Bei dem Filter wird die Groß-/Kleinschreibung nicht berücksichtigt.

Filterbare Eigenschaften werden im Filterfeld aufgeführt. Auf der Seite Von AWS verwaltete Schlüssel können Sie nach Alias und Schlüssel-ID filtern. Auf der Seite Customer managed

keys (Kundenverwaltete Schlüssel) können Sie nach Alias, Schlüssel-ID und Schlüsseltyp-Eigenschaften sowie nach Tags filtern.

- Auf der Seite Von AWS verwaltete Schlüssel können Sie nach Alias und Schlüssel-ID filtern.
- Auf der Seite Customer managed keys (Kundenverwaltete Schlüssel) können Sie nach Tags filtern, oder nach dem Alias, der Schlüssel-ID, des Schlüsseltyps oder den Regionalität-Eigenschaften.

Um nach einem Eigenschaftswert zu filtern, wählen Sie den Filter und den Eigenschaftsnamen aus und wählen Sie dann aus der Liste der tatsächlichen Eigenschaftswerte aus. Um nach einem Tag zu filtern, wählen Sie den Tag-Schlüssel aus, und wählen Sie dann aus der Liste der tatsächlichen Tag-Werte. Nachdem Sie eine Eigenschaft oder einen Tag-Schlüssel ausgewählt haben, können Sie den Eigenschaftswert oder den Tag-Wert auch ganz oder teilweise eingeben. Sie sehen eine Vorschau der Ergebnisse, bevor Sie Ihre Wahl treffen.

Um beispielsweise KMS-Schlüssel mit einem Aliasnamen anzuzeigen, der `aws/e` enthält, wählen Sie das Filterfeld und danach Alias aus, geben Sie `aws/e` ein und drücken Sie dann `Enter` oder `Return`, um den Filter hinzuzufügen.

Vorgeschlagene Filter für KMS-Schlüsseltabellen

Filter für asymmetrische KMS-Schlüssel

Um nur asymmetrische KMS-Schlüssel auf der Seite Customer managed keys (kundenverwaltete Schlüssel) anzuzeigen, klicken Sie auf das Filterfeld, wählen Sie Schlüsseltyp und dann Schlüsseltyp: Asymmetrisch aus. Die Option Asymmetrisch wird nur angezeigt, wenn asymmetrische KMS-Schlüssel in der Tabelle vorhanden sind.

Filter für Schlüssel mit mehreren Regionen

Um nur multiregionale Schlüssel anzuzeigen, wählen Sie auf der Seite Customer managed keys (kundenverwaltete Schlüssel) im Filterfeld Regionality (Regionalität) aus, und dann Regionality: Multi-Region. Die Multi-Region-Option wird nur angezeigt, wenn multiregionale Schlüssel in der Tabelle vorhanden sind.

Nach Stichwörtern filtern

Um nur KMS-Schlüssel mit einem bestimmten Tag anzuzeigen, wählen Sie das Filterfeld aus, wählen Sie den Tag-Schlüssel aus, und wählen Sie dann einen der tatsächlichen Tag-Werte aus. Sie können den Tag-Wert auch ganz oder teilweise eingeben.

Es werden in der Tabelle alle KMS-Schlüssel mit dem ausgewählten Tag angezeigt. Das Tag wird jedoch nicht angezeigt. Um das Tag anzuzeigen, wählen Sie die Schlüssel-ID oder den Alias des KMS-Schlüssels aus und auf der Detailseite die Tags-Registerkarte. Die Registerkarte wird unter dem Abschnitt General Configuration (allgemeine Konfiguration) angezeigt.

Dieser Filter erfordert sowohl den Tag-Schlüssel als auch den Tag-Wert. Es wird keine KMS-Schlüssel finden, indem nur der Tag-Schlüssel oder nur dessen Wert eingegeben wird. Um Tags ganz oder teilweise nach dem Tag-Schlüssel oder -Wert zu filtern, verwenden Sie den ListResourceTagsVorgang, um markierte KMS-Schlüssel abzurufen, und verwenden Sie dann die Filterfunktionen Ihrer Programmiersprache.

Nach Text filtern

Um nach Text zu suchen, geben Sie im Filterfeld einen Alias, eine Schlüssel-ID, einen Schlüsseltyp oder einen Tag-Schlüssel ganz oder teilweise ein. (Nachdem Sie den Tag-Schlüssel ausgewählt haben, können Sie nach einem Tag-Wert suchen.) Sie sehen eine Vorschau der Ergebnisse, bevor Sie Ihre Wahl treffen.

Um beispielsweise KMS-Schlüssel mit `test` in ihren Tag-Schlüsseln oder filterbaren Eigenschaften anzuzeigen, geben Sie `test` in das Filterfeld ein. Die Vorschau zeigt die KMS-Schlüssel, die der Filter auswählen wird. In diesem Fall wird `test` nur in der Alias-Eigenschaft angezeigt.

Passen Sie Ihre KMS-Schlüsseltabellen an

Sie können die Tabellen, die auf den Seiten Von AWS verwaltete Schlüsselund Vom Kunden verwaltete Schlüssel angezeigt werden AWS Management Console , an Ihre Bedürfnisse anpassen. Sie können die Tabellenspalten, die Anzahl der Spalten AWS KMS keys auf jeder Seite (Seitengröße) und den Textumbruch auswählen. Die von Ihnen gewählte Konfiguration wird gespeichert, wenn Sie sie bestätigen, und bei jedem Öffnen der Seiten erneut angewendet.

Anpassen Ihrer KMS-Schlüsseltabellen

1. Wählen Sie auf der Seite Von AWS verwaltete Schlüssel oder Customer managed keys (Vom Kunden verwaltete Schlüssel) das Einstellungen-Symbol ( ) oben rechts auf der Seite aus.
2. Wählen Sie auf der Seite Preferences (Einstellungen) die gewünschten Einstellungen aus und wählen Sie dann Confirm (Bestätigen) aus.

Erwägen Sie, mit der Einstellung Page size (Seitengröße) die Anzahl der auf jeder Seite angezeigten KMS-Schlüssel zu erhöhen, insbesondere wenn Sie normalerweise ein Gerät verwenden, das leicht zu scrollen ist.

Die angezeigten Datenspalten können abhängig von der Tabelle, Ihrer Aufgabenrolle und den Typen von KMS-Schlüsseln im Konto und in der Region variieren. Die folgende Tabelle enthält einige vorgeschlagene Konfigurationen. Beschreibungen der Spalten finden Sie unter Verwenden der Konsole AWS KMS.

Empfohlene Konfigurationen für KMS-Schlüsseltabellen

Sie können die Spalten anpassen, die in der KMS-Schlüsseltabelle angezeigt werden, damit die benötigten KMS-Schlüsseldaten sichtbar werden.

Von AWS verwaltete Schlüssel

Standardmäßig enthält die Von AWS verwalteter Schlüssel-Tabelle die Spalten Aliases (Aliase), Key ID (Schlüssel-ID) und Status. Diese Spalten sind für die meisten Anwendungsfälle gut geeignet.

KMS-Schlüssel mit symmetrischer Verschlüsselung

Wenn Sie nur KMS-Schlüssel mit symmetrischer Verschlüsselung mit Schlüsselmaterial verwenden, das von AWS KMS generiert wurde, sind die Spalten Aliases (Aliase), Key ID (Schlüssel-ID), Status und Creation date (Erstellungsdatum) wahrscheinlich am nützlichsten.

Asymmetrische KMS-Schlüssel

Wenn Sie asymmetrische KMS-Schlüssel verwenden, sollten Sie zusätzlich zu den Spalten Aliases (Aliase), Key ID (Schlüssel-ID) und Status die Spalten Key type (Schlüsseltyp), Key spec (Schlüsselspezifikation) und Key usage (Schlüsselnutzung) hinzufügen. In diesen Spalten wird angezeigt, ob ein KMS-Schlüssel symmetrisch oder asymmetrisch ist, der Typ des Schlüsselmaterials und ob der KMS-Schlüssel für die Verschlüsselung oder Signatur verwendet werden kann.

HMAC-KMS-Schlüssel

Wenn Sie HMAC-KMS-Schlüssel verwenden, sollten Sie zusätzlich zu den Spalten Aliases (Aliase), Key ID (Schlüssel-ID) und Status das Hinzufügen der Spalten Key type (Schlüsseltyp), Key spec (Schlüsselspezifikation) und Key usage (Schlüsselnutzung) erwägen. Diese Spalten zeigen Ihnen, ob ein KMS-Schlüssel ein HMAC-Schlüssel ist. Da Sie KMS-Schlüssel nicht nach Schlüsselspezifikation oder Schlüsselverwendung sortieren können, verwenden Sie Aliase und

Tags, um Ihre HMAC-Schlüssel zu identifizieren, und verwenden Sie dann die Filterfunktionen der AWS KMS Konsole, um nach Aliasen oder Tags zu filtern.

Importiertes Schlüsselmaterial

Wenn Sie KMS-Schlüssel mit importiertem Schlüsselmaterial haben, erwägen Sie, die Spalten Origin (Ursprung) und Expiration date (Ablaufdatum) hinzuzufügen. In diesen Spalten wird angezeigt, ob das Schlüsselmaterial in einem KMS-Schlüssel importiert oder generiert wurde AWS KMS und wann das Schlüsselmaterial abläuft, wenn überhaupt. Das Feld Creation date (Erstellungsdatum) zeigt das Datum an, an dem der KMS-Schlüssel erstellt wurde (ohne Schlüsselmaterial). Es spiegelt keine Eigenschaften des Schlüsselmaterials wider.

Schlüssel in benutzerdefinierten Schlüsselspeichern

Wenn Sie KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern haben, erwägen Sie, die Spalten Origin (Ursprung) und Custom key store ID (ID des benutzerdefinierten Schlüsselspeichers) hinzuzufügen. Diese Spalten geben an, dass sich der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher befindet, zeigen den Typ des benutzerdefinierten Schlüsselspeichers an und identifizieren den benutzerdefinierten Schlüsselspeicher.

Multiregionale Schlüssel

Wenn Sie multiregionale Schlüssel haben, erwägen Sie, die Spalte Regionality (Regionalität) hinzuzufügen. Gibt an, ob ein KMS-Schlüssel ein einzelregionaler Schlüssel, ein multiregionaler Primärschlüssel oder ein multiregionaler Replikatschlüssel ist.

Suchen Sie KMS-Schlüssel und Schlüsselmaterial in einem AWS CloudHSM Schlüsselspeicher

Wenn Sie einen AWS CloudHSM Schlüsselspeicher verwalten, müssen Sie möglicherweise die KMS-Schlüssel in jedem AWS CloudHSM Schlüsselspeicher identifizieren. Beispielsweise könnte es sein, dass Sie eine der folgenden Aufgaben ausführen müssen:

- Verfolgen Sie die KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher in AWS CloudTrail Protokollen.
- Prognostizieren Sie die Auswirkungen einer Unterbrechung der Verbindung zu einem AWS CloudHSM Schlüsselspeicher auf KMS-Schlüssel.
- Planen Sie das Löschen von KMS-Schlüsseln, bevor Sie einen AWS CloudHSM Schlüsselspeicher löschen.

Darüber hinaus möchten Sie möglicherweise die Schlüssel in Ihrem AWS CloudHSM Cluster identifizieren, die als Schlüsselmaterial für Ihre KMS-Schlüssel dienen. AWS KMS Verwaltet zwar die KMS-Schlüssel und das Schlüsselmaterial, Sie behalten jedoch weiterhin die Kontrolle und Verantwortung für die Verwaltung Ihres AWS CloudHSM Clusters sowie für die Backups HSMs und Schlüssel in der HSMs. Möglicherweise müssen Sie die Schlüssel identifizieren, um das Schlüsselmaterial zu prüfen, es vor versehentlichem Löschen zu schützen oder es nach dem Löschen des KMS-Schlüssels aus HSMs den Cluster-Backups zu löschen.

Das gesamte Schlüsselmaterial für die KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher gehört dem `kmsuser`Crypto User (CU). AWS KMS setzt das Schlüsselbezeichnungsattribut, das nur in sichtbar ist AWS CloudHSM, auf den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels.

Wenden Sie zum Suchen der KMS-Schlüssel und Schlüsselmaterial eine der folgenden Methoden an.

- Suchen Sie die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher— So identifizieren Sie die KMS-Schlüssel in einem oder allen Ihrer AWS CloudHSM Schlüsselspeicher.
- Suchen Sie nach allen Schlüsseln für einen AWS CloudHSM Schlüsselspeicher – Ermitteln aller Schlüssel in Ihrem Cluster, die als Schlüsselmaterial für die KMS-Schlüssel in Ihrem AWS CloudHSM -Schlüsselspeicher dienen.
- Suchen Sie den AWS CloudHSM Schlüssel für einen KMS-Schlüssel— So finden Sie den Schlüssel in Ihrem Cluster, der als Schlüsselmaterial für einen bestimmten KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher dient.
- Suchen Sie den KMS-Schlüssel für einen AWS CloudHSM Schlüssel – Ermitteln des KMS-Schlüssels für einen bestimmten Schlüssel in Ihrem Cluster.

Suchen Sie die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher

Wenn Sie einen AWS CloudHSM Schlüsselspeicher verwalten, müssen Sie möglicherweise die KMS-Schlüssel in jedem AWS CloudHSM Schlüsselspeicher identifizieren. Sie können diese Informationen verwenden, um die KMS-Schlüsseloperationen in AWS CloudTrail Protokollen nachzuverfolgen, die Auswirkungen der Trennung eines benutzerdefinierten Schlüsselspeichers auf KMS-Schlüssel vorherzusagen oder das Löschen von KMS-Schlüsseln zu planen, bevor Sie einen AWS CloudHSM Schlüsselspeicher löschen.

So finden Sie die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher (Konsole)

Um die KMS-Schlüssel in einem bestimmten AWS CloudHSM Schlüsselspeicher zu finden, sehen Sie sich auf der Seite vom Kunden verwaltete Schlüssel die Werte in den Feldern Custom Key Store Name oder Custom Key Store ID an. Um KMS-Schlüssel in einem beliebigen AWS CloudHSM Schlüsselspeicher zu identifizieren, suchen Sie nach KMS-Schlüsseln mit dem Origin-Wert von AWS CloudHSM. Wenn Sie der Anzeige optionale Spalten hinzufügen möchten, wählen Sie das Zahnradsymbol rechts oben auf der Seite aus.

So finden Sie die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher (API)

Um die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher zu finden, verwenden Sie die DescribeKeyOperationen ListKeysund und filtern Sie dann nach `CustomKeyStoreId` Wert. Bevor Sie die folgenden Beispiele ausführen, ersetzen Sie die fiktiven ID-Werte des benutzerdefinierten Schlüsselspeichers durch einen gültigen Wert.

Bash

Um KMS-Schlüssel in einem bestimmten AWS CloudHSM Schlüsselspeicher zu finden, rufen Sie alle Ihre KMS-Schlüssel im Konto und in der Region ab. Filtern Sie anschließend nach der ID des benutzerdefinierten Schlüsselspeichers.

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;
do aws kms describe-key --key-id $key |
grep '"CustomKeyStoreId": "cks-1234567890abcdef0"' --context 100; done
```

Um KMS-Schlüssel in einem beliebigen AWS CloudHSM Schlüsselspeicher des Kontos und der Region abzurufen, suchen Sie nach `CustomKeyStoreType` dem Wert`AWS_CloudHSM`.

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;
do aws kms describe-key --key-id $key |
grep '"CustomKeyStoreType": "AWS_CloudHSM"' --context 100; done
```

PowerShell

Um KMS-Schlüssel in einem bestimmten AWS CloudHSM Schlüsselspeicher zu finden, verwenden Sie die KmsKey Cmdlets Get- KmsKeyList und Get-, um all Ihre KMS-Schlüssel im Konto und in der Region abzurufen. Filtern Sie anschließend nach der ID des benutzerdefinierten Schlüsselspeichers.

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyStoreId -eq
 'cks-1234567890abcdef0'
```

Um KMS-Schlüssel in einem beliebigen AWS CloudHSM Schlüsselspeicher des Kontos und der Region abzurufen, filtern Sie nach dem CustomKeyStoreType Wert von. `AWS_CLOUDHSM`

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyStoreType -eq 'AWS_CLOUDHSM'
```

Suchen Sie nach allen Schlüsseln für einen AWS CloudHSM Schlüsselspeicher

Sie können die Schlüssel in Ihrem AWS CloudHSM Cluster identifizieren, die als Schlüsselmaterial für Ihren AWS CloudHSM Schlüsselspeicher dienen. Verwenden Sie dazu den Befehl key list in der CloudHSM-CLI.

Sie können auch den Befehl key list verwenden, um nach einem AWS CloudHSM Schlüssel zu suchen. AWS KMS Wenn das Schlüsselmaterial für einen KMS-Schlüssel in Ihrem AWS CloudHSM Cluster AWS KMS erstellt wird, schreibt es den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels in die Schlüsselbezeichnung. Der Befehl key list gibt das `key-reference` und das zurück`label`.

> **Hinweise**
>
> Die folgenden Verfahren verwenden das AWS CloudHSM Client SDK 5-Befehlszeilentool CloudHSM CLI. Die CloudHSM-CLI ersetzt durch`key-handle`. `key-reference`
> Am 1. Januar 2025 AWS CloudHSM wird der Support für die Client SDK 3-Befehlszeilentools, das CloudHSM Management Utility (CMU) und das Key Management Utility (KMU) eingestellt. Weitere Informationen zu den Unterschieden zwischen den Client SDK 3-Befehlszeilentools und dem Client SDK 5-Befehlszeilentool finden Sie unter Migrieren von Client SDK 3 CMU und KMU zu Client SDK 5 CloudHSM CLI im Benutzerhandbuch.AWS CloudHSM

Um dieses Verfahren auszuführen, müssen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher vorübergehend trennen, sodass Sie sich als CU anmelden können. `kmsuser`

1. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher, falls er nicht bereits getrennt ist, und melden Sie sich dann an als`kmsuser`, wie unter beschriebenTrennen und Anmelden.

> **Note**
>
> Sämtliche Versuche, KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zu erstellen oder vorhandene KMS-Schlüssel in kryptografischen Produktionen zu nutzen, schlagen fehl, während der benutzerdefinierte Schlüsselspeicher getrennt ist. Diese Aktion kann verhindern, dass Benutzer vertrauliche Daten speichern und darauf zugreifen.

2. Verwenden Sie den Befehl key list in der CloudHSM-CLI, um alle Schlüssel für den aktuellen Benutzer in Ihrem AWS CloudHSM Cluster zu finden.

   Standardmäßig werden nur 10 Schlüssel des aktuell angemeldeten Benutzers angezeigt, und nur die Schlüssel `key-reference` und `label` werden als Ausgabe angezeigt. Weitere Optionen finden Sie in der Schlüsselliste im AWS CloudHSM Benutzerhandbuch.

```
aws-cloudhsm > key list
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000000123",
        "attributes": {
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
      },
      {
        "key-reference": "0x0000000000000456",
        "attributes": {
          "label": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
      },.
      ...8 keys later...
    ],
    "total_key_count": 56,
```

```
    "returned_key_count": 10,
    "next_token": "10"
  }
}
```

3. Melden Sie sich ab und stellen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher wieder her, wie unter beschriebenAbmelden und erneutes Verbinden.

Suchen Sie den KMS-Schlüssel für einen AWS CloudHSM Schlüssel

Wenn Sie die Schlüsselreferenz oder ID eines Schlüssels kennen, den er im Cluster `kmsuser` besitzt, können Sie diesen Wert verwenden, um den zugehörigen KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher zu identifizieren.

Wenn das Schlüsselmaterial für einen KMS-Schlüssel in Ihrem AWS CloudHSM Cluster AWS KMS erstellt wird, schreibt es den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels in die Schlüsselbezeichnung. Sofern Sie den Labelwert nicht geändert haben, können Sie den Befehl key list in der CloudHSM CLI verwenden, um den KMS-Schlüssel zu identifizieren, der dem Schlüssel zugeordnet ist. AWS CloudHSM

> **Hinweise**
>
> Die folgenden Verfahren verwenden das AWS CloudHSM Client SDK 5-Befehlszeilentool CloudHSM CLI. Die CloudHSM-CLI ersetzt durch`key-handle`. `key-reference`
> Am 1. Januar 2025 AWS CloudHSM wird der Support für die Client SDK 3-Befehlszeilentools, das CloudHSM Management Utility (CMU) und das Key Management Utility (KMU) eingestellt. Weitere Informationen zu den Unterschieden zwischen den Client SDK 3-Befehlszeilentools und dem Client SDK 5-Befehlszeilentool finden Sie unter Migrieren von Client SDK 3 CMU und KMU zu Client SDK 5 CloudHSM CLI im Benutzerhandbuch.AWS CloudHSM

Um diese Verfahren auszuführen, müssen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher vorübergehend trennen, sodass Sie sich als CU anmelden können. `kmsuser`

> **Note**
>
> Sämtliche Versuche, KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zu erstellen oder vorhandene KMS-Schlüssel in kryptografischen Produktionen zu nutzen,

schlagen fehl, während der benutzerdefinierte Schlüsselspeicher getrennt ist. Diese Aktion kann verhindern, dass Benutzer vertrauliche Daten speichern und darauf zugreifen.

Themen

- Identifizieren Sie den KMS-Schlüssel, der einer Schlüsselreferenz zugeordnet ist
- Identifizieren Sie den KMS-Schlüssel, der einer Backing-Key-ID zugeordnet ist

Identifizieren Sie den KMS-Schlüssel, der einer Schlüsselreferenz zugeordnet ist

Die folgenden Verfahren zeigen, wie Sie den Befehl key list in der CloudHSM-CLI mit dem `key-reference` Attributfilter verwenden, um den Schlüssel in Ihrem Cluster zu finden, der als Schlüsselmaterial für einen bestimmten KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher dient.

1. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher, falls er nicht bereits getrennt ist, und melden Sie sich dann wie unter `kmsuser` beschrieben an. Trennen und Anmelden
2. Verwenden Sie den Befehl key list in der CloudHSM-CLI, um nach dem `key-reference` Attribut zu filtern. Geben Sie das `verbose` Argument an, das alle Attribute und Schlüsselinformationen für den übereinstimmenden Schlüssel enthalten soll. Wenn Sie das `verbose` Argument nicht angeben, gibt die Schlüssellistenoperation nur die Schlüsselreferenz und das Labelattribut des übereinstimmenden Schlüssels zurück.

   Bevor Sie diesen Befehl ausführen, ersetzen Sie das Beispiel `key-reference` durch ein gültiges Beispiel aus Ihrem Konto.

```
aws-cloudhsm > key list --filter attr.key-reference="0x0000000000120034" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000120034",
        "key-info": {
          "key-owners": [
            {
              "username": "kmsuser",
```

```
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "id": "0xbacking-key-id",
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": false,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": false,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. Melden Sie sich ab und stellen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher wieder her, wie unter beschriebenAbmelden und erneutes Verbinden.

Identifizieren Sie den KMS-Schlüssel, der einer Backing-Key-ID zugeordnet ist

Alle CloudTrail Protokolleinträge für kryptografische Operationen mit einem KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher enthalten ein `additionalEventData` Feld mit dem `customKeyStoreId` und`backingKeyId`. Der im `backingKeyId` Feld zurückgegebene Wert korreliert mit dem CloudHSM-Schlüsselattribut. `id` Sie können den Schlüssellistenvorgang nach dem `id` Attribut filtern, um den KMS-Schlüssel zu identifizieren, der einem bestimmten Schlüssel zugeordnet ist. `backingKeyId`

1. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher, falls er nicht bereits getrennt ist, und melden Sie sich dann an als`kmsuser`, wie unter beschriebenTrennen und Anmelden.
2. Verwenden Sie den Befehl key list in der CloudHSM-CLI mit dem Attributfilter, um den Schlüssel in Ihrem Cluster zu finden, der als Schlüsselmaterial für einen bestimmten KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher dient.

   Das folgende Beispiel zeigt, wie Sie nach dem `id` Attribut filtern. AWS CloudHSM erkennt den `id` Wert als Hexadezimalwert. Um den Schlüssellistenvorgang nach dem `id` Attribut zu filtern, müssen Sie zuerst den `backingKeyId` Wert, den Sie in Ihrem CloudTrail Protokolleintrag angegeben haben, in ein Format konvertieren, das AWS CloudHSM ihn erkennt.

   a. Verwenden Sie den folgenden Linux-Befehl, um die `backingKeyId` in eine hexadezimale Darstellung zu konvertieren.

   ```
   echo backingKeyId | tr -d '\n' |  xxd -p
   ```

   Das folgende Beispiel zeigt, wie das `backingKeyId` Byte-Array in eine hexadezimale Darstellung konvertiert wird.

   ```
   echo 5890723622dc15f699aa9ab2387a9f744b2b884c18b2186ee8ada4f556a2eb9d | tr -d
    '\n' |  xxd -p
   353839303732333632326463313566363939616139616232333837613966373434623262383834633138623
   ```

   b. Stellen Sie der hexadezimalen Darstellung von mit voran. `backingKeyId` `0x`

   ```
   0x353839303732333632326463313566363939616139616232333837613966373434623262383834633138
   ```

   c. Verwenden Sie den konvertierten `backingKeyId` Wert, um nach dem Attribut zu filtern. `id` Geben Sie das `verbose` Argument an, um alle Attribute und Schlüsselinformationen für

den übereinstimmenden Schlüssel einzubeziehen. Wenn Sie das `verbose` Argument nicht angeben, gibt die Schlüssellistenoperation nur die Schlüsselreferenz und das Labelattribut des übereinstimmenden Schlüssels zurück.

```
aws-cloudhsm > key list --filter
 attr.id="0x353839303732333632326463313566363939616139616232333837613966373434623262383
 --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000120034",
        "key-info": {
          "key-owners": [
            {
              "username": "kmsuser",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "id":
 "0x353839303732333632326463313566363939616139616232333837613966373434623262383834633 13
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": false,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
```

```
            "sign": false,
            "trusted": false,
            "unwrap": true,
            "verify": false,
            "wrap": true,
            "wrap-with-trusted": false,
            "key-length-bytes": 32
          }
        }
      ],
      "total_key_count": 1,
      "returned_key_count": 1
    }
}
```

3. Melden Sie sich ab und stellen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher wieder her, wie unter beschrieben. Abmelden und erneutes Verbinden

Suchen Sie den AWS CloudHSM Schlüssel für einen KMS-Schlüssel

Sie können die KMS-Schlüssel-ID eines KMS-Schlüssels in einem AWS CloudHSM Schlüsselspeicher verwenden, um den Schlüssel in Ihrem AWS CloudHSM Cluster zu identifizieren, der als Schlüsselmaterial dient.

Wenn das Schlüsselmaterial für einen KMS-Schlüssel in Ihrem AWS CloudHSM Cluster AWS KMS erstellt wird, schreibt es den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels in die Schlüsselbezeichnung. Sofern Sie den Labelwert nicht geändert haben, können Sie den Befehl key list in der CloudHSM-CLI verwenden, um die Schlüsselressource und die ID des Schlüsselmaterials für den KMS-Schlüssel zu finden.

Alle CloudTrail Protokolleinträge für kryptografische Operationen mit einem KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher enthalten ein `additionalEventData` Feld mit dem und. `customKeyStoreId` `backingKeyId` Der im `backingKeyId` Feld zurückgegebene Wert ist das `id` AWS CloudHSM Schlüsselattribut. Sie können den AWS CloudHSM CLI-Vorgang mit der Schlüsselliste nach dem KMS-Schlüssel-ARN filtern, um das `id` CloudHSM-Schlüsselattribut zu identifizieren, das einem bestimmten KMS-Schlüssel zugeordnet ist.

Um dieses Verfahren auszuführen, müssen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher vorübergehend trennen, damit Sie sich als CU anmelden können. `kmsuser`

> **Hinweise**
>
> Die folgenden Verfahren verwenden das AWS CloudHSM Client SDK 5-Befehlszeilentool CloudHSM CLI. Die CloudHSM-CLI ersetzt durch`key-handle`. `key-reference`
> Am 1. Januar 2025 AWS CloudHSM wird der Support für die Client SDK 3-Befehlszeilentools, das CloudHSM Management Utility (CMU) und das Key Management Utility (KMU) eingestellt. Weitere Informationen zu den Unterschieden zwischen den Client SDK 3-Befehlszeilentools und dem Client SDK 5-Befehlszeilentool finden Sie unter Migrieren von Client SDK 3 CMU und KMU zu Client SDK 5 CloudHSM CLI im Benutzerhandbuch.AWS CloudHSM

1. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher, falls er nicht bereits getrennt ist, und melden Sie sich dann an als, wie `kmsuser` unter beschrieben. Trennen und Anmelden

   > **Note**
   >
   > Sämtliche Versuche, KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zu erstellen oder vorhandene KMS-Schlüssel in kryptografischen Produktionen zu nutzen, schlagen fehl, während der benutzerdefinierte Schlüsselspeicher getrennt ist. Diese Aktion kann verhindern, dass Benutzer vertrauliche Daten speichern und darauf zugreifen.

2. Verwenden Sie den Befehl key list in der CloudHSM-CLI und filtern Sie `label` nach, um den KMS-Schlüssel für einen bestimmten Schlüssel in Ihrem AWS CloudHSM Cluster zu finden. Geben Sie das `verbose` Argument an, das alle Attribute und Schlüsselinformationen für den übereinstimmenden Schlüssel enthalten soll. Wenn Sie das `verbose` Argument nicht angeben, gibt die Schlüssellistenoperation nur die Schlüsselreferenz - und Labelattribute des übereinstimmenden Schlüssels zurück.

   Das folgende Beispiel zeigt, wie nach dem `label` Attribut gefiltert wird, das den KMS-Schlüssel ARN speichert. Bevor Sie diesen Befehl ausführen, ersetzen Sie den Beispiel-KMS-Schlüssel-ARN durch einen gültigen aus Ihrem Konto.

   ```
   aws-cloudhsm > key list --filter attr.label="arn:aws:kms:us-
   west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" --verbose
   {
   ```

```
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000120034",
        "key-info": {
          "key-owners": [
            {
              "username": "kmsuser",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "id": "0xbacking-key-id",
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": false,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": false,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
```

```
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. Melden Sie sich ab und stellen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher wieder her, wie unter beschriebenAbmelden und erneutes Verbinden.

Tasten aktivieren und deaktivieren

Sie können kundenverwaltete Schlüssel aktivieren und deaktivieren. Beim Erstellen eines KMS-Schlüssels ist er standardmäßig aktiviert. Wenn Sie einen KMS-Schlüssel deaktivieren, kann er in keiner kryptografischen Produktion verwendet werden, bis Sie die Option erneut aktivieren.

Da es vorübergehend ist und leicht rückgängig gemacht werden kann, ist das Deaktivieren eines KMS-Schlüssels eine sichere Alternative zum Löschen eines KMS-Schlüssels, einer Aktion, die endgültig und irreversibel ist. Wenn Sie erwägen, einen KMS-Schlüssel zu löschen, deaktivieren Sie ihn zunächst und stellen Sie einen CloudWatch Alarm oder einen ähnlichen Mechanismus ein, um sicherzustellen, dass Sie den Schlüssel niemals zum Entschlüsseln verschlüsselter Daten verwenden müssen.

Wenn Sie einen KMS-Schlüssel deaktivieren, wird er sofort unbrauchbar (je nach letztendlicher Konsistenz). Ressourcen, die mit durch den KMS-Schlüssel geschützten Datenschlüsseln verschlüsselt wurden, sind jedoch nicht betroffen, bis der KMS-Schlüssel erneut verwendet wird, z. B. zur Entschlüsselung des Datenschlüssels. Dieses Problem betrifft AWS-Services, dass viele von ihnen Datenschlüssel verwenden, um Ihre Ressourcen zu schützen. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

Sie können oder nicht aktivieren Von AWS verwaltete Schlüsseloder deaktivieren AWS-eigene Schlüssel. Von AWS verwaltete Schlüssel sind dauerhaft für die Nutzung durch Dienste aktiviert, die AWS KMS AWS-eigene Schlüssel werden ausschließlich von dem Dienst verwaltet, dem sie gehören.

> **Note**
>
> AWS KMS rotiert das Schlüsselmaterial von vom Kunden verwalteten Schlüsseln nicht, solange diese deaktiviert sind. Weitere Informationen finden Sie unter So funktioniert die Schlüsselrotation.

Verwenden der AWS KMS Konsole

Sie können die AWS KMS Konsole verwenden, um vom Kunden verwaltete Schlüssel zu aktivieren und zu deaktivieren.

1. Melden Sie sich bei der Konsole AWS Key Management Service (AWS KMS) an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Aktivieren Sie das Kontrollkästchen für die KMS-Schlüssel, die Sie aktivieren oder deaktivieren möchten.
5. Um einen KMS-Schlüssel zu aktivieren, wählen Sie Key actions (Schlüsselaktionen) und Enable (aktivieren) aus. Um einen KMS-Schlüssel zu deaktivieren, wählen Sie Key actions (Schlüsselaktionen) und Disable (deaktivieren) aus.

Verwenden der API AWS KMS

Die EnableKeyOperation aktiviert eine deaktivierte AWS KMS key. Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen. Der Parameter `key-id` muss angegeben werden.

Diese Produktion gibt keine Ausgabe zurück. Verwenden Sie die DescribeKeyOperation, um den Schlüsselstatus zu sehen.

```
$ aws kms enable-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Der DisableKeyVorgang deaktiviert einen aktivierten KMS-Schlüssel. Der Parameter `key-id` muss angegeben werden.

```
$ aws kms disable-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Diese Produktion gibt keine Ausgabe zurück. Um den Schlüsselstatus zu sehen, verwenden Sie den DescribeKeyVorgang und sehen Sie sich das `Enabled` Feld an.

```
$ aws kms describe-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyMetadata": {
        "Origin": "AWS_KMS",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Description": "",
        "KeyManager": "CUSTOMER",
```

```
        "MultiRegion": false,
        "Enabled": false,
        "KeyState": "Disabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "CreationDate": 1502910355.475,
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333"
        "KeySpec": "SYMMETRIC_DEFAULT",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

Rotieren AWS KMS keys

Um neue kryptografische Daten für Ihre kundenverwalteten Schlüssel zu erstellen, können Sie neue KMS-Schlüssel erstellen und anschließend Ihre Anwendungen oder Aliasse so ändern, dass diese die neuen KMS-Schlüssel verwenden. Oder Sie können das Schlüsselmaterial, das einem vorhandenen KMS-Schlüssel zugeordnet ist, rotieren, indem Sie die automatische Schlüsselrotation aktivieren oder eine Rotation bei Bedarf durchführen.

Wenn Sie die automatische Schlüsselrotation für einen KMS-Schlüssel aktivieren, AWS KMS wird standardmäßig jedes Jahr neues kryptografisches Material für den KMS-Schlüssel generiert. Sie können auch einen benutzerdefinierten Wert angebenrotation-period, um die Anzahl der Tage nach der Aktivierung der automatischen Schlüsselrotation zu definieren, bei der das Schlüsselmaterial rotiert AWS KMS wird, sowie die Anzahl der Tage zwischen den einzelnen automatischen Rotationen danach. Wenn Sie die Schlüsselrotation sofort einleiten müssen, können Sie die Rotation bei Bedarf durchführen, unabhängig davon, ob die automatische Schlüsselrotation aktiviert ist oder nicht. Rotationen auf Anforderung ändern keine bestehenden Zeitpläne für die automatische Rotation.

Sie können die Rotation des Schlüsselmaterials für Ihre KMS-Schlüssel in Amazon CloudWatch und in der AWS Key Management Service Konsole verfolgen. AWS CloudTrail Mithilfe von GetKeyRotationStatusOperation können Sie auch überprüfen, ob die automatische Rotation für einen KMS-Schlüssel aktiviert ist, und alle laufenden On-Demand-Rotationen identifizieren. Mithilfe von ListKeyRotationsOperation können Sie sich die Details abgeschlossener Rotationen anzeigen lassen.

Bei der Schlüsselrotation wird nur das aktuelle Schlüsselmaterial geändert, bei dem es sich um das kryptografische Geheimnis handelt, das bei Verschlüsselungsvorgängen verwendet wird. Wenn Sie den gedrehten KMS-Schlüssel zum Entschlüsseln von Chiffretext verwenden, AWS KMS verwendet er das Schlüsselmaterial, mit dem er verschlüsselt wurde. Sie können kein bestimmtes Schlüsselmaterial für Entschlüsselungsvorgänge auswählen, es AWS KMS wird automatisch das richtige Schlüsselmaterial ausgewählt. Da mit dem entsprechenden Schlüsselmaterial AWS KMS transparent entschlüsselt wird, können Sie einen rotierten KMS-Schlüssel problemlos in Anwendungen und AWS-Services ohne Codeänderungen verwenden.

Der KMS-Schlüssel ist dieselbe logische Ressource, unabhängig davon, ob oder wie oft das Schlüsselmaterial geändert wird. Die Eigenschaften des KMS-Schlüssels werden nicht geändert, wie in der folgenden Abbildung dargestellt.

Möglicherweise möchten Sie einen neuen KMS-Schlüssel erstellen und anstelle des ursprünglichen KMS-Schlüssels verwenden. Dies hat den gleichen Effekt wie das Drehen des Schlüsselmaterials in einem vorhandenen KMS-Schlüssel, sodass es häufig als manuelle Drehung des Schlüssels betrachtet wird. Die manuelle Rotation ist eine gute Wahl, wenn Sie KMS-Schlüssel rotieren möchten, die nicht für eine automatische oder bedarfsgesteuerte Schlüsselrotation in Frage kommen. Dazu gehören asymmetrische KMS-Schlüssel, HMAC-KMS-Schlüssel, KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern und KMS-Schlüssel für mehrere Regionen mit importiertem Schlüsselmaterial.

> **Note**
>
> Die Schlüsselrotation hat keine Auswirkung auf die Daten, die der KMS-Schlüssel schützt. Die Datenschlüssel, die der KMS-Schlüssel generiert hat, werden nicht rotiert, und es werden keine durch den KMS-Schlüssel geschützten Daten erneut verschlüsselt. Durch die Schlüsselrotation werden die Auswirkungen eines kompromittierten Datenschlüssels nicht gemildert.

Schlüsselrotation und Preise

AWS KMS erhebt eine monatliche Gebühr für die erste und zweite Rotation des für Ihren KMS-Schlüssel verwalteten Schlüsselmaterials. Diese Preiserhöhung ist bei der zweiten Rotation begrenzt, und alle nachfolgenden Rotationen werden nicht in Rechnung gestellt. Für Einzelheiten vgl. AWS Key Management Service -Preise.

> **Note**
>
> Sie können den AWS Cost Explorer Service verwenden, um eine Aufschlüsselung Ihrer Gebühren für die Aufbewahrung Ihrer Schlüssel einzusehen. Sie können Ihre Ansicht beispielsweise so filtern, dass die Gesamtkosten für Schlüssel angezeigt werden, die als aktuelle und rotierte KMS-Schlüssel abgerechnet werden, indem Sie den `$REGION-KMS-Keys` als den Nutzungstyp angeben und die Daten nach API-Vorgang gruppieren. Möglicherweise werden Ihnen immer noch Instances des älteren `Unknown`-API-Vorgangs für historische Daten angezeigt.

Schlüsselrotation und Kontingente

Jeder KMS-Schlüssel zählt als Schlüssel bei der Berechnung wichtiger Ressourcenkontingente, unabhängig von der Anzahl der gedrehten Schlüsselmaterialversionen.

Weitere Informationen zu Schlüsselmaterial und Drehung finden Sie unter AWS Key Management Service Kryptografische Details.

Themen

- Warum sollten KMS-Schlüssel rotiert werden?
- So funktioniert die Schlüsselrotation
- Automatische Schlüsselrotation aktivieren
- Deaktivieren Sie die automatische Schlüsselrotation
- Führen Sie die Schlüsselrotation bei Bedarf durch
- Rotationen und wichtige Materialien auflisten
- Schlüssel manuell drehen
- Ändern Sie den Primärschlüssel in einer Reihe von Schlüsseln für mehrere Regionen

Warum sollten KMS-Schlüssel rotiert werden?

Bewährte kryptografische Verfahren verhindern die umfassende Wiederverwendung von Schlüsseln, die Daten direkt verschlüsseln, wie z. B. generierte Datenschlüssel. AWS KMS Wenn 256-Bit-Datenschlüssel Millionen von Nachrichten verschlüsseln, können sie erschöpft sein und anfangen, Geheimtext mit subtilen Mustern zu erzeugen, den clevere Akteure ausnutzen können, um die Bits im Schlüssel zu entdecken. Es empfiehlt sich, Datenschlüssel einmal oder nur ein paar Mal zu verwenden, um diese Schlüsselerschöpfung zu verringern.

KMS-Schlüssel werden jedoch am häufigsten als Mantelschlüssel verwendet, die auch als Schlüssel zur Schlüsselverschlüsselung bezeichnet werden. Anstatt Daten zu verschlüsseln, verschlüsseln Mantelschlüssel die Datenschlüssel, die Ihre Daten verschlüsseln. Daher werden sie weitaus seltener verwendet als Datenschlüssel und werden fast nie so oft wiederverwendet, dass das Risiko besteht, dass die Schlüssel erschöpft werden.

Trotz dieses sehr geringen Erschöpfungsrisikos müssen Sie Ihre KMS-Schlüssel möglicherweise aufgrund von Geschäfts- oder Vertragsregeln oder behördlichen Vorschriften rotieren. Wenn Sie gezwungen sind, KMS-Schlüssel zu rotieren, empfehlen wir, die automatische Schlüsselrotation zu verwenden, sofern sie unterstützt wird, die Drehung auf Anforderung zu verwenden, wenn die

automatische Rotation nicht unterstützt wird, und die manuelle Schlüsselrotation, wenn weder automatische noch On-Demand-Schlüsselrotation unterstützt wird.

Sie könnten erwägen, Rotationen auf Anfrage durchzuführen, um wichtige Funktionen der Materialrotation zu demonstrieren oder Automatisierungsskripte zu validieren. Wir empfehlen, für ungeplante Rotationen On-Demand-Rotationen und, wann immer möglich, die automatische Schlüsselrotation mit einer benutzerdefinierten Rotationsperiode zu verwenden.

So funktioniert die Schlüsselrotation

AWS KMS Die Schlüsselrotation ist so konzipiert, dass sie transparent und einfach zu bedienen ist. AWS KMS unterstützt die optionale automatische und bedarfsgesteuerte Schlüsselrotation nur für vom Kunden verwaltete Schlüssel.

Automatische Schlüsselrotation

AWS KMS rotiert den KMS-Schlüssel automatisch am nächsten Rotationsdatum, das durch Ihren Rotationsperiode definiert ist. Sie müssen das Update nicht selbst planen.

Die automatische Schlüsselrotation wird nur bei KMS-Schlüsseln mit symmetrischer Verschlüsselung unterstützt, deren Schlüsselmaterial AWS KMS generiert wird (`AWS_KMS`Herkunft).

Die automatische Rotation ist für vom Kunden verwaltete KMS-Schlüssel optional. AWS KMS wechselt das Schlüsselmaterial für AWS verwaltete KMS-Schlüssel jedes Jahr. Die Rotation der AWS eigenen KMS-Schlüssel wird von demjenigen verwaltet AWS-Service , dem der Schlüssel gehört.

Rotation auf Anfrage

Initiieren Sie sofort die Rotation des mit Ihrem KMS-Schlüssel verknüpften Schlüsselmaterials, unabhängig davon, ob die automatische Schlüsselrotation aktiviert ist oder nicht.

Die On-Demand-Schlüsselrotation wird für KMS-Schlüssel mit symmetrischer Verschlüsselung unterstützt, bei denen Schlüsselmaterial AWS KMS generiert wird (`AWS_KMS`Herkunft), und KMS-Schlüssel mit symmetrischer Verschlüsselung für einzelne Regionen mit importiertem Schlüsselmaterial (`EXTERNAL`Ursprung).

Manuelle Rotation

Für die folgenden Typen von KMS-Schlüsseln wird weder die automatische noch die On-Demand-Schlüsselrotation unterstützt. Sie können diese KMS-Schlüssel jedoch manuell rotieren.

- Asymmetrische KMS-Schlüssel
- HMAC-KMS-Schlüssel
- KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern
- KMS-Schlüssel für mehrere Regionen mit importiertem Schlüsselmaterial

Verwalten von Schlüsselmaterial

AWS KMS behält das gesamte Schlüsselmaterial für einen KMS-Schlüssel mit `AWS_KMS` Ursprung bei, auch wenn die Schlüsselrotation deaktiviert ist. AWS KMS löscht Schlüsselmaterial nur, wenn Sie den KMS-Schlüssel löschen.

Sie verwalten das Schlüsselmaterial für symmetrische Verschlüsselungsschlüssel mit `EXTERNAL` Herkunft. Sie können jedes Schlüsselmaterial mithilfe dieses DeleteImportedKeyMaterialVorgangs löschen oder beim Import des Materials eine Ablaufzeit festlegen. Der KMS-Schlüssel wird unbrauchbar, sobald eines seiner Materialien abläuft oder gelöscht wird.

Verwenden von Schlüsselmaterial

Wenn Sie einen rotierten KMS-Schlüssel zum Verschlüsseln von Daten verwenden, AWS KMS wird das aktuelle Schlüsselmaterial verwendet. Wenn Sie mit dem KMS-Schlüssel Daten entschlüsseln, verwendet AWS KMS das Schlüsselmaterial, das zum Verschlüsseln verwendet wurde. Sie können keine bestimmte Version des Schlüsselmaterials für Entschlüsselungsvorgänge auswählen. Es AWS KMS wird automatisch die richtige Version ausgewählt.

Zeitraum der Rotation

Der Rotationszeitraum definiert die Anzahl der Tage nach der Aktivierung der automatischen Schlüsselrotation, wodurch Ihr Schlüsselmaterial rotiert AWS KMS wird, und die Anzahl der Tage zwischen den einzelnen automatischen Schlüsselrotationen danach. Wenn Sie keinen Wert für `RotationPeriodInDays` die Aktivierung der automatischen Schlüsselrotation angeben, ist der Standardwert 365 Tage.

Sie können den RotationPeriodInDays Bedingungsschlüssel kms: verwenden, um die Werte, die Prinzipale im Parameter angeben können, weiter einzuschränken. `RotationPeriodInDays`

Datum der Rotation

Das Rotationsdatum gibt das Datum an, an dem das aktuelle Schlüsselmaterial für einen KMS-Schlüssel entweder als Ergebnis einer automatischen (geplanten) Rotation oder einer Schlüsselrotation auf Anfrage aktualisiert wurde.

Rotationsdatum

AWS KMS rotiert den KMS-Schlüssel automatisch an dem durch Ihre Rotationsperiode definierten Rotationsdatum. Der Standardrotationszeitraum beträgt 365 Tage.

Kundenverwaltete Schlüssel

Da die automatische Schlüsselrotation bei vom Kunden verwalteten Schlüsseln optional ist und jederzeit aktiviert und deaktiviert werden kann, hängt das Rotationsdatum von dem Datum ab, an dem die Rotation zuletzt aktiviert wurde. Das Datum kann sich ändern, wenn Sie den Rotationszeitraum für einen Schlüssel ändern, für den Sie zuvor die automatische Schlüsselrotation aktiviert haben. Das Rotationsdatum kann sich im Laufe der Lebensdauer des Schlüssels mehrfach ändern.

Wenn Sie beispielsweise am 1. Januar 2022 einen vom Kunden verwalteten Schlüssel erstellen und die automatische Schlüsselrotation mit dem AWS KMS Standardrotationszeitraum von 365 Tagen am 15. März 2022 aktivieren, wird das Schlüsselmaterial am 15. März 2023, 15. März 2024 und danach alle 365 Tage rotiert.

In den folgenden Beispielen wird davon ausgegangen, dass die automatische Schlüsselrotation mit dem Standardrotationszeitraum von 365 Tagen aktiviert wurde. Diese Beispiele zeigen Sonderfälle, die sich auf die Rotationsperiode eines Schlüssels auswirken können.

- Schlüsselrotation deaktivieren – Wenn Sie zu einem beliebigen Zeitpunkt die automatische Schlüsselrotation deaktivieren, verwendet der KMS-Schlüssel weiterhin die Version des Schlüsselmaterials, das er verwendet hat, als die Rotation deaktiviert wurde. Wenn Sie die automatische Schlüsselrotation wieder aktivieren, AWS KMS rotiert das Schlüsselmaterial auf der Grundlage des neuen Aktivierungsdatums für die Rotation.
- Deaktivierte KMS-Schlüssel — Solange ein KMS-Schlüssel deaktiviert ist, wird AWS KMS er nicht rotiert. Der Status der Schlüsseldrehung ändert sich jedoch nicht, und Sie können ihn nicht ändern, solange der KMS-Schlüssel deaktiviert ist. Wenn der KMS-Schlüssel wieder aktiviert wird und das Schlüsselmaterial sein letztes geplantes Rotationsdatum überschritten hat, wird es AWS KMS sofort rotiert. Wenn das Schlüsselmaterial seinen letzten geplanten Rotationstermin nicht verpasst hat, AWS KMS wird der ursprüngliche Zeitplan für die Schlüsselrotation wieder aufgenommen.
- KMS-Schlüssel, deren Löschung noch aussteht — Solange ein KMS-Schlüssel noch gelöscht werden AWS KMS muss, wird er nicht rotiert. Als Status der Schlüsselrotation wird `false` eingestellt und er kann von Ihnen nicht mehr geändert werden, während der Löschvorgang ansteht. Wenn der Löschvorgang abgebrochen wird, wird der vorherige

Status der Schlüsselrotation wiederhergestellt. Wenn das Schlüsselmaterial seinen letzten geplanten Rotationstermin überschritten hat, wird es sofort AWS KMS rotiert. Wenn das Schlüsselmaterial seinen letzten geplanten Rotationstermin nicht verpasst hat, AWS KMS wird der ursprüngliche Zeitplan für die Schlüsselrotation wieder aufgenommen.

Von AWS verwaltete Schlüssel

AWS KMS wechselt automatisch Von AWS verwaltete Schlüssel jedes Jahr (ungefähr 365 Tage). Sie können die Schlüsseldrehung von Von AWS verwaltete Schlüssel nicht aktivieren oder deaktivieren.

Das Schlüsselmaterial für eine Von AWS verwalteter Schlüssel wird zunächst ein Jahr nach dem Erstellungsdatum rotiert und danach jedes Jahr (ungefähr 365 Tage nach der letzten Rotation).

> **Note**
>
> Im Mai 2022 AWS KMS wurde der Rotationsplan Von AWS verwaltete Schlüssel von allen drei Jahren (etwa 1.095 Tage) auf jedes Jahr (ungefähr 365 Tage) geändert.

AWS-eigene Schlüssel

Sie können die Schlüsseldrehung von AWS-eigene Schlüssel nicht aktivieren oder deaktivieren. Die Strategie der Schlüsselrotation für eine AWS-eigener Schlüssel wird durch den AWS Dienst bestimmt, der den Schlüssel erstellt und verwaltet. Weitere Informationen finden Sie im Benutzerhandbuch oder Entwicklerhandbuch für den Service unter dem Thema Verschlüsselung im Ruhezustand.

Rotierende Schlüssel für mehrere Regionen

Sie können die automatische Rotation aktivieren und deaktivieren und bei Bedarf eine Rotation des Schlüsselmaterials bei symmetrischer Verschlüsselung von Schlüsseln aus mehreren Regionen mit Ursprung durchführen. `AWS_KMS` Die Schlüsselrotation ist eine gemeinsame Eigenschaft von Schlüsseln mit mehreren Regionen.

Sie können die automatische Schlüsseldrehung nur für einen Primärschlüssel aktivieren oder deaktivieren. Sie initiieren die Rotation bei Bedarf nur für den Primärschlüssel.

- Bei der AWS KMS Synchronisierung der Schlüssel für mehrere Regionen wird die Eigenschaftseinstellung für die Schlüsselrotation vom Primärschlüssel auf alle zugehörigen Replikatschlüssel kopiert.

- Bei der AWS KMS Rotation des Schlüsselmaterials wird neues Schlüsselmaterial für den Primärschlüssel erstellt. Anschließend wird das neue Schlüsselmaterial über Regionsgrenzen hinweg in alle zugehörigen Replikatschlüssel kopiert. Das Schlüsselmaterial wird niemals unverschlüsselt verlassen AWS KMS . Dieser Schritt wird sorgfältig überwacht, um sicherzustellen, dass das Schlüsselmaterial vollständig synchronisiert wird, bevor ein Schlüssel in einer kryptografischen Operation verwendet wird.
- AWS KMS verschlüsselt keine Daten mit dem neuen Schlüsselmaterial, bis dieses Schlüsselmaterial im Primärschlüssel und in jedem seiner Replikatschlüssel verfügbar ist.
- Wenn Sie einen gedrehten Primärschlüssel replizieren, enthält der neue Replikatschlüssel das aktuelle Schlüsselmaterial und alle früheren Versionen des Schlüsselmaterials für seine verwandten multiregionalen Schlüssel.

Dieses Muster stellt sicher, dass verwandte multiregionale Schlüssel vollständig interoperabel sind. Jeder multiregionale Schlüssel kann Chiffretext entschlüsseln, der mit einem verwandten multiregionalen Schlüssel verschlüsselt wurde, selbst wenn der Chiffretext vor dem Erstellen des Schlüssels verschlüsselt wurde.

AWS Dienste

Sie können für die kundenverwalteten KMS-Schlüssel, die Sie für die serverseitige Verschlüsselung in AWS -Services verwenden, eine automatische Schlüsseldrehung aktivieren. Die jährliche Drehung ist transparent und mit AWS -Services kompatibel.

Überwachen der Schlüsselrotation.

Wenn das Schlüsselmaterial für einen Von AWS verwalteter Schlüsseloder vom Kunden verwalteten Schlüssel AWS KMS rotiert wird, schreibt es ein `KMS CMK Rotation` Ereignis an Amazon EventBridge und ein RotateKey Ereignis in Ihr AWS CloudTrail Protokoll. Sie können die diese Datensätze verwenden, um zu überprüfen, ob der KMS-Schlüssel gedreht wurde.

In der AWS Key Management Service Konsole können Sie sich die Anzahl der verbleibenden On-Demand-Rotationen und eine Liste aller abgeschlossenen Schlüsselmaterialrotationen für einen KMS-Schlüssel anzeigen lassen.

Sie können den ListKeyRotationsVorgang verwenden, um die Details der abgeschlossenen Rotationen einzusehen.

Letztendliche Datenkonsistenz

Die Schlüsselrotation unterliegt letztlich den gleichen Konsistenzeffekten wie andere AWS KMS Verwaltungsvorgänge. Es kann zu einer leichten Verzögerung kommen, bevor das

neue Schlüsselmaterial in allen Bereichen von AWS KMS verfügbar ist. Das Drehen von Schlüsselmaterial verursacht jedoch keine Unterbrechung oder Verzögerung kryptografischer Operationen. Das aktuelle Schlüsselmaterial wird in kryptografischen Operationen verwendet, bis das neue Schlüsselmaterial überall in AWS KMS verfügbar ist. Wenn das Schlüsselmaterial für einen Schlüssel mit mehreren Regionen automatisch rotiert wird, wird das aktuelle Schlüsselmaterial AWS KMS verwendet, bis das neue Schlüsselmaterial in allen Regionen mit einem zugehörigen Schlüssel für mehrere Regionen verfügbar ist.

Automatische Schlüsselrotation aktivieren

Wenn Sie die automatische Schlüsselrotation für einen KMS-Schlüssel aktivieren, AWS KMS wird standardmäßig jedes Jahr neues kryptografisches Material für den KMS-Schlüssel generiert. Sie können auch einen benutzerdefinierten Wert angebenrotation-period, um die Anzahl der Tage nach der Aktivierung der automatischen Schlüsselrotation zu definieren, bei der das Schlüsselmaterial rotiert AWS KMS wird, sowie die Anzahl der Tage zwischen den einzelnen automatischen Rotationen danach.

Die automatische Schlüsselrotation bietet folgende Vorteile:

- Die Eigenschaften des KMS-Schlüssels, einschließlich Schlüssel-ID, Schlüssel-ARN, Region, Richtlinien und Berechtigungen, werden bei der Drehung des Schlüssels nicht geändert.
- Sie müssen keine Anwendungen oder Aliasse ändern, die sich auf die KMS-Schlüssel-ID oder den ARN beziehen.
- Drehendes Schlüsselmaterial beeinträchtigt die Verwendung des KMS-Schlüssels in keinem AWS-Service.
- Nachdem Sie die Schlüsselrotation aktiviert haben, wird der KMS-Schlüssel automatisch am nächsten Rotationsdatum AWS KMS rotiert, das durch Ihre Rotationsperiode definiert wird. Sie müssen das Update nicht selbst planen.

Sie können die automatische Schlüsselrotation in der AWS KMS Konsole oder mithilfe des EnableKeyRotationVorgangs aktivieren. Um die automatische Schlüsselrotation zu aktivieren, benötigen Sie `kms:EnableKeyRotation` Berechtigungen. Weitere Informationen zu AWS KMS Berechtigungen finden Sie unterBerechtigungsreferenz.

Verwenden der AWS KMS Konsole

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. (Sie können die Drehung von Von AWS verwaltete Schlüssel nicht aktiviern oder deaktivieren. Diese werden automatisch jedes Jahr gedreht.)
4. Wählen Sie den Alias oder die Schlüssel-ID eines KMS-Schlüssels.
5. Wählen Sie die Registerkarte Key rotation (Schlüsselrotation).

   Die Registerkarte Schlüsselrotation wird nur auf der Detailseite von KMS-Schlüsseln mit symmetrischer Verschlüsselung angezeigt, wobei das Schlüsselmaterial AWS KMS generiert wurde (der Ursprung ist AWS_KMS), einschließlich KMS-Schlüsseln mit symmetrischer Verschlüsselung für mehrere Regionen.

   Asymmetrische KMS-Schlüssel, HMAC-KMS-Schlüssel, KMS-Schlüssel mit importiertem Schlüsselmaterial oder KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern können nicht automatisch gedreht werden. Sie können sie jedoch manuell rotieren.
6. Wählen Sie im Abschnitt Automatische Schlüsselrotation die Option Bearbeiten aus.
7. Wählen Sie für Schlüsselrotation die Option Aktivieren aus.

   > **Note**
   >
   > Wenn ein KMS-Schlüssel deaktiviert ist oder noch gelöscht werden AWS KMS muss, wird das Schlüsselmaterial nicht rotiert, und Sie können den Status oder den Rotationszeitraum der automatischen Schlüsselrotation nicht aktualisieren. Aktivieren Sie den KMS-Schlüssel oder brechen Sie den Löschvorgang ab, um die Konfiguration der automatischen Schlüsselrotation zu aktualisieren. Details dazu finden Sie unter So funktioniert die Schlüsselrotation und Wichtige Zustände von AWS KMS Schlüsseln.
8. (Optional) Geben Sie einen Rotationszeitraum zwischen 90 und 2560 Tagen ein. Der Standardwert ist 365 Tage. Wenn Sie keinen benutzerdefinierten Rotationszeitraum angeben, AWS KMS wird das Schlüsselmaterial jedes Jahr rotiert.

Sie können den RotationPeriodInDays Bedingungsschlüssel kms: verwenden, um die Werte einzuschränken, die Prinzipale für den Rotationsperiode angeben können.

9. Wählen Sie Speichern.

Verwenden der API AWS KMS

Sie können die API AWS Key Management Service (AWS KMS) verwenden, um die automatische Schlüsselrotation zu aktivieren und den aktuellen Rotationsstatus aller vom Kunden verwalteten Schlüssel einzusehen. Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Der EnableKeyRotationVorgang ermöglicht die automatische Schlüsselrotation für den angegebenen KMS-Schlüssel. Um den KMS-Schlüssel bei diesem Vorgang zu identifizieren, verwenden Sie seine Schlüssel-ID oder seinen Schlüssel-ARN. Standardmäßig ist die Schlüsseldrehung für kundenverwaltete KMS-Schlüssel deaktiviert.

Sie können den kms:RotationPeriodInDaysBedingungsschlüssel verwenden, um die Werte einzuschränken, die Prinzipale für den `RotationPeriodInDays` Parameter einer `EnableKeyRotation` Anforderung angeben können.

Das folgende Beispiel aktiviert die Schlüsselrotation mit einer Rotationsperiode von 180 Tagen für den angegebenen KMS-Schlüssel mit symmetrischer Verschlüsselung und verwendet den GetKeyRotationStatusVorgang, um das Ergebnis zu sehen.

```
$ aws kms enable-key-rotation \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --rotation-period-in-days 180

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyRotationEnabled": true,
    "RotationPeriodInDays": 180,
    "NextRotationDate": "2024-02-14T18:14:33.587000+00:00"
}
```

Deaktivieren Sie die automatische Schlüsselrotation

Nachdem Sie die automatische Schlüsselrotation für einen vom Kunden verwalteten Schlüssel aktiviert haben, können Sie sie jederzeit deaktivieren.

Wenn Sie die automatische Schlüsselrotation deaktivieren, verwendet der KMS-Schlüssel weiterhin die Version des Schlüsselmaterials, die er bei der Deaktivierung der Schlüsselrotation verwendet hat. Wenn Sie die automatische Schlüsselrotation wieder aktivieren, wird das Schlüsselmaterial auf der Grundlage des neuen Aktivierungsdatums für die Rotation AWS KMS rotiert.

Die Deaktivierung der automatischen Rotation hat keinen Einfluss auf Ihre Fähigkeit, Rotationen auf Anfrage durchzuführen, und es werden auch keine laufenden On-Demand-Rotationen storniert.

Sie können die automatische Tastenrotation in der AWS KMS Konsole oder mithilfe des Vorgangs deaktivieren. DisableKeyRotation Um die automatische Schlüsselrotation zu deaktivieren, benötigen Sie `kms:DisableKeyRotation` Berechtigungen. Weitere Informationen zu AWS KMS Berechtigungen finden Sie unterBerechtigungsreferenz.

Verwenden der AWS KMS Konsole

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. (Sie können die Drehung von Von AWS verwaltete Schlüssel nicht aktiviern oder deaktivieren. Diese werden automatisch jedes Jahr gedreht.)
4. Wählen Sie den Alias oder die Schlüssel-ID eines KMS-Schlüssels.
5. Wählen Sie die Registerkarte Key rotation (Schlüsselrotation).

   Die Registerkarte Schlüsselrotation wird nur auf der Detailseite von KMS-Schlüsseln mit symmetrischer Verschlüsselung angezeigt, wobei das Schlüsselmaterial AWS KMS generiert wurde (der Ursprung ist AWS_KMS), einschließlich KMS-Schlüsseln mit symmetrischer Verschlüsselung für mehrere Regionen.

   Asymmetrische KMS-Schlüssel, HMAC-KMS-Schlüssel, KMS-Schlüssel mit importiertem Schlüsselmaterial oder KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern können nicht automatisch gedreht werden. Sie können sie jedoch manuell rotieren.

6. Wählen Sie im Abschnitt Automatische Schlüsselrotation die Option Bearbeiten aus.
7. Wählen Sie für Schlüsselrotation die Option Deaktivieren aus.

> **Note**
>
> Wenn ein KMS-Schlüssel deaktiviert ist oder noch gelöscht werden AWS KMS muss, wird das Schlüsselmaterial nicht rotiert, und Sie können den Status oder den Rotationszeitraum der automatischen Schlüsselrotation nicht aktualisieren. Aktivieren Sie den KMS-Schlüssel oder brechen Sie den Löschvorgang ab, um die Konfiguration der automatischen Schlüsselrotation zu aktualisieren. Details dazu finden Sie unter So funktioniert die Schlüsselrotation und Wichtige Zustände von AWS KMS Schlüsseln.

8. Wählen Sie Speichern.

Verwenden der AWS KMS API

Sie können die API AWS Key Management Service (AWS KMS) verwenden, um die automatische Schlüsselrotation zu deaktivieren und den aktuellen Rotationsstatus aller vom Kunden verwalteten Schlüssel einzusehen. In diesem Beispiel wird die AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jede unterstützte Programmiersprache verwenden.

Der DisableKeyRotationVorgang deaktiviert die automatische Schlüsselrotation. Um den KMS-Schlüssel bei diesem Vorgang zu identifizieren, verwenden Sie seine Schlüssel-ID oder seinen Schlüssel-ARN. Standardmäßig ist die Schlüsseldrehung für kundenverwaltete KMS-Schlüssel deaktiviert.

Im folgenden Beispiel wird die automatische Schlüsselrotation für den angegebenen KMS-Schlüssel mit symmetrischer Verschlüsselung deaktiviert und das Ergebnis anhand des GetKeyRotationStatusVorgangs angezeigt.

```
$ aws kms disable-key-rotation --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyRotationEnabled": false
}
```

Führen Sie die Schlüsselrotation bei Bedarf durch

Sie können das Schlüsselmaterial in vom Kunden verwalteten KMS-Schlüsseln bei Bedarf rotieren, unabhängig davon, ob die automatische Schlüsselrotation aktiviert ist oder nicht. Die Deaktivierung der automatischen Rotation (DisableKeyRotation) hat keinen Einfluss auf Ihre Fähigkeit, Rotationen auf Anfrage durchzuführen, und es werden auch keine laufenden On-Demand-Rotationen storniert. Rotationen auf Abruf haben keine Auswirkungen auf bestehende Zeitpläne für automatische Rotationen. Stellen Sie sich zum Beispiel einen KMS-Schlüssel vor, für den die automatische Schlüsselrotation aktiviert ist und für den eine Rotationsperiode von 730 Tagen gilt. Wenn für den Schlüssel eine automatische Rotation am 14. April 2024 geplant ist und Sie am 10. April 2024 eine On-Demand-Rotation durchführen, wird der Schlüssel wie geplant am 14. April 2024 und danach alle 730 Tage automatisch rotiert.

Sie können die Schlüsselrotation bei Bedarf maximal zehnmal pro KMS-Schlüssel durchführen. Sie können die AWS KMS Konsole verwenden, um die Anzahl der verbleibenden On-Demand-Rotationen anzuzeigen, die für einen KMS-Schlüssel verfügbar sind.

Die On-Demand-Schlüsselrotation wird nur für KMS-Schlüssel mit symmetrischer Verschlüsselung unterstützt. Sie können keine On-Demand-Rotation von asymmetrischen KMS-Schlüsseln, HMAC-KMS-Schlüsseln, KMS-Schlüsseln mit mehreren Regionen mit importiertem Schlüsselmaterial oder KMS-Schlüsseln in einem benutzerdefinierten Schlüsselspeicher durchführen. Rufen Sie die On-Demand-Rotation für einen Satz verwandter Schlüssel mit mehreren Regionen auf Anforderung für den Primärschlüssel auf.

Autorisierte Benutzer mit `kms:RotateKeyOnDemand` und `kms:GetKeyRotationStatus` Berechtigungen können die AWS KMS Konsole und die AWS KMS API verwenden, um eine On-Demand-Schlüsselrotation einzuleiten und den Status der Schlüsselrotation einzusehen. Wird verwendet ListKeyRotations, um abgeschlossene Rotationen für einen KMS-Schlüssel anzuzeigen.

Themen

- Initiierung der Schlüsselrotation bei Bedarf (Konsole)
- Initiierung der On-Demand-Schlüsselrotation (API)AWS KMS

Initiierung der Schlüsselrotation bei Bedarf (Konsole)

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.

2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. (Sie können die Rotation von nicht bei Bedarf durchführen. Von AWS verwaltete Schlüssel Sie werden jedes Jahr automatisch rotiert.)
4. Wählen Sie den Alias oder die Schlüssel-ID eines KMS-Schlüssels.
5. Wählen Sie die Registerkarte Schlüsselmaterial und Rotationen.

   Die Registerkarte Schlüsselmaterial und Rotationen wird nur auf der Detailseite von KMS-Schlüsseln mit symmetrischer Verschlüsselung angezeigt, die automatische Rotation oder Rotation bei Bedarf unterstützen. Dazu gehören KMS-Schlüssel mit AWS KMS generiertem Schlüsselmaterial (AWS_KMSUrsprung) und KMS-Schlüssel für einzelne Regionen mit importiertem Schlüsselmaterial (EXTERNER Ursprung).

   Sie können keine On-Demand-Rotation von asymmetrischen KMS-Schlüsseln, HMAC-KMS-Schlüsseln, KMS-Schlüsseln mit importiertem Schlüsselmaterial aus mehreren Regionen oder KMS-Schlüsseln in benutzerdefinierten Schlüsselspeichern durchführen. Sie können sie jedoch manuell rotieren.
6. Wählen Sie „Jetzt drehen“. Bei symmetrischen Verschlüsselungsschlüsseln mit nur einer Region und importiertem Schlüsselmaterial ist die Option „Jetzt drehen“ nur verfügbar, wenn Sie zuvor neues Schlüsselmaterial importiert haben und es sich im Rotationsstatus Ausstehend befindet.
7. Lesen und berücksichtigen Sie die Warnung und die Informationen über die Anzahl der verbleibenden On-Demand-Rotationen für den Schlüssel. Außerdem werden Ihnen Informationen wie die ID, die Beschreibung und die Ablaufzeit des Schlüsselmaterials angezeigt, die nach der Rotation aktuell werden. Wenn Sie entscheiden, dass Sie mit der On-Demand-Rotation nicht fortfahren möchten, wählen Sie Abbrechen.
8. Wählen Sie die Taste „Drehen“, um die Drehung bei Bedarf zu bestätigen.

> **Note**
>
> Die Rotation nach Bedarf unterliegt letztlich den gleichen Konsistenzeffekten wie andere AWS KMS Verwaltungsvorgänge. Es kann zu einer leichten Verzögerung kommen, bevor das neue Schlüsselmaterial in allen Bereichen von AWS KMS verfügbar ist. Das Banner oben in der Konsole informiert Sie, wenn die On-Demand-Rotation abgeschlossen ist.

Initiierung der On-Demand-Schlüsselrotation (API)AWS KMS

Sie können die API AWS Key Management Service (AWS KMS) verwenden, um eine On-Demand-Schlüsselrotation zu initiieren und den aktuellen Rotationsstatus aller vom Kunden verwalteten Schlüssel einzusehen. In diesem Beispiel wird die AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jede unterstützte Programmiersprache verwenden.

Der RotateKeyOnDemandVorgang initiiert sofort die On-Demand-Schlüsselrotation für den angegebenen KMS-Schlüssel. Um den KMS-Schlüssel bei diesen Operationen zu identifizieren, verwenden Sie seine Schlüssel-ID oder den Schlüssel-ARN.

Im folgenden Beispiel wird eine On-Demand-Schlüsselrotation für den angegebenen KMS-Schlüssel mit symmetrischer Verschlüsselung initiiert und anhand des GetKeyRotationStatusVorgangs überprüft, ob die On-Demand-Rotation ausgeführt wird. `OnDemandRotationStartDate`In der `kms:GetKeyRotationStatus` Antwort werden Datum und Uhrzeit angegeben, zu denen eine laufende On-Demand-Rotation initiiert wurde. In diesem Beispiel ist für den KMS-Schlüssel auch die automatische Rotation mit einem Zeitraum von 365 Tagen aktiviert.

```
$ aws kms rotate-key-on-demand --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyRotationEnabled": true,
    "NextRotationDate": "2024-03-14T18:14:33.587000+00:00",
    "OnDemandRotationStartDate": "2024-02-24T18:44:48.587000+00:00"
    "RotationPeriodInDays": 365
}
```

Wenn der KMS-Schlüssel keine automatische Rotation unterstützt oder die automatische Rotation nicht aktiviert ist, würde die `kms:GetKeyRotationStatus` Antwort weniger Felder enthalten, wie im folgenden Beispiel gezeigt:

```
$ aws kms rotate-key-on-demand --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

```
$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyRotationEnabled": false,
    "OnDemandRotationStartDate": "2024-02-24T18:44:48.587000+00:00"
}
```

Rotationen und wichtige Materialien auflisten

KMS-Schlüsseln, die die automatische oder bedarfsgesteuerte Schlüsselrotation unterstützen, können mehrere Schlüsselmaterialien zugeordnet sein. Diese Schlüssel haben ein anfängliches Schlüsselmaterial und ein zusätzliches Schlüsselmaterial für jede automatische oder bedarfsgesteuerte Rotation.

Autorisierte Benutzer mit entsprechender `kms:ListKeyRotations` Genehmigung können die AWS KMS Konsole und die ListKeyRotationsAPI verwenden, um alle wichtigen Materialien aufzulisten, die mit einem KMS-Schlüssel verknüpft sind, einschließlich der Materialien aus abgeschlossenen automatischen und On-Demand-Rotationen.

Themen

- Rotationen und wichtige Materialien auflisten (Konsole)
- Rotationen und Schlüsselmaterialien auflisten (API)AWS KMS

Rotationen und wichtige Materialien auflisten (Konsole)

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Wählen Sie den Alias oder die Schlüssel-ID eines KMS-Schlüssels.
5. Wählen Sie die Registerkarte Schlüsselmaterial und Rotationen.

   - Die Registerkarte Schlüsselmaterial und Rotationen wird nur auf der Detailseite von KMS-Schlüsseln mit symmetrischer Verschlüsselung angezeigt, die automatische Rotation oder Rotation bei Bedarf unterstützen. Dazu gehören KMS-Schlüssel mit AWS KMS generiertem

Schlüsselmaterial (`AWS_KMS`Ursprung) und KMS-Schlüssel für einzelne Regionen mit importiertem Schlüsselmaterial (`EXTERNAL`Ursprung).

- In der Tabelle Schlüsselmaterialien auf der Registerkarte Schlüsselmaterial und Rotationen sind alle Schlüsselmaterialien aufgeführt, die dem KMS-Schlüssel zugeordnet sind. Für jedes Schlüsselmaterial zeigt der entsprechende Eintrag die zugewiesene eindeutige Kennung AWS KMS, das Rotationsdatum und den Status des Schlüsselmaterials an. Das Rotationsdatum gibt an, wann das Schlüsselmaterial nach einer automatischen oder bedarfsgesteuerten Schlüsselrotation aktuell wurde. Dem ersten Material oder dem `Pending rotation` Schlüsselmaterial ist kein Rotationsdatum zugeordnet. Der Zustand des Schlüsselmaterials bestimmt, wie das Schlüsselmaterial AWS KMS verwendet wird. Das aktuelle Schlüsselmaterial wird sowohl für die Verschlüsselung als auch für die Entschlüsselung verwendet. Nicht aktuelles Schlüsselmaterial wird nur zur Entschlüsselung verwendet. Ein Schlüsselmaterialstatus von `Pending rotation` gibt an, dass das Schlüsselmaterial für die Rotation bereitgestellt wurde. Dieses Schlüsselmaterial wird erst dann für kryptografische Operationen verwendet, wenn es durch eine On-Demand-Schlüsselrotation zum aktuellen Schlüsselmaterial wird. Zusätzliche Informationen, die für das Schlüsselmaterial angezeigt werden, hängen vom Typ des KMS-Schlüssels ab.
- Bei KMS-Schlüsseln mit symmetrischer Verschlüsselung mit `AWS_KMS` Ursprung wird in jeder Zeile auch der Rotationstyp angezeigt — `On-demand` oder`Automatic`.
- KMS-Schlüssel mit symmetrischer Verschlüsselung für einzelne Regionen mit importiertem Schlüsselmaterial (`EXTERNAL`Ursprung) unterstützen nur die `On-demand` Rotation, sodass es keine Spalte mit dem Rotationstyp gibt. Stattdessen werden in jeder Zeile ein Importstatus, eine benutzerdefinierte Beschreibung, Ablaufinformationen und ein Aktionsmenü angezeigt. Der Importstatus ist entweder Importiert, was bedeutet, dass das Schlüsselmaterial innerhalb verfügbar ist, AWS KMS oder Ausstehender Import, was bedeutet, dass das Schlüsselmaterial nicht verfügbar ist. AWS KMS Das Aktionsmenü kann verwendet werden, um importiertes Schlüsselmaterial zu löschen oder Schlüsselmaterial erneut zu importieren. Die Aktion Schlüsselmaterial löschen ist deaktiviert, wenn der Importstatus des Schlüsselmaterials auf Ausstehender Import eingestellt ist. Die Aktion Schlüsselmaterial erneut importieren ist immer verfügbar. Sie müssen nicht warten, bis ein Schlüsselmaterial abläuft oder gelöscht wurde, bevor Sie es erneut importieren.

Rotationen und Schlüsselmaterialien auflisten (API)AWS KMS

Sie können die API AWS Key Management Service (AWS KMS) verwenden, um eine On-Demand-Schlüsselrotation zu initiieren und den aktuellen Rotationsstatus aller vom Kunden verwalteten

Schlüssel einzusehen. In diesem Beispiel wird die AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jede unterstützte Programmiersprache verwenden.

Der ListKeyRotationsVorgang listet alle Rotationen und Schlüsselmaterialien für den angegebenen KMS-Schlüssel auf. Um den KMS-Schlüssel bei diesen Operationen zu identifizieren, verwenden Sie seine Schlüssel-ID oder den Schlüssel-ARN.

Dieser Vorgang unterstützt einen optionalen `IncludeKeyMaterial` Parameter. Der Standardwert für diesen Parameter ist `ROTATIONS_ONLY`. Wenn Sie diesen Parameter weglassen, werden Informationen zu den wichtigsten Materialien AWS KMS zurückgegeben, die durch automatische oder bedarfsgesteuerte Schlüsselrotation erzeugt wurden. Wenn Sie einen Wert von angeben`ALL_KEY_MATERIAL`, werden das erste Schlüsselmaterial und alle importierten Schlüsselmaterialien, deren Rotation noch aussteht, zur Antwort AWS KMS hinzugefügt. Dieser Parameter kann nur mit KMS-Schlüsseln verwendet werden, die die automatische oder bedarfsgesteuerte Schlüsselrotation unterstützen.

```
$ aws kms list-key-rotations --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --inlcude-key-material ALL_KEY_MATERIAL
{
    "Rotations": [
        {
            "KeyId": 1234abcd-12ab-34cd-56ef-1234567890ab,
            "KeyMaterialId":
 123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0,
            "KeyMaterialDescription": "KeyMaterialA",
            "ImportState": "PENDING_IMPORT",
            "KeyMaterialState": "NON_CURRENT"
        },
        {
            "KeyId": 1234abcd-12ab-34cd-56ef-1234567890ab,
            "KeyMaterialId":
 96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068,
            "ImportState": "IMPORTED",
            "KeyMaterialState": "CURRENT",
            "ExpirationModel": "KEY_MATERIAL_DOES_NOT_EXPIRE",
            "RotationDate": "2025-05-01T15:50:51.045000-07:00",
            "RotationType": "ON_DEMAND"
        }
    ],
    "Truncated": false
}
```

Schlüssel manuell drehen

Möglicherweise möchten Sie einen neuen KMS-Schlüssel erstellen und ihn anstelle eines aktuellen KMS-Schlüssels verwenden, anstatt die automatische oder bedarfsgesteuerte Schlüsselrotation zu verwenden. Wenn der neue KMS-Schlüssel andere kryptografische Daten aufweist als der aktuelle KMS-Schlüssel, hat die Verwendung des neuen KMS-Schlüssels den gleichen Effekt wie die Änderung des Schlüsselmaterials in einem vorhandenen KMS-Schlüssel. Das Ersetzen eines KMS-Schlüssels durch einen anderen wird als manuelle Schlüsseldrehung bezeichnet.

Die manuelle Rotation ist eine gute Wahl, wenn Sie KMS-Schlüssel rotieren möchten, die nicht für eine automatische oder On-Demand-Schlüsselrotation in Frage kommen, z. B. asymmetrische KMS-Schlüssel, HMAC-KMS-Schlüssel, KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern und KMS-Schlüssel für mehrere Regionen mit importiertem Schlüsselmaterial.

> **Note**
>
> Wenn Sie mit der Verwendung des neuen KMS-Schlüssels beginnen, achten Sie darauf, dass der ursprüngliche KMS-Schlüssel aktiviert bleibt, damit Daten, die mit dem ursprünglichen KMS-Schlüssel verschlüsselt wurden, entschlüsselt werden AWS KMS können.

Wenn Sie KMS-Schlüssel manuell rotieren, müssen Sie auch die Verweise auf die ID oder den ARN des KMS-Schlüssels in Ihren Anwendungen aktualisieren. Dieses Verfahren lässt sich durch Aliasse, die einen Anzeigenamen mit einem KMS-Schlüssel verknüpfen, vereinfachen. Verwenden Sie einen Alias, um auf einen KMS-Schlüssel in Ihren Anwendungen zu verweisen. Wenn Sie dann den von der Anwendung verwendeten KMS-Schlüssel ändern möchten, anstatt den Anwendungscode zu bearbeiten, ändern Sie den Ziel-KMS-Schlüssel des Alias. Details hierzu finden Sie unter Erfahren Sie, wie Sie Aliase in Ihren Anwendungen verwenden.

> **Note**
>
> Aliase, die auf die neueste Version eines manuell rotierten KMS-Schlüssels verweisen DescribeKey, sind eine gute Lösung für kryptografische Operationen wie Verschlüsseln DeriveSharedSecret,,,, GenerateDataKeyGenerateDataKeyPairGenerateMacVerifyMac,

Signieren GetPublicKeyund Überprüfen. Aliase sind bei Vorgängen, die KMS-Schlüssel verwalten, nicht zulässig, wie z. B. oder. DisableKeyScheduleKeyDeletion
Wenn Sie den Vorgang Decrypt für manuell rotierte symmetrische KMS-Schlüssel aufrufen, lassen Sie den `KeyId` Parameter im Befehl weg. AWS KMS verwendet automatisch den KMS-Schlüssel, mit dem der Chiffretext verschlüsselt wurde.
Der `KeyId` Parameter ist erforderlich, wenn Sie mit einem asymmetrischen KMS-Schlüssel `Decrypt` oder Verify oder VerifyMacmit einem HMAC-KMS-Schlüssel anrufen. Diese Anfragen schlagen fehl, wenn der Wert vom `KeyId`-Parameter ein Alias ist, der nicht mehr auf den KMS-Schlüssel verweist, der die kryptografische Operation ausgeführt hat, z. B. wenn ein Schlüssel manuell gedreht wird. Um diesen Fehler zu vermeiden, müssen Sie den richtigen KMS-Schlüssel für jeden Vorgang verfolgen und angeben.

Um den KMS-Zielschlüssel eines Alias zu ändern, verwenden Sie UpdateAliasOperation in der AWS KMS API. Beispielsweise aktualisiert dieser Befehl den `alias/TestKey`-Alias, so dass er auf einen neuen KMS-Schlüssel verweist. Da der Vorgang keine Ausgabe zurückgibt, zeigt das Beispiel anhand des ListAliasesVorgangs, dass der Alias jetzt einem anderen KMS-Schlüssel zugeordnet ist und das `LastUpdatedDate` Feld aktualisiert wird. Die ListAliases Befehle verwenden den queryParameter in AWS CLI , um nur den `alias/TestKey` Alias abzurufen.

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/TestKey`]'
{
    "Aliases": [
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/TestKey",
            "AliasName": "alias/TestKey",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1521097200.123,
            "LastUpdatedDate": 1521097200.123
        },
    ]
}


$ aws kms update-alias --alias-name alias/TestKey --target-key-id
 0987dcba-09fe-87dc-65ba-ab0987654321

$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/TestKey`]'
{
    "Aliases": [
```

```
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/TestKey",
            "AliasName": "alias/TestKey",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1521097200.123,
            "LastUpdatedDate": 1604958290.722
        },
    ]
}
```

Ändern Sie den Primärschlüssel in einer Reihe von Schlüsseln für mehrere Regionen

Jeder Satz verwandter multiregionaler Schlüssel muss über einen Primärschlüssel verfügen. Sie können jedoch den Primärschlüssel ändern. Diese Aktion, bekannt als Aktualisieren der primären Region, konvertiert den aktuellen Primärschlüssel in einen Replikatschlüssel und konvertiert einen der verwandten Replikatschlüssel in den Primärschlüssel. Sie können dies tun, wenn Sie den aktuellen Primärschlüssel löschen müssen, während die Replikatschlüssel beibehalten werden, oder um den Primärschlüssel in derselben Region wie die Schlüsseladministratoren zu platzieren.

Sie können einen beliebigen verwandten Replikatschlüssel als neuen Primärschlüssel auswählen. Sowohl der Primärschlüssel als auch der Replikatschlüssel müssen sich im Schlüsselstatus `Enabled` befinden, wenn die Operation gestartet wird.

Der **`Updating`** Schlüsselstatus

> Auch nach Abschluss des `UpdatePrimaryRegion` Vorgangs dauert die Aktualisierung der primären Region möglicherweise noch einige Sekunden. Während dieser Zeit haben der alte und der neue Primärschlüssel den vorübergehenden Schlüsselstatus Updating. Während der Schlüsselstatus lautet `Updating` ist, können Sie die Schlüssel in kryptografischen Vorgängen verwenden, aber Sie können den neuen Primärschlüssel nicht replizieren oder bestimmte Verwaltungsvorgänge ausführen, wie z. B. das Aktivieren oder Deaktivieren dieser Schlüssel. Bei Vorgängen werden DescribeKeymöglicherweise sowohl der alte als auch der neue Primärschlüssel als Replikate angezeigt. Die `Enabled`-Schlüsselstatus wird wiederhergestellt, wenn die Aktualisierung abgeschlossen ist.
>
> Informationen zu den Auswirkungen des `Updating`-Schlüsselstatus finden Sie unter Wichtige Zustände von AWS KMS Schlüsseln.

Funktionsweise

Angenommen, Sie haben einen Primärschlüssel in USA Ost (Nord-Virginia) (us-east-1) und einen Replikatschlüssel in Europa (Irland) (eu-west-1). Sie können die Aktualisierungsfunktion verwenden, um den Primärschlüssel in USA Ost (Nord-Virginia) (us-east-1) in einen Replikatschlüssel zu ändern und den Replikatschlüssel in Europa (Irland) (eu-west-1) in den Primärschlüssel zu ändern.

Wenn der Aktualisierungsvorgang abgeschlossen ist, ist der multiregionaler Schlüssel in der Region Europa (Irland) (eu-west-1) ein multiregionaler Primärschlüssel und der Schlüssel in der Region USA Ost (Nord-Virginia) (us-east-1) ist sein Replikatschlüssel. Wenn andere verwandte Replikatschlüssel vorhanden sind, werden sie zu Replikaten des neuen Primärschlüssels. Beim nächsten AWS KMS Synchronisieren der gemeinsamen Eigenschaften der Schlüssel mit mehreren Regionen werden die gemeinsamen Eigenschaften aus dem neuen Primärschlüssel abgerufen und in die zugehörigen Replikatschlüssel kopiert, einschließlich des vorherigen Primärschlüssels.

Der Aktualisierungsvorgang hat keine Auswirkungen auf den Schüssel-ARN eines beliebigen multiregionalen Schlüssels. Es hat auch keine Auswirkungen auf gemeinsame Eigenschaften, wie das Schlüsselmaterial, oder auf unabhängige Eigenschaften, wie die Schlüsselrichtlinie. Sie können ggf. die Schlüsselrichtlinie des neuen Primärschlüssels aktualisieren. Sie könnten zum Beispiel dem neuen Primärschlüssel kms: ReplicateKey Permission for Trusted Principals hinzufügen und ihn aus dem neuen Replikatschlüssel entfernen.

Aktualisieren Sie die primäre Region

Sie können einen Replikatschlüssel in einen Primärschlüssel konvertieren, wodurch der frühere Primärschlüssel in ein Replikat umgewandelt wird. Um die primäre Region zu aktualisieren, benötigen Sie in beiden Regionen die UpdatePrimaryRegion Berechtigung kms:.

Sie können die primäre Region in der AWS KMS Konsole oder mithilfe des UpdatePrimaryRegionVorgangs aktualisieren.

Mithilfe der AWS KMS Konsole

Sie können den Primärschlüssel in der AWS KMS Konsole aktualisieren. Starten Sie auf der Seite mit den Schlüsseldetails des aktuellen Primärschlüssels.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Wählen Sie die Schlüssel-ID oder den Alias des multiregionalen Primärschlüssels aus. Dadurch wird die Seite mit den Schlüsseldetails des Primärschlüssels geöffnet.

   Um einen multiregionalen Primärschlüssel zu identifizieren, verwenden Sie das Werkzeugsymbol in der oberen rechten Ecke, um die Spalte Regionality (Regionalität) zur Tabelle hinzuzufügen.
5. Wählen Sie die Registerkarte Regionality (Regionalität) aus.
6. In dem Abschnitt Primary key (Primärschlüssel) wählen Sie Change primary Region (primäre Region ändern) aus.
7. Wählen Sie die Region des neuen Primärschlüssels aus. Sie können nur eine Region aus dem Menü auswählen.

   Das Menü Change primary Regions (primären Regionen ändern) enthält nur Regionen, die über einen verwandten multiregionalen Schlüssel verfügen. Möglicherweise haben Sie nicht die Berechtigung zum Aktualisieren der primären Region in allen Regionen auf dem Menü.
8. Wählen Sie Change primary region (Primäre Region ändern) aus.

Verwenden der API AWS KMS

Verwenden Sie den UpdatePrimaryRegionVorgang, um den Primärschlüssel in einer Reihe verwandter Schlüssel für mehrere Regionen zu ändern.

Verwenden Sie den `KeyId`-Parameter, um den aktuellen Primärschlüssel zu identifizieren. Verwenden Sie den `PrimaryRegion` Parameter, um den AWS-Region des neuen Primärschlüssels anzugeben. Wenn der Primärschlüssel noch kein Replikat in der neuen primären Region hat, schlägt die Operation fehl.

Im folgenden Beispiel wird der Primärschlüssel aus dem multiregionalen Schlüssel in der `us-west-2`-Region zu seinem Replikat im `eu-west-1`-Region geändert. Der `KeyId`-Parameter identifiziert den aktuellen Primärschlüssel in der Region `us-west-2`. Der `PrimaryRegion` Parameter gibt den AWS-Region des neuen Primärschlüssels an,`eu-west-1`.

```
$ aws kms update-primary-region \
```

```
      --key-id arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \
      --primary-region eu-west-1
```

Wenn diese Operation erfolgreich ist, gibt sie keine Ausgabe zurück, sondern nur den HTTP-Statuscode. Um den Effekt zu sehen, rufen Sie den DescribeKeyVorgang für einen der Multi-Region-Schlüssel auf. Möglicherweise möchten Sie warten, bis der Schlüsselstatus zu `Enabled` zurückkehrt. Während der Schlüsselstatus Updating ist, befinden sich die Werte für den Schlüssel möglicherweise noch im Fluss.

Beispielsweise erhält der folgenden `DescribeKey`-Aufruf die Details über den multiregionalen Schlüssel in der `eu-west-1`-Region. Die Ausgabe zeigt, dass der multiregionale Schlüssel in der Region `eu-west-1` ist jetzt der Primärschlüssel. Der verwandte multiregionale Schlüssel (gleiche Schlüssel-ID) in der Region `us-west-2` ist jetzt ein Replikatschlüssel.

```
$ aws kms describe-key \
      --key-id arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \

{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "CreationDate": 1609193147.831,
        "Enabled": true,
        "Description": "multi-region-key",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": true,
        "MultiRegionConfiguration": {
           "MultiRegionKeyType": "PRIMARY",
           "PrimaryKey": {
```

```
              "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
              "Region": "eu-west-1"
           },
           "ReplicaKeys": [
              {
                 "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                 "Region": "us-west-2"
              }
           ]
        }
    }
}
```

Lösche eine AWS KMS key

Das Löschen von AWS KMS key ist destruktiv und potenziell gefährlich. Damit löschen Sie das Schlüsselmaterial und alle Metadaten im Zusammenhang mit dem KMS-Schlüssel. Dies kann nicht rückgängig gemacht werden. Nach dem Löschen eines KMS-Schlüssels können Sie die Daten nicht mehr entschlüsseln, die mit diesem KMS-Schlüssel verschlüsselt wurden. Das bedeutet, dass die Daten nicht wiederhergestellt werden können. (Die einzigen Ausnahmen sind Replikatschlüssel für mehrere Regionen sowie asymmetrische und HMAC-KMS-Schlüssel mit importiertem Schlüsselmaterial.) Dieses Risiko ist bei asymmetrischen KMS-Schlüsseln, die zur Verschlüsselung verwendet werden, erheblich. Benutzer können ohne Warnung oder Fehler weiterhin Chiffretexte mit dem öffentlichen Schlüssel generieren, die nach dem Löschen des privaten Schlüssels nicht entschlüsselt werden können. AWS KMS

Löschen Sie einen KMS-Schlüssel nur, wenn Sie sicher sind, dass Sie ihn nicht mehr benötigen. Wenn Sie nicht sicher sind, sollten Sie den KMS-Schlüssel deaktivieren, anstatt ihn zu löschen. Sie können einen deaktivierten KMS-Schlüssel wieder aktivieren und die geplante Löschung eines Schlüssels abbrechen. Ein gelöschter KMS-Schlüssel kann jedoch nicht wiederhergestellt werden.

Sie können nur das Löschen eines vom Kunden verwalteten Schlüssels planen. Sie können nicht löschen oder. Von AWS verwaltete Schlüssel AWS-eigene Schlüssel

Bevor Sie einen KMS-Schlüssel löschen, möchten Sie vielleicht wissen, wie viele Chiffretexte unter diesem KMS-Schlüssel verschlüsselt wurden. AWS KMS speichert diese Informationen nicht und speichert auch keinen der Chiffretexte. Um diese Informationen abzurufen, müssen Sie selbst die bisherige Nutzung eines KMS-Schlüssels feststellen. Um Hilfe zu erhalten, gehen Sie zu Ermitteln Sie die frühere Verwendung eines KMS-Schlüssels.

AWS KMS löscht Ihre KMS-Schlüssel niemals, es sei denn, Sie planen ausdrücklich das Löschen und die vorgeschriebene Wartezeit läuft ab.

Für die Löschung eines KMS-Schlüssels können jedoch verschiedene Gründe sprechen:

- Der Lebenszyklus eines nicht mehr benötigten KMS-Schlüssels soll beendet werden.
- Der Aufwand und der Preis im Zusammenhang mit der Erhaltung ungenutzter KMS-Schlüssel sollen vermieden werden.
- Die Anzahl der KMS-Schlüssel, die auf das KMS-Schlüsselressourcen-Kontingent angerechnet werden, soll verringert werden.

> **Note**
>
> Wenn Sie Ihre schließen AWS-Konto, können Sie nicht mehr auf Ihre KMS-Schlüssel zugreifen und sie werden Ihnen nicht mehr in Rechnung gestellt.

AWS KMS zeichnet einen Eintrag in Ihrem AWS CloudTrail Protokoll auf, wenn Sie das Löschen des KMS-Schlüssels planen und wann der KMS-Schlüssel tatsächlich gelöscht wird.

Über die Wartezeit

Da das Löschen eines KMS-Schlüssels zerstörerisch und potenziell gefährlich ist, AWS KMS müssen Sie eine Wartezeit von 7 bis 30 Tagen festlegen. Die Standardwartezeit beträgt 30 Tage.

Die tatsächliche Wartezeit kann jedoch bis zu 24 Stunden länger sein als die, die Sie geplant haben. Verwenden Sie den DescribeKeyVorgang, um das tatsächliche Datum und die Uhrzeit der Löschung des KMS-Schlüssels zu ermitteln. Oder shcauen Sie in der AWS KMS -Konsole auf der Detailseite für den KMS-Schlüssel im Abschnitt General configuration (allgemeine Konfiguration) unter Scheduled deletion date (geplantes Löschdatum). Achten Sie darauf, die Zeitzone zu notieren.

Während der Wartezeit lauten der KMS-Schlüssel-Status und der Schlüsselstatus Pending deletion (Löschung ausstehend).

- Ein KMS-Schlüssel, der zur Löschung ansteht, kann nicht in kryptografischen Operationen verwendet werden.
- AWS KMS rotiert das Schlüsselmaterial von KMS-Schlüsseln, deren Löschung noch aussteht, nicht.

AWS KMS Löscht nach Ablauf der Wartezeit den KMS-Schlüssel, seine Aliase und alle zugehörigen AWS KMS Metadaten.

Die Planung des Löschens eines KMS-Schlüssels wirkt sich möglicherweise nicht sofort auf Datenschlüssel aus, die mit dem KMS-Schlüssel verschlüsselt wurden. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

Nutzen Sie die Wartezeit, um sicher zu stellen, dass Sie den KMS-Schlüssel jetzt oder in der Zukunft nicht mehr benötigen. Sie können einen CloudWatch Amazon-Alarm so konfigurieren, dass er Sie

warnt, wenn eine Person oder Anwendung während der Wartezeit versucht, den KMS-Schlüssel zu verwenden. Um den KMS-Schlüssel wiederherzustellen, können Sie die Löschung des Schlüssels abbrechen, bevor die Wartezeit endet. Nach Ablauf der Wartezeit können Sie das Löschen des Schlüssels nicht mehr abbrechen und der KMS-Schlüssel AWS KMS wird gelöscht.

Besondere Überlegungen

Bevor Sie die Löschung Ihrer Schlüssel planen, sollten Sie sich die folgenden besonderen Überlegungen zum Löschen von KMS-Schlüsseln für besondere Zwecke ansehen.

Löschen von asymmetrischen KMS-Schlüsseln

Benutzer mit entsprechender Autorisierung können symmetrische oder asymmetrische KMS-Schlüssel löschen. Das Verfahren zum Planen des Löschens dieser KMS-Schlüssel ist für beide Arten von Schlüsseln gleich. Da der öffentliche Schlüssel eines asymmetrischen KMS-Schlüssels jedoch heruntergeladen und außerhalb von verwendet werden kann, birgt der Vorgang erhebliche zusätzliche Risiken AWS KMS, insbesondere bei asymmetrischen KMS-Schlüsseln, die zur Verschlüsselung verwendet werden (die Schlüsselverwendung ist). `ENCRYPT_DECRYPT`

- Wenn Sie das Löschen eines KMS-Schlüssels planen, ändert sich der Schlüsselstatus des KMS-Schlüssels zu Pending deletion (Löschung ausstehend) und der KMS-Schlüssel kann nicht in kryptografischen Operationen verwendet werden. Die geplante Löschung hat jedoch keine Auswirkungen auf öffentliche Schlüssel außerhalb von. AWS KMS Benutzer, die über den öffentlichen Schlüssel verfügen, können ihn weiterhin zum Verschlüsseln von Nachrichten verwenden. Sie erhalten keine Benachrichtigung, dass sich der Schlüsselstatus geändert hat. Wenn der Löschvorgang nicht abgebrochen wird, kann der mit dem öffentlichen Schlüssel erstellte Chiffretext nicht entschlüsselt werden.
- Alarme, Protokolle und andere Strategien, die die versuchte Verwendung eines KMS-Schlüssels erkennen, dessen Löschung aussteht, können die Verwendung des öffentlichen Schlüssels außerhalb von AWS KMS nicht erkennen.
- Wenn der KMS-Schlüssel gelöscht wird, schlagen alle AWS KMS Aktionen fehl, die diesen KMS-Schlüssel betreffen. Benutzer, die über den öffentlichen Schlüssel verfügen, können sie jedoch weiterhin zum Verschlüsseln von Nachrichten verwenden. Diese Chiffretexte können nicht entschlüsselt werden.

Wenn Sie einen asymmetrischen KMS-Schlüssel mit einer Schlüsselverwendung von löschen müssen`ENCRYPT_DECRYPT`, ermitteln Sie anhand Ihrer CloudTrail Protokolleinträge, ob der öffentliche Schlüssel heruntergeladen und gemeinsam genutzt wurde. Wenn dies der Fall ist,

stellen Sie sicher, dass der öffentliche Schlüssel außerhalb von AWS KMS nicht verwendet wird. Ziehen Sie dann in Betracht, den KMS-Schlüssel zu deaktivieren, anstatt ihn zu löschen.

Das Risiko, das durch das Löschen eines asymmetrischen KMS-Schlüssels entsteht, wird für asymmetrische KMS-Schlüssel mit importiertem Schlüsselmaterial verringert. Details hierzu finden Sie unter Deleting KMS keys with imported key material.

Schlüssel für mehrere Regionen löschen

Um einen Primärschlüssel zu löschen, müssen Sie das Löschen all seiner Replikatschlüssel planen und dann warten, bis die Replikatschlüssel gelöscht werden. Die erforderliche Wartezeit für das Löschen eines Primärschlüssels beginnt, wenn der letzte seiner Replikatschlüssel gelöscht wird. Wenn Sie einen Primärschlüssel aus einer bestimmten Region löschen müssen, ohne dessen Replikatschlüssel zu löschen, ändern Sie den Primärschlüssel in einen Replikatschlüssel, indem Sie die primäre Region aktualisieren.

Sie können einen Replikatschlüssel jederzeit löschen. Es hängt nicht vom Schlüsselstatus eines anderen KMS-Schlüssels ab. Wenn Sie versehentlich einen Replikatschlüssel löschen, können Sie ihn neu erstellen, indem Sie denselben Primärschlüssel in derselben Region replizieren. Der neue Replikatschlüssel, den Sie erstellen, hat die gleichen gemeinsamen Eigenschaften wie der ursprüngliche Replikatschlüssel.

Löschen von KMS-Schlüsseln mit importiertem Schlüsselmaterial

Das Löschen des Schlüsselmaterials eines KMS-Schlüssels mit importiertem Schlüsselmaterial ist temporär und reversibel. Um den Schlüssel wiederherzustellen, importieren Sie das zugehörige Schlüsselmaterial erneut.

Im Gegensatz dazu kann das Löschen eines KMS-Schlüssels nicht rückgängig gemacht werden. Wenn Sie das Löschen von Schlüsseln planen und die erforderliche Wartezeit abläuft, werden der KMS-Schlüssel, sein Schlüsselmaterial und alle mit dem KMS-Schlüssel verknüpften Metadaten AWS KMS dauerhaft und unwiderruflich gelöscht.

Das Risiko und die Folgen des Löschens eines KMS-Schlüssels mit importiertem Schlüsselmaterial hängen jedoch vom Typ („Schlüsselspezifikation“) des KMS-Schlüssels ab.

- Symmetrische Verschlüsselungsschlüssel – Wenn Sie einen KMS-Schlüssel mit symmetrischer Verschlüsselung löschen, können alle verbleibenden Geheimtexte, die mit diesem Schlüssel verschlüsselt wurden, nicht wiederhergestellt werden. Sie können keinen neuen symmetrischen KMS-Schlüssel erstellen, der die Geheimtexte eines gelöschten symmetrischen KMS-Schlüssels entschlüsseln kann, selbst wenn Sie dasselbe Schlüsselmaterial haben. Metadaten,

die für jeden KMS-Schlüssel einzigartig sind, werden kryptografisch an jeden symmetrischen Geheimtext gebunden. Dieses Sicherheits-Feature garantiert, dass nur der KMS-Schlüssel, mit dem der symmetrische Geheimtext verschlüsselt wurde, ihn entschlüsseln kann, verhindert jedoch, dass Sie einen entsprechenden KMS-Schlüssel neu erstellen können.

- Asymmetrische Schlüssel und HMAC-Schlüssel — Wenn Sie über das ursprüngliche Schlüsselmaterial verfügen, können Sie einen neuen KMS-Schlüssel mit denselben kryptografischen Eigenschaften wie ein asymmetrischer oder gelöschter HMAC-KMS-Schlüssel erstellen. AWS KMS generiert standardmäßige RSA-Verschlüsselungstexte und -Signaturen, ECC-Signaturen und HMAC-Tags, die keine eindeutigen Sicherheitsmerkmale enthalten. Sie können auch einen HMAC-Schlüssel oder den privaten Schlüssel eines asymmetrischen Schlüsselpaares außerhalb von AWS verwenden.

  Ein neuer KMS-Schlüssel, den Sie mit demselben asymmetrischen oder HMAC-Schlüsselmaterial erstellen, hat eine andere Schlüssel-ID. Sie müssen eine neue Schlüsselrichtlinie erstellen, alle Aliase neu erstellen und bestehende IAM-Richtlinien und -Berechtigungen aktualisieren, damit sie auf den neuen Schlüssel verweisen.

Löschen von KMS-Schlüsseln aus einem Schlüsselspeicher AWS CloudHSM

Wenn Sie das Löschen eines KMS-Schlüssels aus einem AWS CloudHSM Schlüsselspeicher planen, ändert sich sein Schlüsselstatus in Ausstehende Löschung. Der KMS-Schlüssel verbleibt auch während der Wartezeit im Status Pending deletion (Löschung ausstehend), selbst wenn der KMS-Schlüssel nicht mehr verfügbar ist, weil Sie die Verbindung zu dem benutzerdefinierten Schlüsselspeicher getrennt haben. Dies erlaubt es Ihnen, die Löschung des KMS-Schlüssels vor Ablauf der Wartezeit jederzeit abzubrechen.

Wenn die Wartezeit abgelaufen ist, AWS KMS löscht den KMS-Schlüssel von AWS KMS. AWS KMS bemüht sich dann nach besten Kräften, das Schlüsselmaterial aus dem zugehörigen AWS CloudHSM Cluster zu löschen. Wenn AWS KMS Schlüsselmaterial nicht löschen kann, beispielsweise, wenn der Schlüsselspeicher von AWS KMS getrennt wurde, müssen Sie möglicherweise verwaistes Schlüsselmaterial manuell aus dem Cluster löschen.

AWS KMS löscht das Schlüsselmaterial nicht aus Cluster-Backups. Selbst wenn Sie den KMS-Schlüssel AWS KMS und das zugehörige Schlüsselmaterial aus Ihrem AWS CloudHSM Cluster löschen, können aus Backups erstellte Cluster das gelöschte Schlüsselmaterial enthalten. Um das Schlüsselmaterial dauerhaft zu löschen, verwenden Sie den DescribeKeyVorgang, um das Erstellungsdatum des KMS-Schlüssels zu ermitteln. Löschen Sie dann Cluster-Sicherungen, die das Schlüsselmaterial noch enthalten könnten.

Wenn Sie das Löschen eines KMS-Schlüssels aus einem AWS CloudHSM Schlüsselspeicher planen, wird der KMS-Schlüssel sofort unbrauchbar (abhängig von der eventuellen Konsistenz). Ressourcen, die mit durch den KMS-Schlüssel geschützten Datenschlüsseln verschlüsselt wurden, sind jedoch nicht betroffen, bis der KMS-Schlüssel erneut verwendet wird, z. B. zur Entschlüsselung des Datenschlüssels. Dieses Problem betrifft AWS-Services, dass viele von ihnen Datenschlüssel verwenden, um Ihre Ressourcen zu schützen. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

Löschen von KMS-Schlüsseln aus einem externen Schlüsselspeicher

Das Löschen eines KMS-Schlüssels aus einem externen Schlüsselspeicher hat keine Auswirkungen auf den externen Schlüssel, der als sein Schlüsselmaterial diente.

Wenn Sie die Löschung eines KMS-Schlüssels aus einem externen Schlüsselspeicher planen, ändert sich der Status des Schlüssels in Pending deletion (Löschung ausstehend). Der KMS-Schlüssel verbleibt auch während der Wartezeit im Status Pending deletion (Löschung ausstehend), selbst wenn der KMS-Schlüssel nicht mehr verfügbar ist, weil Sie die Verbindung zum externen Schlüsselspeicher getrennt haben. Dies erlaubt es Ihnen, die Löschung des KMS-Schlüssels vor Ablauf der Wartezeit jederzeit abzubrechen. Wenn die Wartezeit abgelaufen ist, AWS KMS löscht den KMS-Schlüssel von AWS KMS.

Wenn Sie die Löschung eines KMS-Schlüssels aus einem externen Schlüsselspeicher planen, wird der KMS-Schlüssel sofort unbrauchbar (vorbehaltlich einer letztendlichen Konsistenz). Ressourcen, die mit durch den KMS-Schlüssel geschützten Datenschlüsseln verschlüsselt wurden, sind jedoch nicht betroffen, bis der KMS-Schlüssel erneut verwendet wird, z. B. zur Entschlüsselung des Datenschlüssels. Dieses Problem betrifft AWS-Services, von denen viele Datenschlüssel verwenden, um Ihre Ressourcen zu schützen. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

Steuert den Zugriff auf das Löschen von Schlüsseln

Wenn Sie IAM-Richtlinien verwenden, um AWS KMS Berechtigungen zuzulassen, sind IAM-Identitäten mit AWS Administratorzugriff (`"Action": "*"`) oder AWS KMS Vollzugriff (`"Action": "kms:*"`) bereits berechtigt, das Löschen von KMS-Schlüsseln per Schlüssel zu planen und abzubrechen. Um es Schlüsseladministratoren zu ermöglichen, das Löschen von Schlüsseln in der Schlüsselrichtlinie zu planen und abzubrechen, verwenden Sie die AWS KMS Konsole oder die API. AWS KMS

In der Regel sind nur Schlüsseladministratoren berechtigt, das Löschen von Schlüsseln zu planen oder abzubrechen. Sie können diese Berechtigungen jedoch auch anderen IAM-Identitäten erteilen, indem Sie der Schlüsselrichtlinie oder einer IAM-Richtlinie die Berechtigungen `kms:ScheduleKeyDeletion` und `kms:CancelKeyDeletion` hinzufügen. Sie können den `kms:ScheduleKeyDeletionPendingWindowInDays`Bedingungsschlüssel auch verwenden, um die Werte, die Prinzipale im `PendingWindowInDays` Parameter einer ScheduleKeyDeletionAnfrage angeben können, weiter einzuschränken.

Erlauben Sie Schlüsseladministratoren, das Löschen von Schlüsseln zu planen und abzubrechen

Verwenden der AWS KMS -Konsole

So erteilen Sie Schlüsseladministratoren die Berechtigung, das Löschen von Schlüsseln zu planen und abzubrechen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Wählen Sie den Alias oder die Schlüssel-ID des KMS-Schlüssels, dessen Berechtigungen Sie ändern möchten.
5. Wählen Sie die Registerkarte Key policy (Schlüsselrichtlinie).
6. Der nächste Schritt unterscheidet sich für die Standardansicht und die Richtlinienansicht Ihrer Schlüsselrichtlinie. Die Standardansicht ist nur verfügbar, wenn Sie die standardmäßige Konsolenschlüsselrichtlinie verwenden. Andernfalls ist nur die Richtlinienansicht verfügbar.

   Wenn die Standardansicht verfügbar ist, wird auf der Registerkarte Key policy (Schlüsselrichtlinie) die Schaltfläche Switch to policy view (zur Richtlinienansicht wechseln) oder Switch to default view (zur Standardansicht wechseln) angezeigt.

   - In der Standardansicht:
     - Wählen Sie unter Key deletion (Schlüssel-Löschung) die Option Allow key administrators to delete this key (Schlüsseladministratoren das Löschen dieses Schlüssels erlauben) aus.
   - In der Richtlinienansicht:

a. Wählen Sie Bearbeiten aus.

b. Fügen Sie in der Richtlinienanweisung für Schlüsseladministratoren dem `Action`-Element die Berechtigungen `kms:ScheduleKeyDeletion` und `kms:CancelKeyDeletion` hinzu.

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSKeyAdmin"},
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

c. Wählen Sie Änderungen speichern aus.

Verwenden der API AWS KMS

Sie können das verwenden AWS Command Line Interface , um Berechtigungen für das Planen und Abbrechen des Löschens von Schlüsseln hinzuzufügen.

So fügen Sie Berechtigungen zum Planen und Abbrechen der Löschung eines Schlüssels hinzu

1. Verwenden Sie den Befehl `aws kms get-key-policy`, um die vorhandene Schlüsselrichtlinie aufzurufen, und speichern Sie das Richtliniendokument dann in einer Datei.
2. Öffnen Sie die Richtlinien in Ihrem bevorzugten Texteditor. Fügen Sie in der Richtlinienerklärung für Schlüsseladministratoren die Berechtigungen`kms:ScheduleKeyDeletion` und

`kms:CancelKeyDeletion` hinzu. Das folgende Beispiel zeigt eine Richtlinienanweisung mit diesen zwei Berechtigungen:

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSKeyAdmin"},
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms:Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

3. Verwenden Sie den Befehl `aws kms put-key-policy`, um die Schlüsselrichtlinie auf den KMS-Schlüssel anzuwenden.

Löschen von Schlüsseln planen

In den folgenden Verfahren wird beschrieben, wie das Löschen von Schlüsseln geplant und das Löschen von Schlüsseln AWS KMS keys (KMS-Schlüsseln) AWS KMS mithilfe der AWS Management Console und der AWS KMS API abgebrochen wird.

> **Warning**
>
> Das Löschen eines KMS-Schlüssels ist ein endgültiger und potenziell gefährlicher Vorgang. Fahren Sie nur fort, wenn Sie sicher sind, dass Sie den KMS-Schlüssel später nicht mehr verwenden müssen. Wenn Sie nicht sicher sind, sollten Sie den KMS-Schlüssel deaktivieren, anstatt ihn zu löschen.

Bevor Sie einen KMS-Schlüssel löschen können, müssen Sie über die entsprechende Berechtigung verfügen. Hinweise zum Erteilen dieser Berechtigungen an Schlüsseladministratoren finden Sie unter Steuert den Zugriff auf das Löschen von Schlüsseln. Sie können den Bedingungsschlüssel `kms:ScheduleKeyDeletionPendingWindowInDays` auch verwenden, um die Wartezeit weiter einzuschränken, z. B. um eine Mindestwartezeit durchzusetzen.

AWS KMS zeichnet einen Eintrag in Ihrem AWS CloudTrail Protokoll auf, wenn Sie das Löschen des KMS-Schlüssels planen und wann der KMS-Schlüssel tatsächlich gelöscht wird.

Verwenden der AWS KMS -Konsole

In der AWS Management Console können Sie das Löschen mehrerer KMS-Schlüssel gleichzeitig planen und abbrechen.

So planen Sie die Löschung

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.

   Sie können das Löschen von Von AWS verwaltete Schlüssel oder AWS-eigene Schlüssel nicht planen.
4. Aktivieren Sie das Kontrollkästchen neben dem KMS-Schlüssel, den Sie löschen möchten.
5. Wählen Sie Key actions (Schlüsselaktionen), Schedule key deletion (Schlüssellöschung planen).
6. Lesen und berücksichtigen Sie die Warnung und die Informationen zum Abbrechen des Löschens während der Wartezeit. Wenn Sie den Löschvorgang abbrechen möchten, wählen Sie unten auf der Seite Cancel (Abbrechen).
7. Geben Sie für Waiting period (in days) (Wartezeit (in Tagen)) eine Anzahl von Tagen zwischen 7 und 30 ein.
8. Überprüfen Sie die KMS-Schlüssel, die Sie löschen.
9. Aktivieren Sie das Kontrollkästchen neben Bestätigen Sie, dass Sie den Löschvorgang für diesen Schlüssel in ***<number of days>*** Tagen planen möchten. .
10. Wählen Sie Schedule deletion.

Der Status des Schlüssels wechselt zu Pending Deletion (Löschung ausstehend).

Verwenden der AWS KMS -API

Verwenden Sie den Befehl `aws kms schedule-key-deletion`, um die Löschung eines vom Kunden verwalteten Schlüssels, wie im folgenden Beispiel gezeigt, zu planen.

Sie können das Löschen eines Von AWS verwalteter Schlüssel Oders nicht planen AWS-eigener Schlüssel.

```
$ aws kms schedule-key-deletion --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --
pending-window-in-days 10
```

Bei erfolgreicher Verwendung wird eine Ausgabe wie die im folgenden Beispiel gezeigte Ausgabe AWS CLI zurückgegeben:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "DeletionDate": 1598304792.0,
    "KeyState": "PendingDeletion",
    "PendingWindowInDays": 10
}
```

Löschen des Schlüssels abbrechen

Nachdem Sie das Löschen eines KMS-Schlüssels geplant haben, können Sie das Löschen des Schlüssels abbrechen, solange sich der Schlüssel noch im Status Ausstehende Löschung befindet. Sie können das Löschen von Schlüsseln in der AWS KMS Konsole oder mithilfe des CancelKeyDeletionVorgangs abbrechen. Nachdem Sie das ausstehende Löschen eines KMS-Schlüssels abgebrochen haben, lautet der Schlüsselstatus des KMS-Schlüssels`Disabled`. Weitere Informationen zur Aktivierung des KMS-Schlüssels finden Sie unterTasten aktivieren und deaktivieren.

Verwenden der AWS KMS Konsole

So brechen Sie die Löschung eines Schlüssels ab

1. Öffnen Sie die AWS KMS Konsole unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.

3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Aktivieren Sie das Kontrollkästchen neben dem KMS-Schlüssel, den Sie wiederherstellen möchten.
5. Wählen Sie Key actions (Schlüsselaktionen), Cancel key deletion (Schlüssellöschung abbrechen).

Der Status des KMS-Schlüssels wechselt von Pending Deletion (Löschung ausstehend zu Disabled (deaktiviert). Um den KMS-Schlüssel verwenden zu können, müssen Sie ihn aktivieren.

Verwenden der API AWS KMS

Verwenden Sie den `aws kms cancel-key-deletion`Befehl, um das Löschen von Schlüsseln aus dem abzubrechen, AWS CLI wie im folgenden Beispiel gezeigt.

```
$ aws kms cancel-key-deletion --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Bei erfolgreicher Verwendung wird eine Ausgabe wie die im folgenden Beispiel gezeigte Ausgabe AWS CLI zurückgegeben:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Der Status des KMS-Schlüssels wechselt von Pending deletion (Löschung aussthend) zu Disabled (deaktiviert). Um den KMS-Schlüssel verwenden zu können, müssen Sie ihn aktivieren.

Erstellen Sie einen Alarm, der die Verwendung eines KMS-Schlüssels erkennt, dessen Löschung noch aussteht

Sie können die Funktionen von AWS CloudTrail Amazon CloudWatch Logs und Amazon Simple Notification Service (Amazon SNS) kombinieren, um einen CloudWatch Amazon-Alarm zu erstellen, der Sie benachrichtigt, wenn jemand in Ihrem Konto versucht, einen KMS-Schlüssel zu verwenden, dessen Löschung noch aussteht. Wenn Sie diese Benachrichtigung erhalten, sollten Sie eventuell das Löschen des KMS-Schlüssels stornieren und erneut abwägen, ob es sinnvoll ist, diesen KMS-Schlüssel zu löschen.

Die folgenden Verfahren erzeugen einen Alarm, der Sie jedes Mal benachrichtigt, wenn die "*Key ARN* `is pending deletion`" -Fehlermeldung in Ihre CloudTrail Protokolldateien geschrieben wird. Diese Fehlermeldung gibt an, dass ein Benutzer oder eine Anwendung versucht hat, den KMS-Schlüssel in einer kryptografischen Operation zu verwenden. Da diese Benachrichtigung mit der Fehlermeldung verknüpft ist, wird sie nicht ausgelöst, wenn Sie API-Operationen ausführen, die für KMS-Schlüssel zulässig sind, bei denen die Löschung aussteht, z. B. `ListKeys`, `CancelKeyDeletion` und `PutKeyPolicy`. Eine Liste der AWS KMS API-Operationen, die diese Fehlermeldung zurückgeben, finden Sie unterWichtige Zustände von AWS KMS Schlüsseln.

Die Benachrichtigungs-E-Mail, die Sie erhalten, enthält weder den KMS-Schlüssel noch die kryptografische Operation. Diese Informationen finden Sie in Ihrem CloudTrail-Protokoll. Die E-Mail informiert Sie darüber, dass der Alarmstatus von OK zu Alarm geändert wurde. Weitere Informationen zu CloudWatch Alarmen und Statusänderungen finden Sie unter Verwenden von CloudWatch Amazon-Alarmen im CloudWatch Amazon-Benutzerhandbuch.

> **Warning**
>
> Dieser CloudWatch Amazon-Alarm kann die Verwendung des öffentlichen Schlüssels eines asymmetrischen KMS-Schlüssels außerhalb von AWS KMS nicht erkennen. Weitere Informationen zu den besonderen Risiken des Löschens asymmetrischer KMS-Schlüssel, die für die Kryptographie von öffentlichen Schlüsseln verwendet werden, einschließlich der Erstellung von Chiffretexten, die nicht entschlüsselt werden können, finden Sie unter Deleting asymmetric KMS keys.

In diesem Verfahren erstellen Sie einen Metrikfilter für CloudWatch Protokollgruppen, der Instanzen der Ausnahme „Ausstehende Löschung“ findet. Anschließend erstellen Sie einen CloudWatch Alarm, der auf der Metrik der Protokollgruppe basiert. Informationen zu Metrikfiltern für Protokollgruppen finden Sie unter Metriken aus Protokollereignissen mithilfe von Filtern erstellen im Amazon CloudWatch Logs-Benutzerhandbuch.

1. Erstellen Sie einen CloudWatch Metrikfilter, der CloudTrail Logs analysiert.

   Folgen Sie den Anweisungen unter Create a metric filter for a log group (Erstellen eines Metrikfilters für eine Protokollgruppe) mit den folgenden erforderlichen Werten. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Value (Wert)
Filtermuster	`{ $.eventSource = kms* && $.errorMessage = "* is pending deletion."}`
Metrikwert	1

2. Erstellen Sie einen CloudWatch Alarm auf der Grundlage des Metrikfilters, den Sie in Schritt 1 erstellt haben.

   Folgen Sie den Anweisungen unter Erstellen eines CloudWatch Alarms auf der Grundlage eines Metrikfilters für Protokollgruppen und verwenden Sie dabei die folgenden erforderlichen Werte. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Value (Wert)
Metrikfilter	Der Name des Metrikfilters, den Sie in Schritt 1 erstellt haben.
Threshold-Typ	Statisch
Bedingungen	Wann immer ***metric-name*** ist Größer/Gleich als 1
Daten deuten auf einen Alarm hin	1 von 1
Fehlende Datenbehandlung	Fehlende Daten als gut behandeln (keine Verletzung des Schwellenwerts)

Nachdem Sie diesen Vorgang abgeschlossen haben, erhalten Sie jedes Mal eine Benachrichtigung, wenn Ihr neuer CloudWatch Alarm den `ALARM` Status erreicht. Wenn Sie eine Benachrichtigung für diesen Alarm erhalten, kann das bedeuten, dass zum Verschlüsseln oder Entschlüsseln von Daten immer noch ein KMS-Schlüssel erforderlich ist, der gelöscht werden soll. In diesem Fall stornieren Sie das Löschen des KMS-Schlüssels und überdenken Sie, ob es sinnvoll ist, diesen KMS-Schlüssel zu löschen.

Ermitteln Sie die frühere Verwendung eines KMS-Schlüssels

Bevor Sie einen KMS-Schlüssel löschen, möchten Sie vielleicht wissen, wie viele Chiffretexte unter diesem Schlüssel verschlüsselt wurden. AWS KMS speichert diese Informationen nicht und speichert auch keinen der Chiffretexte. Wenn Sie wissen, wie ein KMS-Schlüssel in der Vergangenheit verwendet wurde, kann das Ihnen bei der Entscheidung helfen, ob Sie ihn in der Zukunft benötigen. In diesem Thema werden verschiedene Strategien vorgeschlagen, mit denen Sie die frühere Nutzung eines KMS-Schlüssels feststellen können.

> **Warning**
>
> Diese Strategien zur Bestimmung der vergangenen und tatsächlichen Nutzung sind nur für AWS Benutzer und AWS KMS Betriebsabläufe wirksam. Sie können die Nutzung des öffentlichen Schlüssels eines asymmetrischen KMS-Schlüssels außerhalb von AWS KMS nicht erkennen. Weitere Informationen zu den besonderen Risiken des Löschens asymmetrischer KMS-Schlüssel, die für die Kryptographie von öffentlichen Schlüsseln verwendet werden, einschließlich der Erstellung von Chiffretexten, die nicht entschlüsselt werden können, finden Sie unter Deleting asymmetric KMS keys.

Themen

- Untersuchen Sie die Schlüsselberechtigungen für KMS, um den Umfang der potenziellen Nutzung zu ermitteln
- Untersuchen Sie die AWS CloudTrail Protokolle, um die tatsächliche Nutzung zu ermitteln

Untersuchen Sie die Schlüsselberechtigungen für KMS, um den Umfang der potenziellen Nutzung zu ermitteln

Wenn Sie bestimmen, wer oder was derzeit Zugriff auf einen KMS-Schlüssel hat, kann Ihnen das helfen zu bestimmen, wie weit der KMS-Schlüssel verwendet wurde und ob er noch benötigt wird. Um zu erfahren, wie man feststellt, wer oder was derzeit Zugriff auf einem KMS-Schlüssel hat, öffnen Sie Bestimmung des Zugriffs auf AWS KMS keys.

Untersuchen Sie die AWS CloudTrail Protokolle, um die tatsächliche Nutzung zu ermitteln

Sie können mit der Nutzungshistorik eines KMS-Schlüssels feststellen, ob Chiffretexte unter einem bestimmten KMS-Schlüssel verschlüsselt sind.

Alle AWS KMS API-Aktivitäten werden in AWS CloudTrail Protokolldateien aufgezeichnet. Wenn Sie in der Region, in der sich Ihr KMS-Schlüssel befindet, einen CloudTrail Trail erstellt haben, können Sie Ihre CloudTrail Protokolldateien überprüfen, um einen Verlauf aller AWS KMS API-Aktivitäten für einen bestimmten KMS-Schlüssel einzusehen. Wenn Sie keinen Trail haben, können Sie sich aktuelle Ereignisse trotzdem in Ihrem CloudTrail Eventverlauf ansehen. Einzelheiten zur AWS KMS Nutzung finden Sie CloudTrail unterAWS KMS API-Aufrufe protokollieren mit AWS CloudTrail.

Die folgenden Beispiele zeigen CloudTrail Protokolleinträge, die generiert werden, wenn ein KMS-Schlüssel zum Schutz eines in Amazon Simple Storage Service (Amazon S3) gespeicherten Objekts verwendet wird. In diesem Beispiel wird das Objekt unter Verwendung der serverseitigen Verschlüsselung mit KMS-Schlüsseln (SSE-KMS) in Simple Storage Service (Amazon S3) hochgeladen. Wenn Sie ein Objekt auf Simple Storage Service (Amazon S3) mit SSE-KMS hochladen, geben Sie den KMS-Schlüssel an, mit dem das Objekts geschützt werden soll. Amazon S3 verwendet den AWS KMS GenerateDataKeyVorgang, um einen eindeutigen Datenschlüssel für das Objekt anzufordern, und dieses Anforderungsereignis wird CloudTrail mit einem Eintrag angemeldet, der dem folgenden ähnelt:

```
{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROACKCEVSQ6C2EXAMPLE:example-user",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admins/example-user",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2015-09-10T23:12:48Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admins",
```

```
        "accountId": "111122223333",
        "userName": "Admins"
      }
    },
    "invokedBy": "internal.amazonaws.com"
  },
  "eventTime": "2015-09-10T23:58:18Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "internal.amazonaws.com",
  "userAgent": "internal.amazonaws.com",
  "requestParameters": {
    "encryptionContext": {"aws:s3:arn": "arn:aws:s3:::example_bucket/example_object"},
    "keySpec": "AES_256",
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": null,
  "requestID": "cea04450-5817-11e5-85aa-97ce46071236",
  "eventID": "80721262-21a5-49b9-8b63-28740e7ce9c9",
  "readOnly": true,
  "resources": [{
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Wenn Sie dieses Objekt später von Amazon S3 herunterladen, sendet Amazon S3 eine `Decrypt` Anfrage AWS KMS an, den Datenschlüssel des Objekts mithilfe des angegebenen KMS-Schlüssels zu entschlüsseln. Wenn Sie dies tun, enthalten Ihre CloudTrail Protokolldateien einen Eintrag, der dem folgenden ähnelt:

```
{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROACKCEVSQ6C2EXAMPLE:example-user",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admins/example-user",
    "accountId": "111122223333",
```

```
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2015-09-10T23:12:48Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admins",
        "accountId": "111122223333",
        "userName": "Admins"
      }
    },
    "invokedBy": "internal.amazonaws.com"
  },
  "eventTime": "2015-09-10T23:58:39Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "internal.amazonaws.com",
  "userAgent": "internal.amazonaws.com",
  "requestParameters": {
    "encryptionContext": {"aws:s3:arn": "arn:aws:s3:::example_bucket/example_object"}},
  "responseElements": null,
  "requestID": "db750745-5817-11e5-93a6-5b87e27d91a0",
  "eventID": "ae551b19-8a09-4cfc-a249-205ddba330e3",
  "readOnly": true,
  "resources": [{
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```


Alle AWS KMS API-Aktivitäten werden von protokolliert CloudTrail. Durch die Auswertung dieser Protokolleinträge können Sie die bisherige Nutzung eines bestimmten KMS-Schlüssels feststellen, und so können Sie bestimmen, ob Sie ihn löschen möchten.

Weitere Beispiele dafür, wie AWS KMS API-Aktivitäten in Ihren CloudTrail Protokolldateien erscheinen, finden Sie unterAWS KMS API-Aufrufe protokollieren mit AWS CloudTrail. Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Importiertes Schlüsselmaterial löschen

Sie können das importierte Schlüsselmaterial jederzeit aus einem KMS-Schlüssel löschen. Außerdem wird das Schlüsselmaterial AWS KMS gelöscht, wenn importiertes Schlüsselmaterial mit einem Ablaufdatum abläuft. In beiden Fällen ändert sich der Schlüsselstatus des KMS-Schlüssels in Ausstehender Import, wenn das Schlüsselmaterial gelöscht wird, und der KMS-Schlüssel kann nicht für kryptografische Operationen verwendet werden.

Symmetrischen Verschlüsselungsschlüsseln mit einer Region können mehrere Schlüsselmaterialien zugeordnet sein, und das Löschen oder Ablaufen von Schlüsselmaterial in einem anderen Zustand als *PENDING_ROTATION* ändert den Schlüsselstatus in Ausstehender Import. Für diese Schlüssel weist KMS jedem Schlüsselmaterial eine eindeutige Kennung zu. Sie können die ListKeyRotationsAPI verwenden, um diese wichtigen Materialkennungen einzusehen. Sie können ein bestimmtes Schlüsselmaterial löschen, indem Sie seinen Bezeichner mithilfe des `key-material-id` Parameters in der DeleteImportedKeyMaterialAPI angeben.

> **Warning**
>
> Der `key-material-id` Parameter ist optional. Wenn Sie ihn nicht angeben, AWS KMS wird das aktuelle Schlüsselmaterial gelöscht.

Neben dem Deaktivieren des KMS-Schlüssels und dem Entziehen von Berechtigungen kann das Löschen von Schlüsselmaterial als Strategie verwendet werden, um die Verwendung des KMS-Schlüssels schnell, aber vorübergehend einzustellen. Im Gegensatz dazu wird bei der Planung des Löschens eines KMS-Schlüssels mit importiertem Schlüsselmaterial auch schnell die Verwendung des KMS-Schlüssels gestoppt. Wenn der Löschvorgang jedoch während der Wartezeit nicht abgebrochen wird, werden der KMS-Schlüssel, die zugehörigen Schlüsselmaterialien und alle wichtigen Metadaten dauerhaft gelöscht. Details hierzu finden Sie unter Deleting KMS keys with imported key material.

Um Schlüsselmaterial zu löschen, können Sie die AWS KMS Konsole oder den DeleteImportedKeyMaterialAPI-Vorgang verwenden. AWS KMS zeichnet einen Eintrag in Ihrem AWS

CloudTrail Protokoll auf, wenn Sie importiertes Schlüsselmaterial löschen und wenn abgelaufenes Schlüsselmaterial AWS KMS gelöscht wird.

Wie sich das Löschen von Schlüsselmaterial auf Dienste auswirkt AWS

Wenn Sie Schlüsselmaterial löschen, wird der KMS-Schlüssel sofort unbrauchbar (vorausgesetzt, dass er irgendwann konsistent ist). Ressourcen, die mit durch den KMS-Schlüssel geschützten Datenschlüsseln verschlüsselt wurden, sind jedoch nicht betroffen, bis der KMS-Schlüssel erneut verwendet wird, z. B. zur Entschlüsselung des Datenschlüssels. Dieses Problem betrifft AWS-Services, von denen viele Datenschlüssel verwenden, um Ihre Ressourcen zu schützen. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

Verwenden der Konsole AWS KMS

Sie können die AWS KMS Konsole verwenden, um Schlüsselmaterial zu löschen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Führen Sie eine der folgenden Aktionen aus:
   - Aktivieren Sie das Kontrollkästchen für einen KMS-Schlüssel mit importiertem Schlüsselmaterial. Wählen Sie unter Key actions die Option Delete key material. Bei symmetrischen Verschlüsselungsschlüsseln, denen mehrere Schlüsselmaterialien zugeordnet sind, wird dadurch das aktuelle Schlüsselmaterial gelöscht.
   - Wählen Sie für KMS-Schlüssel mit symmetrischer Verschlüsselung für einzelne Regionen mit importiertem Schlüsselmaterial den Alias oder die Schlüssel-ID eines KMS-Schlüssels aus. Wählen Sie die Registerkarte Schlüsselmaterial und Rotationen. In der Tabelle mit den Schlüsselmaterialien werden alle Schlüsselmaterialien aufgeführt, die dem Schlüssel zugeordnet sind. Wählen Sie im Menü Aktionen in der Zeile, die dem Schlüsselmaterial entspricht, das Sie löschen möchten, die Option Schlüsselmaterial löschen.
5. Bestätigen Sie, dass Sie das Schlüsselmaterial löschen möchten, und klicken Sie dann auf Delete key material. Der Zustand des KMS-Schlüssels, der dem Schlüsselstatus entspricht, ändert sich in Pending import (Import ausstehend). Wenn sich das gelöschte Schlüsselmaterial im `PENDING_ROTATION` Status befand, ändert sich der Status des KMS-Schlüssels nicht.

Verwenden der AWS KMS API

Um die AWS KMS API zum Löschen von Schlüsselmaterial zu verwenden, senden Sie eine DeleteImportedKeyMaterialAnfrage. Das folgende Beispiel zeigt, wie Sie dies mit dem AWS CLI tun können.

Ersetzen Sie *1234abcd-12ab-34cd-56ef-1234567890ab* mit der Schlüssel-ID des KMS-Schlüssels, dessen Schlüsselmaterial Sie löschen möchten. Für diese Operation können Sie die Schlüssel-ID oder den ARN des KMS-Schlüssels verwenden, aber keinen Alias. Der folgende Befehl löscht das aktuelle Schlüsselmaterial, das möglicherweise das einzige Schlüsselmaterial ist, das dem Schlüssel zugeordnet ist.

```
$ aws kms delete-imported-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Um ein bestimmtes Schlüsselmaterial zu löschen, geben Sie das Schlüsselmaterial an, das mithilfe des `key-material-id` Parameters identifiziert wurde. *123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0*Ersetzen Sie es durch den Bezeichner des Schlüsselmaterials, das Sie löschen möchten.

```
$ aws kms delete-imported-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --key-material-id 123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0
```

Datenschlüssel generieren

Datenschlüssel sind symmetrische Schlüssel, mit denen Sie Daten verschlüsseln können. Dazu gehören große Datenmengen und andere Datenverschlüsselungsschlüssel. Im Gegensatz zu symmetrischen KMS-Schlüsseln, die nicht heruntergeladen werden können, werden Datenschlüssel an Sie zurückgegeben, damit Sie sie außerhalb von verwenden können. AWS KMS

Beim AWS KMS Generieren von Datenschlüsseln werden ein Klartext-Datenschlüssel zur sofortigen Verwendung (optional) und eine verschlüsselte Kopie des Datenschlüssels zurückgegeben, die Sie sicher zusammen mit den Daten speichern können. Wenn Sie bereit sind, die Daten zu entschlüsseln, bitten Sie zunächst darum, den verschlüsselten Datenschlüssel AWS KMS zu entschlüsseln.

AWS KMS generiert, verschlüsselt und entschlüsselt Datenschlüssel. AWS KMS Speichert, verwaltet oder verfolgt Ihre Datenschlüssel jedoch nicht und führt auch keine kryptografischen Operationen mit Datenschlüsseln durch. Sie müssen Datenschlüssel außerhalb von AWS KMS verwenden und verwalten. Hilfe bei der sicheren Verwendung von Datenschlüsseln finden Sie unter AWS Encryption SDK.

Themen

- Erstellen eines Datenschlüssels
- Wie funktionieren kryptografische Operationen mit Datenschlüsseln
- Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel

Erstellen eines Datenschlüssels

Rufen Sie den GenerateDataKeyVorgang auf, um einen Datenschlüssel zu erstellen. AWS KMS generiert den Datenschlüssel. Anschließend wird eine Kopie des Datenschlüssels unter dem von Ihnen angegebenen KMS-Schlüssel mit symmetrischer Verschlüsselung verschlüsselt. Diese Operation gibt eine Klartextkopie des Datenschlüssels und die unter dem KMS-Schlüssel verschlüsselte Kopie des Datenschlüssels zurück. Die folgende Abbildung zeigt diese Operation.

AWS KMS unterstützt auch den GenerateDataKeyWithoutPlaintextVorgang, der nur einen verschlüsselten Datenschlüssel zurückgibt. Wenn Sie den Datenschlüssel verwenden müssen, bitten Sie darum, ihn AWS KMS zu entschlüsseln.

Wie funktionieren kryptografische Operationen mit Datenschlüsseln

In den folgenden Themen wird erklärt, wie Datenschlüssel funktionieren, die durch eine GenerateDataKeyGenerateDataKeyWithoutPlaintextOR-Operation generiert wurden.

Verschlüsseln von Daten mit einem Datenschlüssel

AWS KMS kann keinen Datenschlüssel zum Verschlüsseln von Daten verwenden. Sie können den Datenschlüssel jedoch auch außerhalb von verwenden AWS KMS, z. B. mithilfe von OpenSSL oder einer kryptografischen Bibliothek wie der. AWS Encryption SDK

Entfernen Sie den Klartext-Datenschlüssel nach dem Verschlüsseln der Daten möglichst schnell aus dem Arbeitsspeicher. Sie können den verschlüsselten Datenschlüssel sicher zusammen mit den verschlüsselten Daten speichern, damit er zum Entschlüsseln der Daten verfügbar ist.

Entschlüsseln von Daten mit einem Datenschlüssel

Um Ihre Daten zu entschlüsseln, übergeben Sie den verschlüsselten Datenschlüssel an den Vorgang Decrypt. AWS KMS verwendet Ihren KMS-Schlüssel, um den Datenschlüssel zu entschlüsseln, und gibt dann den Klartext-Datenschlüssel zurück. Entschlüsseln Sie die Daten mit dem Klartext-Datenschlüssel und entfernen den Klartext-Datenschlüssel dann schnellstmöglich aus dem Arbeitsspeicher.

Das folgende Diagramm zeigt, wie Sie mit der Operation `Decrypt` einen verschlüsselten Datenschlüssel entschlüsseln.

Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel

Wenn ein KMS-Schlüssel unbrauchbar wird, wirkt sich das fast sofort aus (vorbehaltlich einer letztendlichen Konsistenz). Der Schlüsselstatus des KMS-Schlüssels ändert sich, um seinen neuen Zustand widerzuspiegeln, und alle Anforderungen der Verwendung des KMS-Schlüssels in kryptografischen Vorgängen schlagen fehl.

Die Auswirkungen auf die mit dem KMS-Schlüssel verschlüsselten Datenschlüssel und auf die mit dem Datenschlüssel verschlüsselten Daten werden jedoch so lange verzögert, bis der KMS-Schlüssel erneut verwendet wird, z. B. zur Entschlüsselung des Datenschlüssels.

KMS-Schlüssel können aus einer Vielzahl von Gründen unbrauchbar werden, einschließlich der folgenden Aktionen, die Sie möglicherweise durchführen:

- Deaktivieren des KMS-Schlüssels
- Planen der Löschung des KMS-Schlüssels
- Löschen des Schlüsselmaterials aus einem KMS-Schlüssel mit importiertem Schlüsselmaterial oder Ablaufenlassen des importierten Schlüsselmaterials Wenn einem KMS-Schlüssel mit `EXTERNAL` Ursprung mehrere Schlüsselmaterialien zugeordnet sind, führt das Löschen oder Ablaufen aller Schlüsselmaterialien dazu, dass der Schlüssel unbrauchbar wird.
- Trennen der Verbindung zum AWS CloudHSM Schlüsselspeicher, der den KMS-Schlüssel hostet, oder Löschen des Schlüssels aus dem AWS CloudHSM Cluster, der als Schlüsselmaterial für den KMS-Schlüssel dient.
- Trennen des externen Schlüsselspeichers, der den KMS-Schlüssel hostet, oder jede andere Aktion, die Verschlüsselungs- und Entschlüsselungsanforderungen an den Proxy des externen Schlüsselspeichers beeinträchtigt, einschließlich der Löschung des externen Schlüssels aus seinem externen Schlüsselmanager

Dieser Effekt ist besonders wichtig für viele Unternehmen, die Datenschlüssel verwenden AWS-Services , um die Ressourcen zu schützen, die der Dienst verwaltet. Das folgende Beispiel verwendet Amazon Elastic Block Store (Amazon EBS) und Amazon Elastic Compute Cloud (Amazon EC2). Verschiedene AWS-Services verwenden Datenschlüssel auf unterschiedliche Weise. Einzelheiten finden Sie im Abschnitt „Datenschutz“ des Kapitels „Sicherheit“ für den AWS-Service.

Betrachten Sie beispielsweise folgendes Szenario:

1. Sie erstellen ein verschlüsseltes EBS-Volume und geben einen KMS-Schlüssel an, um es zu schützen. Amazon EBS fordert AWS KMS auf, den KMS-Schlüssel zum Generieren eines verschlüsselten Datenschlüssels für das Volume zu verwenden. Amazon EBS speichert den verschlüsselten Datenschlüssel zusammen mit den Metadaten des Volumes.
2. Wenn Sie das EBS-Volume an eine EC2 Instance anhängen, EC2 verwendet Amazon Ihren KMS-Schlüssel, um den verschlüsselten Datenschlüssel des EBS-Volumes zu entschlüsseln. Amazon EC2 verwendet den Datenschlüssel in der Nitro-Hardware, die für die Verschlüsselung aller Festplatten I/O auf dem EBS-Volume verantwortlich ist. Der Datenschlüssel verbleibt in der Nitro-Hardware, solange das EBS-Volume an die Instance angehängt ist. EC2
3. Sie führen eine Aktion aus, die den KMS-Schlüssel unbrauchbar macht. Dies hat keine unmittelbaren Auswirkungen auf die EC2 Instance oder das EBS-Volume. Amazon EC2 verwendet

den Datenschlüssel — nicht den KMS-Schlüssel —, um die gesamte Festplatte zu verschlüsseln, I/O während das Volume mit der Instance verbunden ist.

4. Wenn das verschlüsselte EBS-Volume jedoch von der EC2 Instance getrennt wird, entfernt Amazon EBS den Datenschlüssel von der Nitro-Hardware. Wenn das verschlüsselte EBS-Volume das nächste Mal an eine EC2 Instance angehängt wird, schlägt der Anhang fehl, da Amazon EBS den KMS-Schlüssel nicht verwenden kann, um den verschlüsselten Datenschlüssel des Volumes zu entschlüsseln. Um das EBS-Volume wieder zu verwenden, müssen Sie den KMS-Schlüssel wieder brauchbar machen.

Generieren Sie Datenschlüsselpaare

Ein asymmetrischer KMS-Schlüssel steht für ein Datenschlüsselpaar. Datenschlüsselpaare sind asymmetrische Datenschlüssel, die aus einem mathematisch verwandten öffentlichen Schlüssel und privaten Schlüssel bestehen. Sie sind für die clientseitige Verschlüsselung und Entschlüsselung, Signierung und Überprüfung außerhalb von Peers oder zur Einrichtung eines AWS KMS gemeinsamen Geheimnisses zwischen zwei Peers konzipiert.

Im Gegensatz zu den Datenschlüsselpaaren, die Tools wie OpenSSL generieren, AWS KMS schützt es den privaten Schlüssel in jedem Datenschlüsselpaar unter einem von Ihnen angegebenen symmetrischen Verschlüsselungs-KMS-Schlüssel. AWS KMS AWS KMS Speichert, verwaltet oder verfolgt Ihre Datenschlüsselpaare jedoch nicht und führt auch keine kryptografischen Operationen mit Datenschlüsselpaaren durch. Sie müssen Datenschlüsselpaare außerhalb von verwenden und verwalten AWS KMS.

Themen

- Erstellen eines Datenschlüsselpaars
- So funktionieren kryptografische Operationen mit Datenschlüsselpaaren

Erstellen eines Datenschlüsselpaars

Um ein Datenschlüsselpaar zu erstellen, rufen Sie die GenerateDataKeyPairWithoutPlaintextOperationen GenerateDataKeyPairoder auf. Geben Sie den KMS-Schlüssel mit symmetrischer Verschlüsselung an, den Sie zum Verschlüsseln des privaten Schlüssels verwenden möchten.

`GenerateDataKeyPair` gibt einen öffentlichen Klartextschlüssel, einen privaten Klartextschlüssel und einen verschlüsselten privaten Schlüssel zurück. Verwenden Sie diese Operation, wenn Sie umgehend einen privaten Klartextschlüssel benötigen, z. B. um eine digitale Signatur zu generieren.

`GenerateDataKeyPairWithoutPlaintext` gibt einen öffentlichen Klartextschlüssel und einen verschlüsselten privaten Schlüssel zurück, aber keinen privaten Klartextschlüssel. Verwenden Sie diese Operation, wenn Sie nicht sofort einen privaten Klartextschlüssel benötigen, z. B. wenn Sie mit einem öffentlichen Schlüssel verschlüsseln. Später, wenn Sie einen privaten Klartextschlüssel benötigen, um die Daten zu entschlüsseln, können Sie die Operation Decrypt aufrufen.

Die folgende Abbildung zeigt Operation `GenerateDataKeyPair`. Die Operation `GenerateDataKeyPairWithoutPlaintext` lässt den privaten Klartextschlüssel aus.

So funktionieren kryptografische Operationen mit Datenschlüsselpaaren

In den folgenden Themen wird erklärt, welche kryptografischen Operationen Sie mit Datenschlüsselpaaren ausführen können, die durch eine GenerateDataKeyPairGenerateDataKeyPairWithoutPlaintextOR-Operation generiert wurden, und wie sie funktionieren.

Verschlüsseln von Daten mit einem Datenschlüsselpaar

Wenn Sie mit einem Datenschlüsselpaar verschlüsseln, verwenden Sie den öffentlichen Schlüssel des Paares, um die Daten zu verschlüsseln, und den privaten Schlüssel desselben Paares, um die Daten zu entschlüsseln. In der Regel werden Datenschlüsselpaare verwendet, wenn viele Parteien Daten verschlüsseln müssen, die nur die Partei im Besitz des privaten Schlüssels entschlüsseln darf.

Die Parteien mit dem öffentlichen Schlüssel verwenden diesen Schlüssel, um Daten zu verschlüsseln, wie im folgenden Diagramm dargestellt.

Entschlüsseln von Daten mit einem Datenschlüsselpaar

Um Ihre Daten zu entschlüsseln, verwenden Sie den privaten Schlüssel im Datenschlüsselpaar. Damit der Vorgang erfolgreich ausgeführt wird, müssen die öffentlichen und privaten Schlüssel aus demselben Datenschlüsselpaar stammen, und Sie müssen denselben Verschlüsselungsalgorithmus verwenden.

Um den verschlüsselten privaten Schlüssel zu entschlüsseln, übergeben Sie ihn an den Entschlüsselungsvorgang . Verwenden Sie den privaten Schlüssel, um die Daten zu entschlüsseln. Entfernen Sie dann den privaten Klartextschlüssel so schnell wie möglich aus dem Speicher.

Das folgende Diagramm zeigt, wie Sie den privaten Schlüssel in einem Datenschlüsselpaar verwenden, um Chiffretext zu entschlüsseln.

Signieren von Nachrichten mit einem Datenschlüsselpaar

Um eine kryptografische Signatur für eine Nachricht zu generieren, verwenden Sie den privaten Schlüssel im Datenschlüsselpaar. Jeder im Besitz des öffentlichen Schlüssels kann mit ihm überprüfen, dass die Nachricht mit Ihrem privaten Schlüssel signiert wurde und dass sie sich seit der Signatur nicht geändert hat.

Wenn Sie Ihren privaten Schlüssel verschlüsseln, geben Sie den verschlüsselten privaten Schlüssel an den Vorgang Entschlüsseln weiter. AWS KMS verwendet Ihren KMS-Schlüssel, um den Datenschlüssel zu entschlüsseln, und gibt dann den privaten Klartext-Schlüssel zurück. Verwenden Sie den privaten Klartextschlüssel, um die Signatur zu generieren. Entfernen Sie dann den privaten Klartextschlüssel so schnell wie möglich aus dem Speicher.

Um eine Nachricht zu signieren, erstellen Sie einen Message Digest mit einer kryptografischen Hash-Funktion, z. B. dem Befehl `dgst` in OpenSSL. Übergeben Sie dann Ihren privaten Klartextschlüssel an den Signaturalgorithmus. Das Ergebnis ist eine Signatur, die den Inhalt der Nachricht darstellt. (Möglicherweise können Sie kürzere Nachrichten signieren, ohne vorher einen Digest zu erstellen. Die maximale Nachrichtengröße variiert je nach verwendetem Signatur-Tool.)

Das folgende Diagramm zeigt, wie Sie den privaten Schlüssel in einem Datenschlüsselpaar verwenden, um eine Nachricht zu signieren.

Überprüfen einer Signatur mit einem Datenschlüsselpaar

Jeder, der über den öffentlichen Schlüssel in Ihrem Datenschlüsselpaar verfügt, kann damit die Signatur überprüfen, die Sie mit Ihrem privaten Schlüssel generiert haben. Die Verifizierung bestätigt, dass ein autorisierter Benutzer die Nachricht mit dem angegebenen privaten Schlüssel und dem Signaturalgorithmus signiert hat und sich die Nachricht seit der Signatur nicht geändert hat.

Um erfolgreich zu sein, muss die Partei, die die Signatur überprüft, denselben Digest-Typ generieren, denselben Algorithmus verwenden und den öffentlichen Schlüssel verwenden, der dem privaten Schlüssel entspricht, der zum Signieren der Nachricht verwendet wird.

Das folgende Diagramm zeigt, wie der öffentliche Schlüssel in einem Datenschlüsselpaar verwendet wird, um eine Nachrichtensignatur zu überprüfen.

Leiten Sie ein gemeinsames Geheimnis mit Datenschlüsselpaaren ab

Die Schlüsselvereinbarung ermöglicht es zwei Peers, von denen jeder über ein öffentlich-privates key pair mit elliptischer Kurve verfügt, ein gemeinsames Geheimnis über einen unsicheren Kanal einzurichten. Um ein gemeinsames Geheimnis abzuleiten, müssen die beiden Peers ihre öffentlichen Schlüssel über den unsicheren Kommunikationskanal (wie das Internet) austauschen. Anschließend verwendet jede Partei ihren privaten Schlüssel und den öffentlichen Schlüssel ihres Peers, um dasselbe gemeinsame Geheimnis mithilfe eines Schlüsselvereinbarungsalgorithmus zu berechnen. Sie können den Wert des gemeinsamen geheimen Schlüssels verwenden, um einen symmetrischen Schlüssel abzuleiten, mit dem Daten, die zwischen den beiden Peers gesendet werden, ver- und entschlüsselt oder generiert und verifiziert werden können. HMACs

> **Note**
>
> AWS KMS empfiehlt dringend, zu überprüfen, ob der öffentliche Schlüssel, den Sie erhalten, von der erwarteten Partei stammt, bevor Sie ihn zur Ableitung eines gemeinsamen geheimen Schlüssels verwenden.

Führen Sie Offline-Operationen mit öffentlichen Schlüsseln durch

Bei einem asymmetrischen KMS-Schlüssel wird der private Schlüssel erstellt AWS KMS und verlässt AWS KMS ihn nie unverschlüsselt. Um den privaten Schlüssel zu verwenden, müssen Sie anrufen. AWS KMS Sie können den öffentlichen Schlüssel darin verwenden, AWS KMS indem Sie die AWS KMS API-Operationen aufrufen. Oder Sie können den öffentlichen Schlüssel herunterladen und für die Verwendung außerhalb von freigeben AWS KMS.

Sie können einen öffentlichen Schlüssel teilen, damit andere Daten verschlüsseln können AWS KMS , die Sie nur mit Ihrem privaten Schlüssel entschlüsseln können. Oder, um anderen zu erlauben, eine digitale Signatur, die Sie mit Ihrem privaten Schlüssel generiert haben, außerhalb von AWS KMS zu überprüfen. Oder um Ihren öffentlichen Schlüssel mit einem Peer zu teilen, um daraus ein gemeinsames Geheimnis abzuleiten.

Wenn Sie den öffentlichen Schlüssel in Ihrem internen asymmetrischen KMS-Schlüssel verwenden AWS KMS, profitieren Sie von der Authentifizierung, Autorisierung und Protokollierung, die Teil jedes AWS KMS Vorgangs sind. Außerdem reduzieren Sie das Risiko, Daten zu verschlüsseln, die nicht entschlüsselt werden können. Diese Funktionen sind außerhalb von AWS KMS nicht wirksam. Details hierzu finden Sie unter Besondere Überlegungen zum Herunterladen öffentlicher Schlüssel.

> **Tip**
>
> Suchen Sie nach Datenschlüsseln oder SSH-Schlüsseln? In diesem Thema wird erläutert, wie Sie asymmetrische Schlüssel in AWS Key Management Service verwalten, bei denen der private Schlüssel nicht exportierbar ist. Informationen zu exportierbaren Datenschlüsselpaaren, bei denen der private Schlüssel durch einen KMS-Schlüssel mit symmetrischer Verschlüsselung geschützt ist, finden Sie unter. GenerateDataKeyPair Hilfe zum Herunterladen des öffentlichen Schlüssels, der mit einer EC2 Amazon-Instance verknüpft ist, finden Sie unter Abrufen des öffentlichen Schlüssels im EC2 Amazon-Benutzerhandbuch und im EC2 Amazon-Benutzerhandbuch.

Themen

- Besondere Überlegungen zum Herunterladen öffentlicher Schlüssel
- Laden Sie den öffentlichen Schlüssel herunter

- Beispiel für Offline-Operationen

Besondere Überlegungen zum Herunterladen öffentlicher Schlüssel

Zum Schutz Ihrer KMS-Schlüssel AWS KMS bietet es Zugriffskontrollen, authentifizierte Verschlüsselung und detaillierte Protokolle aller Vorgänge. AWS KMS ermöglicht es Ihnen auch, die Verwendung von KMS-Schlüsseln vorübergehend oder dauerhaft zu verhindern. Schließlich sind die AWS KMS Operationen darauf ausgelegt, das Risiko der Verschlüsselung von Daten, die nicht entschlüsselt werden können, zu minimieren. Diese Funktionen sind nicht verfügbar, wenn Sie heruntergeladene öffentliche Schlüssel außerhalb von verwenden. AWS KMS

Autorisierung

Wichtige Richtlinien und IAM-Richtlinien, die den Zugriff auf den internen KMS-Schlüssel steuern, AWS KMS haben keine Auswirkungen auf Vorgänge, die außerhalb von AWS ausgeführt werden. Jeder Benutzer, der Zugriff auf den öffentlichen Schlüssel hat, kann ihn außerhalb von verwenden, AWS KMS auch wenn er nicht berechtigt ist, Daten zu verschlüsseln oder Signaturen mit dem KMS-Schlüssel zu überprüfen.

Nutzungsbeschränkungen für Schlüssel

Einschränkungen bei der Verwendung von Schlüsseln sind außerhalb von AWS KMS nicht wirksam. Wenn Sie den Verschlüsselungsvorgang mit einem KMS-Schlüssel aufrufen, der den Wert `KeyUsage` von hat`SIGN_VERIFY`, schlägt der AWS KMS Vorgang fehl. Wenn Sie Daten jedoch außerhalb oder AWS KMS mit einem öffentlichen Schlüssel aus einem KMS-Schlüssel mit einem `KeyUsage` von `SIGN_VERIFY` oder verschlüsseln`KEY_AGREEMENT`, können die Daten nicht entschlüsselt werden.

Algorithmusbeschränkungen

Einschränkungen der AWS KMS unterstützten Verschlüsselungs- und Signaturalgorithmen sind außerhalb von nicht wirksam. AWS KMS Wenn Sie Daten mit dem öffentlichen Schlüssel eines externen KMS-Schlüssels verschlüsseln und einen Verschlüsselungsalgorithmus verwenden AWS KMS, der AWS KMS dies nicht unterstützt, können die Daten nicht entschlüsselt werden.

Deaktivieren und Löschen von KMS-Schlüsseln

Maßnahmen, die Sie ergreifen können, um die Verwendung des KMS-Schlüssels bei einem kryptografischen Vorgang innerhalb von zu verhindern, hindern AWS KMS niemanden daran, den öffentlichen Schlüssel außerhalb von zu verwenden. AWS KMS Beispielsweise hat das

Deaktivieren eines KMS-Schlüssels, das Planen des Löschens eines KMS-Schlüssels, das Löschen eines KMS-Schlüssels oder das Löschen des Schlüsselmaterials aus einem KMS-Schlüssel keine Auswirkungen auf einen öffentlichen Schlüssel außerhalb von AWS KMS. Wenn Sie einen asymmetrischen KMS-Schlüssel löschen oder sein Schlüsselmaterial löschen oder verlieren, können Daten, die Sie mit einem externen öffentlichen Schlüssel verschlüsseln, nicht wiederhergestellt werden. AWS KMS

Protokollierung

AWS CloudTrail Protokolle, die jeden AWS KMS Vorgang aufzeichnen, einschließlich der Anfrage, der Antwort, des Datums, der Uhrzeit und des autorisierten Benutzers, zeichnen nicht die Verwendung des öffentlichen Schlüssels außerhalb von auf. AWS KMS

Offline-Überprüfung mit SM2 Schlüsselpaaren (nur Regionen China)

Um eine Signatur außerhalb oder AWS KMS mit einem SM2 öffentlichen Schlüssel zu verifizieren, müssen Sie die identifizierende ID angeben. Wird standardmäßig `1234567812345678` als Unterscheidungs-ID AWS KMS verwendet. Weitere Informationen finden Sie unter Offline-Überprüfung mit SM2 Schlüsselpaaren (nur Regionen China).

Laden Sie den öffentlichen Schlüssel herunter

Sie können den öffentlichen Schlüssel von einem asymmetrischen KMS-Schlüsselpaar in der AWS KMS Konsole oder mithilfe des GetPublicKeyVorgangs herunterladen. Um den öffentlichen Schlüssel herunterzuladen, benötigen Sie die entsprechenden `kms:GetPublicKey` Berechtigungen für den asymmetrischen KMS-Schlüssel.

Der AWS KMS zurückgegebene öffentliche Schlüssel ist ein DER-codierter öffentlicher X.509-Schlüssel, auch bekannt als `SubjectPublicKeyInfo` (SPKI), wie in RFC 5280 definiert. Wenn Sie die HTTP-API oder die verwenden, ist der Wert Base64-codiert AWS CLI. Andernfalls ist er nicht Base64-codiert.

Um den öffentlichen Schlüssel von einem asymmetrischen KMS-Schlüsselpaar herunterzuladen, benötigen Sie `kms:GetPublicKey` Berechtigungen. Weitere Informationen zu AWS KMS Berechtigungen finden Sie unterBerechtigungsreferenz.

Verwenden der AWS KMS Konsole

Mit dem können Sie AWS Management Console den öffentlichen Schlüssel von einem asymmetrischen KMS-Schlüssel in Ihrem AWS-Konto anzeigen, kopieren und herunterladen.

Verwenden Sie die API, um den öffentlichen Schlüssel von einem asymmetrischen KMS-Schlüssel in einem anderen AWS-Konto Format herunterzuladen. AWS KMS

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel.
4. Wählen Sie den Alias oder die Schlüssel-ID eines asymmetrischen KMS-Schlüssels aus.
5. Wählen Sie die Registerkarte Cryptographic configuration (kryptografische Konfiguration) aus. Notieren Sie sich die Werte der Felder Key spec (Schlüsselspezifikation), Key usage (Schlüsselnutzung) und Encryption algorithms (Verschlüsselungsalgorithmen) oder Signing Algorithms (Signaturalgorithmen). Sie müssen diese Werte verwenden, um den öffentlichen Schlüssel außerhalb von zu verwenden. AWS KMS Stellen Sie sicher, dass Sie diese Informationen freigeben, wenn Sie den öffentlichen Schlüssel freigeben.
6. Wählen Sie die Registerkarte Public key (Öffentlicher Schlüssel).
7. Um den öffentlichen Schlüssel in die Zwischenablage zu kopieren, wählen Sie Copy (Kopieren). Um den öffentlichen Schlüssel in eine Datei herunterzuladen, wählen Sie Download (Herunterladen).

Verwenden der AWS KMS API

Der GetPublicKeyVorgang gibt den öffentlichen Schlüssel in einem asymmetrischen KMS-Schlüssel zurück. Außerdem werden wichtige Informationen zurückgegeben, die Sie benötigen, um den öffentlichen Schlüssel außerhalb korrekt zu verwenden AWS KMS, einschließlich der Schlüsselverwendung und der Verschlüsselungsalgorithmen. Achten Sie darauf, diese Werte zu speichern und sie freizugeben, wenn Sie den öffentlichen Schlüssel freigeben.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Um einen KMS-Schlüssel anzugeben, verwenden Sie seine Schlüssel-ID, seinen Schlüssel-ARN, seinen Aliasnamen oder seinen Alias-ARN. Wenn Sie einen Aliasnamen verwenden, stellen Sie ihm alias/ voran. Um einen KMS-Schlüssel in einem anderen anzugeben AWS-Konto, müssen Sie dessen Schlüssel-ARN oder Alias-ARN verwenden.

Bevor Sie diesen Befehl ausführen, ersetzen Sie den Beispiel-Aliasnamen durch einen gültigen Bezeichner für den KMS-Schlüssel. Um diesen Befehl auszuführen, müssen Sie über `kms:GetPublicKey`-Berechtigungen für den KMS-Schlüssel verfügen.

```
$ aws kms get-public-key --key-id alias/example_RSA_3072

{
    "KeySpec": "RSA_3072",
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "EncryptionAlgorithms": [
        "RSAES_OAEP_SHA_1",
        "RSAES_OAEP_SHA_256"
    ],
    "PublicKey": "MIIBojANBgkqhkiG..."
}
```

Beispiel für Offline-Operationen

Nachdem Sie den öffentlichen Schlüssel Ihres asymmetrischen KMS-Schlüsselpaars heruntergeladen haben, können Sie ihn mit anderen teilen und für Offline-Operationen verwenden.

AWS CloudTrail Protokolle, die jeden AWS KMS Vorgang aufzeichnen, einschließlich der Anfrage, der Antwort, des Datums, der Uhrzeit und des autorisierten Benutzers, zeichnen nicht die Verwendung des öffentlichen Schlüssels außerhalb von AWS KMS auf.

Dieses Thema enthält Beispiele für Offline-Operationen und Einzelheiten, die die Tools AWS KMS bereitstellen, um Offline-Operationen zu vereinfachen.

Themen

- Offline-Ableitung gemeinsam genutzter Geheimnisse
- Offline-Überprüfung mit ML-DSA-Schlüsselpaaren
- Offline-Überprüfung mit SM2 Schlüsselpaaren (nur Regionen China)

Offline-Ableitung gemeinsam genutzter Geheimnisse

Sie können den öffentlichen Schlüssel Ihres ECC-Schlüsselpaars herunterladen, um ihn im Offline-Betrieb zu verwenden, d. h. bei Vorgängen außerhalb von AWS KMS.

Die folgende OpenSSL-Komplettlösung zeigt eine Methode zum Ableiten eines gemeinsamen geheimen Schlüssels außerhalb der AWS KMS Verwendung des öffentlichen Schlüssels eines ECC-KMS-Schlüsselpaars und eines mit OpenSSL erstellten privaten Schlüssels.

1. Erstellen Sie ein ECC-Schlüsselpaar in OpenSSL und bereiten Sie es für die Verwendung mit vor. AWS KMS

```
// Create an ECC key pair in OpenSSL and save the private key in
 openssl_ecc_key_priv.pem
export OPENSSL_CURVE_NAME="P-256"
export KMS_CURVE_NAME="ECC_NIST_P256"

export OPENSSL_KEY1_PRIV_PEM="openssl_ecc_key1_priv.pem"
openssl ecparam -name ${OPENSSL_CURVE_NAME} -genkey -out ${OPENSSL_KEY1_PRIV_PEM}


// Derive the public key from the private key
export OPENSSL_KEY1_PUB_PEM="openssl_ecc_key1_pub.pem"
openssl ec -in ${OPENSSL_KEY1_PRIV_PEM} -pubout -outform pem \
    -out ${OPENSSL_KEY1_PUB_PEM}

// View the PEM file containing the public key and extract the public key as a
// Base64 encoded string into OPENSSL_KEY1_PUB_BASE64 for use with AWS KMS
export OPENSSL_KEY1_PUB_BASE64=`cat ${OPENSSL_KEY1_PUB_PEM} | \
    tee /dev/stderr | grep -v "PUBLIC KEY" | tr -d "\n"`
```

2. Erstellen Sie ein key pair für die ECC-Schlüsselvereinbarung AWS KMS und bereiten Sie es für die Verwendung mit OpenSSL vor.

```
// Create a KMS key on the same curve as the key pair from step 1
// with a key usage of KEY_AGREEMENT
// Save its ARN in KMS_KEY1_ARN.
export KMS_KEY1_ARN=`aws kms create-key --key-spec ${KMS_CURVE_NAME} \
    --key-usage KEY_AGREEMENT | tee /dev/stderr | jq -r .KeyMetadata.Arn`

// Download the public key and save the Base64-encoded version in KMS_KEY1_PUB_BASE64

export KMS_KEY1_PUB_BASE64=`aws kms get-public-key --key-id ${KMS_KEY1_ARN} | \
    tee /dev/stderr | jq -r .PublicKey`

// Create a PEM file for the public KMS key for use with OpenSSL
export KMS_KEY1_PUB_PEM="aws_kms_ecdh_key1_pub.pem"
```

```
echo "-----BEGIN PUBLIC KEY-----" > ${KMS_KEY1_PUB_PEM}
echo ${KMS_KEY1_PUB_BASE64} | fold -w 64 >> ${KMS_KEY1_PUB_PEM}
echo "-----END PUBLIC KEY-----" >> ${KMS_KEY1_PUB_PEM}
```

3. Leiten Sie Shared Secret in OpenSSL mithilfe des privaten Schlüssels in OpenSSL und des öffentlichen KMS-Schlüssels ab.

```
export OPENSSL_SHARED_SECRET1_BIN="openssl_shared_secret1.bin"
openssl pkeyutl -derive -inkey ${OPENSSL_KEY1_PRIV_PEM} \
    -peerkey ${KMS_KEY1_PUB_PEM} -out ${OPENSSL_SHARED_SECRET1_BIN}
```

Offline-Überprüfung mit ML-DSA-Schlüsselpaaren

AWS KMS unterstützt eine abgesicherte Variante der ML-DSA-Signatur, wie in Abschnitt 3.4 der Federal Information Processing Standards (FIPS) 204 beschrieben, für Nachrichten mit bis zu 4 KB Byte.

Um Nachrichten zu signieren, die größer als 4 KB sind, führen Sie den Schritt zur Nachrichtenvorverarbeitung außerhalb von aus. AWS KMS Dieser Hashing-Schritt erstellt eine 64-Byte-Nachricht, die für μ steht, wie in NIST FIPS 204, Abschnitt 6.2 definiert.

AWS KMS hat einen Nachrichtentyp, der `EXTERNAL_MU` für Nachrichten mit mehr als 4 KB aufgerufen wird. Wenn Sie diesen anstelle des `RAW` Nachrichtentyps verwenden, gilt Folgendes AWS KMS:

- Geht davon aus, dass Sie den Hashing-Schritt bereits ausgeführt haben
- Überspringt den internen Hashing-Prozess
- Funktioniert mit Nachrichten beliebiger Größe

Wenn Sie eine Nachricht verifizieren, hängt die verwendete Methode von der Größenbeschränkung des externen Systems oder der Bibliothek ab und davon, ob sie den 64-Byte-Nachrichtenrepräsentanten μ unterstützt:

- Wenn die Nachricht kleiner als die Größenbeschränkung ist, verwenden Sie den `RAW` Nachrichtentyp.
- Wenn die Nachricht die Größenbeschränkung überschreitet, verwenden Sie den repräsentativen Wert μ im externen System.

In den folgenden Abschnitten wird gezeigt, wie Nachrichten mit OpenSSL signiert AWS KMS und überprüft werden. Wir bieten Beispiele für Nachrichten, die unter und über der Nachrichtengrößenbeschränkung von AWS KMS 4 KB liegen. OpenSSL legt keine Beschränkung der Nachrichtengröße für die Überprüfung fest.

Holen Sie sich für beide Beispiele zuerst den öffentlichen Schlüssel von AWS KMS. Verwenden Sie den folgenden AWS CLI -Befehl:

```
aws kms get-public-key \
    --key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \
    --output text \
    --query PublicKey | base64 --decode > public_key.der
```

Nachrichtengröße weniger als 4 KB

Verwenden Sie für Nachrichten unter 4 KB den `RAW` Nachrichtentyp mit AWS KMS. Sie können dies zwar verwenden`EXTERNAL_MU`, für Nachrichten innerhalb der Größenbeschränkung ist dies jedoch nicht erforderlich.

Verwenden Sie den folgenden AWS CLI Befehl, um die Nachricht zu signieren:

```
aws kms sign \
    --key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \
    --message 'your message' \
    --message-type RAW \
    --signing-algorithm ML_DSA_SHAKE_256 \
    --output text \
    --query Signature | base64 --decode > ExampleSignature.bin
```

Verwenden Sie den folgenden Befehl, um diese Nachricht mit OpenSSL zu überprüfen:

```
echo -n 'your message' | ./openssl dgst -verify public_key.der -signature
 ExampleSignature.bin
```

Nachrichtengröße mehr als 4 KB

Verwenden Sie den `EXTERNAL_MU` Nachrichtentyp, um Nachrichten zu signieren, die größer als 4 KB sind. Bei der Verwendung `EXTERNAL_MU` wird die Nachricht extern an einen 64-Byte-Repräsentanten μ vorgehasht, wie in NIST FIPS 204 Abschnitt 6.2 definiert, und sie an die Signier-

oder Verifizierungsvorgänge übergeben. Beachten Sie, dass sich dies von dem in NIST FIPS 204 Abschnitt 5.4 definierten „Pre-Hash MLDSA“ oder HashML-DSA unterscheidet.

1. Konstruieren Sie zunächst ein Nachrichtenpräfix. Das Präfix enthält ein Domänentrennzeichen, die Länge eines beliebigen Kontextes und den Kontext. Die Standardeinstellung für das Domänentrennzeichen und die Kontextlänge ist Null.
2. Stellen Sie der Nachricht das Nachrichtenpräfix voran.
3. Verwenden Sie diese Option SHAKE256 , um den öffentlichen Schlüssel zu hashen und ihn dem Ergebnis von Schritt 2 voranzustellen.
4. Hashen Sie abschließend das Ergebnis von Schritt 3, um ein `EXTERNAL_MU` 64-Byte-Ergebnis zu erhalten.

Das folgende Beispiel verwendet OpenSSL 3.5, um das `EXTERNAL_MU` zu erstellen:

```
{
    openssl asn1parse -inform DER -in public_key.der -strparse 17 -noout -out - 2>/dev/
null |
    openssl dgst -provider default -shake256 -xoflen 64 -binary;
    printf '\x00\x00';
    echo -n "your message"
} | openssl dgst -provider default -shake256 -xoflen 64 -binary > mu.bin
```

Nachdem Sie die `mu.bin` Datei erstellt haben, rufen Sie die AWS KMS API mit dem folgenden Befehl auf, um die Nachricht zu signieren:

```
aws kms sign \
    --key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \
    --message fileb://mu.bin \
    --message-type EXTERNAL_MU \
    --signing-algorithm ML_DSA_SHAKE_256 \
    --output text \
    --query Signature | base64 --decode > ExampleSignature.bin
```

Die resultierende Signatur entspricht der `RAW` Signatur der Originalnachricht. Sie können denselben OpenSSL 3.5-Befehl verwenden, um die Nachricht zu überprüfen:

```
echo -n 'your message' | ./openssl dgst -verify public_key.der -signature
 ExampleSignature.bin
```

Offline-Überprüfung mit SM2 Schlüsselpaaren (nur Regionen China)

Um eine Signatur außerhalb oder AWS KMS mit einem SM2 öffentlichen Schlüssel zu verifizieren, müssen Sie die identifizierende ID angeben. Wenn Sie eine Rohnachricht an die Sign-API übergeben `MessageType:RAW`, AWS KMS verwendet sie die standardmäßige Unterscheidungs-ID`1234567812345678`, die von OSCCA in 0009-2012 definiert wurde. GM/T Sie können nicht Ihre eigene unterscheidende ID in AWS KMS angeben.

Wenn Sie jedoch einen Nachrichtenüberblick außerhalb von generieren AWS, können Sie Ihre eigene Unterscheidungs-ID angeben und dann den Nachrichtenüberblick,, zum Signieren an übergeben. `MessageType:DIGEST` AWS KMS Dafür ändern Sie den `DEFAULT_DISTINGUISHING_ID`-Wert in der `SM2OfflineOperationHelper`-Klasse. Die von Ihnen angegebene Unterscheidungs-ID kann eine beliebige Zeichenfolge mit einer Länge von bis zu 8.192 Zeichen sein. Nachdem Sie den Message Digest AWS KMS signiert haben, benötigen Sie entweder den Message Digest oder die Nachricht und die Unterscheidungs-ID, mit der der Digest berechnet wurde, um ihn offline zu verifizieren.

> **Important**
>
> Der `SM2OfflineOperationHelper`-Referenzcode ist so konzipiert, dass er kompatibel mit Bouncy CastleVersion 1.68 ist. Für Hilfe zu anderen Versionen wenden Sie sich an bouncycastle.org.

`SM2OfflineOperationHelper`-Klasse

Um Ihnen bei Offline-Operationen mit SM2 Schlüsseln zu helfen, verfügt die `SM2OfflineOperationHelper` Klasse für Java über Methoden, die die Aufgaben für Sie ausführen. Sie können diese Hilfsklasse als Modell für andere kryptografische Anbieter verwenden.

Darin werden die AWS KMS unformatierten Chiffretext-Konvertierungen und die SM2 DSA-Message-Digest-Berechnungen automatisch durchgeführt. Nicht alle Kryptografieanbieter implementieren SM2 sie auf die gleiche Weise. Einige Bibliotheken, wie OpenSSL-Versionen 1.1.1 und höher, führen diese Aktionen automatisch aus. AWS KMS bestätigte dieses Verhalten beim Testen mit OpenSSL Version 3.0. Verwenden Sie die folgende `SM2OfflineOperationHelper`-Klasse mit Bibliotheken, wie Bouncy Castle, bei denen Sie diese Konvertierungen und Berechnungen manuell durchführen müssen.

Die `SM2OfflineOperationHelper`-Klasse bietet Methoden für die folgenden Offline-Vorgänge:

- Message-Digest-Berechnung

  Verwenden Sie die Methode, um offline einen Nachrichtenüberblick zu generieren, den Sie für die Offline-Überprüfung verwenden oder an den `calculateSM2Digest` Sie AWS KMS zum Signieren übergeben können. Die `calculateSM2Digest` Methode generiert einen Nachrichtenüberblick mit dem SM3 Hashing-Algorithmus. Die GetPublicKeyAPI gibt Ihren öffentlichen Schlüssel im Binärformat zurück. Sie müssen den Binärschlüssel in ein PublicKey Java-Format parsen. Stellen Sie den geparsten öffentlichen Schlüssel mit der Nachricht bereit. Die Methode kombiniert Ihre Nachricht automatisch mit der standardmäßigen Unterscheidungs-ID, `1234567812345678`, aber Sie können Ihre eigene Unterscheidungs-ID festlegen, indem Sie den `DEFAULT_DISTINGUISHING_ID`-Wert ändern.

- Verify

  Um eine Signatur offline zu prüfen, verwenden Sie die `offlineSM2DSAVerify`-Methode. Die `offlineSM2DSAVerify`-Methode verwendet den Message Difest, der anhand der angegebenen unterscheidenden ID berechnet wurde, und die ursprüngliche Nachricht, die Sie zur Überprüfung der digitalen Signatur bereitstellen. Die GetPublicKeyAPI gibt Ihren öffentlichen Schlüssel im Binärformat zurück. Sie müssen den Binärschlüssel in ein PublicKey Java-Format parsen. Stellen Sie den geparsten öffentlichen Schlüssel mit der ursprünglichen Nachricht und der Signatur, die Sie überprüfen möchten, bereit. Weitere Informationen finden Sie unter Offline-Überprüfung mit SM2 Schlüsselpaaren.

- Encrypt

  Um Klartext offline zu verschlüsseln, verwenden Sie die `offlineSM2PKEEncrypt`-Methode. Diese Methode stellt sicher, dass der Chiffretext in einem Format vorliegt, das entschlüsselt AWS KMS werden kann. Die `offlineSM2PKEEncrypt` Methode verschlüsselt den Klartext und konvertiert dann den von PKE erzeugten Rohchiffretext in das ASN.1-Format. SM2 Die GetPublicKeyAPI gibt Ihren öffentlichen Schlüssel im Binärformat zurück. Sie müssen den Binärschlüssel in ein PublicKey Java-Format parsen. Versehen Sie den geparsten öffentlichen Schlüssel mit dem Klartext, den Sie verschlüsseln möchten.

  Wenn Sie sich nicht sicher sind, ob Sie die Konvertierung durchführen müssen, verwenden Sie den folgenden OpenSSL-Vorgang, um das Format Ihres Geheimtextes zu testen. Wenn der Vorgang fehlschlägt, müssen Sie den Geheimtext in das ASN.1-Format konvertieren.

  ```
  openssl asn1parse -inform DER -in ciphertext.der
  ```

Standardmäßig verwendet die `SM2OfflineOperationHelper` Klasse beim Generieren von Nachrichtenübersichten für `1234567812345678` DSA-Operationen die standardmäßige Unterscheidungs-ID. SM2

```
package com.amazon.kms.utils;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.IOException;
import java.math.BigInteger;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;

import org.bouncycastle.crypto.CryptoException;
import org.bouncycastle.jce.interfaces.ECPublicKey;

import java.util.Arrays;

import org.bouncycastle.asn1.ASN1EncodableVector;
import org.bouncycastle.asn1.ASN1Integer;
import org.bouncycastle.asn1.DEROctetString;
import org.bouncycastle.asn1.DERSequence;
import org.bouncycastle.asn1.gm.GMNamedCurves;
import org.bouncycastle.asn1.x9.X9ECParameters;
import org.bouncycastle.crypto.CipherParameters;
import org.bouncycastle.crypto.params.ParametersWithID;
import org.bouncycastle.crypto.params.ParametersWithRandom;
import org.bouncycastle.crypto.signers.SM2Signer;
import org.bouncycastle.jcajce.provider.asymmetric.util.ECUtil;

public class SM2OfflineOperationHelper {
    // You can change the DEFAULT_DISTINGUISHING_ID value to set your own
 distinguishing ID,
    // the DEFAULT_DISTINGUISHING_ID can be any string up to 8,192 characters long.
```

```
   private static final byte[] DEFAULT_DISTINGUISHING_ID =
"1234567812345678".getBytes(StandardCharsets.UTF_8);
   private static final X9ECParameters SM2_X9EC_PARAMETERS =
GMNamedCurves.getByName("sm2p256v1");

   // ***calculateSM2Digest***
   // Calculate message digest
   public static byte[] calculateSM2Digest(final PublicKey publicKey, final byte[]
message) throws
           NoSuchProviderException, NoSuchAlgorithmException {
       final ECPublicKey ecPublicKey = (ECPublicKey) publicKey;

       // Generate SM3 hash of default distinguishing ID, 1234567812345678
       final int entlenA = DEFAULT_DISTINGUISHING_ID.length * 8;
       final byte [] entla = new byte[] { (byte) (entlenA & 0xFF00), (byte) (entlenA &
0x00FF) };
       final byte [] a = SM2_X9EC_PARAMETERS.getCurve().getA().getEncoded();
       final byte [] b = SM2_X9EC_PARAMETERS.getCurve().getB().getEncoded();
       final byte [] xg = SM2_X9EC_PARAMETERS.getG().getXCoord().getEncoded();
       final byte [] yg = SM2_X9EC_PARAMETERS.getG().getYCoord().getEncoded();
       final byte[] xa = ecPublicKey.getQ().getXCoord().getEncoded();
       final byte[] ya = ecPublicKey.getQ().getYCoord().getEncoded();
       final byte[] za = MessageDigest.getInstance("SM3", "BC")
               .digest(ByteBuffer.allocate(entla.length +
DEFAULT_DISTINGUISHING_ID.length + a.length + b.length + xg.length + yg.length +
                       xa.length +
ya.length).put(entla).put(DEFAULT_DISTINGUISHING_ID).put(a).put(b).put(xg).put(yg).put(xa).put
                       .array());

       // Combine hashed distinguishing ID with original message to generate final
digest
       return MessageDigest.getInstance("SM3", "BC")
               .digest(ByteBuffer.allocate(za.length +
message.length).put(za).put(message)
                       .array());
   }

   // ***offlineSM2DSAVerify***
   // Verify digital signature with SM2 public key
   public static boolean offlineSM2DSAVerify(final PublicKey publicKey, final byte []
message,
           final byte [] signature) throws InvalidKeyException {
       final SM2Signer signer = new SM2Signer();
```

```
        CipherParameters cipherParameters =
ECUtil.generatePublicKeyParameter(publicKey);
        cipherParameters = new ParametersWithID(cipherParameters,
DEFAULT_DISTINGUISHING_ID);
        signer.init(false, cipherParameters);
        signer.update(message, 0, message.length);
        return signer.verifySignature(signature);
    }

    // ***offlineSM2PKEEncrypt***
    // Encrypt data with SM2 public key
    public static byte[] offlineSM2PKEEncrypt(final PublicKey publicKey, final byte []
plaintext) throws
            NoSuchPaddingException, NoSuchAlgorithmException, NoSuchProviderException,
InvalidKeyException,
            BadPaddingException, IllegalBlockSizeException, IOException {
        final Cipher sm2Cipher = Cipher.getInstance("SM2", "BC");
        sm2Cipher.init(Cipher.ENCRYPT_MODE, publicKey);

        // By default, Bouncy Castle returns raw ciphertext in the c1c2c3 format
        final byte [] cipherText = sm2Cipher.doFinal(plaintext);

        // Convert the raw ciphertext to the ASN.1 format before passing it to AWS KMS
        final ASN1EncodableVector asn1EncodableVector = new ASN1EncodableVector();
        final int coordinateLength = (SM2_X9EC_PARAMETERS.getCurve().getFieldSize() +
7) / 8 * 2 + 1;
        final int sm3HashLength = 32;
        final int xCoordinateInCipherText = 33;
        final int yCoordinateInCipherText = 65;
        byte[] coords = new byte[coordinateLength];
        byte[] sm3Hash = new byte[sm3HashLength];
        byte[] remainingCipherText = new byte[cipherText.length - coordinateLength -
sm3HashLength];

        // Split components out of the ciphertext
        System.arraycopy(cipherText, 0, coords, 0, coordinateLength);
        System.arraycopy(cipherText, cipherText.length - sm3HashLength, sm3Hash, 0,
sm3HashLength);
        System.arraycopy(cipherText, coordinateLength, remainingCipherText,
0,cipherText.length - coordinateLength - sm3HashLength);

        // Build standard SM2PKE ASN.1 ciphertext vector
        asn1EncodableVector.add(new ASN1Integer(new BigInteger(1,
Arrays.copyOfRange(coords, 1, xCoordinateInCipherText))));
```

```
        asn1EncodableVector.add(new ASN1Integer(new BigInteger(1,
 Arrays.copyOfRange(coords, xCoordinateInCipherText, yCoordinateInCipherText))));
        asn1EncodableVector.add(new DEROctetString(sm3Hash));
        asn1EncodableVector.add(new DEROctetString(remainingCipherText));

        return new DERSequence(asn1EncodableVector).getEncoded("DER");
    }
}
```

Überwachen AWS KMS keys

Die Überwachung ist ein wichtiger Bestandteil, um die Verfügbarkeit, den Zustand und die Nutzung Ihres AWS KMS keys Ins zu verstehen AWS KMS und die Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer AWS Lösungen aufrechtzuerhalten. Durch das Sammeln von Überwachungsdaten aus allen Teilen Ihrer AWS -Lösung können Sie einen Multipoint-Fehler debuggen, falls ein solcher auftritt. Bevor Sie mit der Überwachung Ihrer KMS-Schlüssel beginnen, erstellen Sie einen Überwachungsplan, der Antworten auf folgende Fragen enthält:

- Was sind Ihre Ziele bei der Überwachung?
- Welche Ressourcen werden überwacht?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungswerkzeugte verwenden Sie?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer sollte benachrichtigt werden, wenn etwas passiert?

Als nächsten Schritt überwachen Sie Ihre KMS-Schlüssel, um eine Basis für die normale AWS KMS -Nutzung und Erwartungen in Ihrer Umgebung zu erstellen. Während Sie Ihre KMS-Schlüssel überwachen, speichern Sie historische Überwachungsdaten, damit Sie sie mit aktuellen Daten vergleichen können, identifizieren Sie normale Muster und Anomalien und erarbeiten Sie Methoden, um Probleme zu lösen.

Sie können beispielsweise AWS KMS API-Aktivitäten und Ereignisse überwachen, die sich auf Ihre KMS-Schlüssel auswirken. Wenn Daten Grenzwerte über- oder unterschreiten, müssen Sie möglicherweise untersuchen oder Korrekturmaßnahmen ergreifen.

Um eine Grundlinie für normale Muster zu erstellen, überwachen Sie folgende Punkte:

- AWS KMS API-Aktivität für Datenebenenoperationen. Dabei handelt es sich um kryptografische Operationen, die einen KMS-Schlüssel verwenden, z. B. Decrypt, Encrypt und ReEncrypt. GenerateDataKey
- AWS KMS API-Aktivität für Operationen auf der Kontrollebene, die für Sie wichtig sind. Bei diesen Vorgängen wird ein KMS-Schlüssel verwaltet, und Sie sollten möglicherweise diejenigen Vorgänge überwachen, die die Verfügbarkeit eines KMS-Schlüssels (wie ScheduleKeyDeletionCancelKeyDeletion, DisableKey, EnableKeyImportKeyMaterial, und

DeleteImportedKeyMaterial) oder die Zugriffskontrolle eines KMS-Schlüssels ändern (z. B. PutKeyPolicyund RevokeGrant).

- Andere AWS KMS Messwerte (wie die verbleibende Zeit, bis Ihr importiertes Schlüsselmaterial abläuft) und Ereignisse (wie das Ablaufen von importiertem Schlüsselmaterial oder das Löschen oder die Schlüsselrotation eines KMS-Schlüssels).

Überwachungstools

AWS bietet verschiedene Tools, mit denen Sie Ihre KMS-Schlüssel überwachen können. Sie können einige dieser Tools so konfigurieren, dass diese die Überwachung für Sie übernehmen, während bei anderen Tools ein manuelles Eingreifen nötig ist. Wir empfehlen, dass Sie die Überwachungsaufgaben möglichst automatisieren.

Automatisierte Überwachungstools

Sie können die folgenden automatisierten Überwachungstools verwenden, um Ihre KMS-Schlüssel zu verfolgen und zu berichten, wenn sich etwas geändert hat.

- AWS CloudTrail Protokollüberwachung — Teilen Sie Protokolldateien zwischen Konten, überwachen CloudTrail Sie Protokolldateien in Echtzeit, indem Sie sie an CloudWatch Logs senden, schreiben Sie Protokollverarbeitungsanwendungen mit der CloudTrail Processing Library und überprüfen Sie, ob sich Ihre Protokolldateien nach der Lieferung von nicht geändert haben CloudTrail. Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Arbeiten mit CloudTrail Protokolldateien.
- Amazon CloudWatch Alarms — Überwachen Sie eine einzelne Metrik über einen von Ihnen angegebenen Zeitraum und führen Sie eine oder mehrere Aktionen aus, die auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über mehrere Zeiträume basieren. Die Aktion ist eine Benachrichtigung, die an ein Amazon Simple Notification Service (Amazon SNS) -Thema oder eine Amazon EC2 Auto Scaling Scaling-Richtlinie gesendet wird. CloudWatch Alarme lösen keine Aktionen aus, nur weil sie sich in einem bestimmten Status befinden. Der Status muss sich geändert haben und für eine bestimmte Anzahl von Zeiträumen beibehalten worden sein. Weitere Informationen finden Sie unter Überwachen Sie KMS-Schlüssel mit Amazon CloudWatch.
- Amazon EventBridge — Ordnen Sie Ereignisse zu und leiten Sie sie an eine oder mehrere Zielfunktionen oder -streams weiter, um Statusinformationen zu erfassen und, falls erforderlich, Änderungen vorzunehmen oder Korrekturmaßnahmen zu ergreifen. Weitere Informationen finden

Sie unter Überwachen Sie KMS-Schlüssel mit Amazon EventBridge und im EventBridge Amazon-Benutzerhandbuch.

- Amazon CloudWatch Logs — Überwachen, speichern und greifen Sie auf Ihre Protokolldateien aus AWS CloudTrail oder anderen Quellen zu. Weitere Informationen finden Sie im Amazon CloudWatch Logs-Benutzerhandbuch.

Manuelle Überwachungstools

Ein weiterer wichtiger Teil der Überwachung von KMS-Schlüsseln ist die manuelle Überwachung der Elemente, die von den CloudWatch Alarmen und Ereignissen nicht abgedeckt sind. Die Dashboards AWS KMS CloudWatch AWS Trusted Advisor,, und andere AWS Dashboards bieten einen at-a-glance Überblick über den Zustand Ihrer AWS Umgebung.

Sie können die Seiten Von AWS verwaltete Schlüssel und Customer managed keys (kundenverwaltete Schlüssel) der AWS KMS -Konsole anpassen, um folgende Informationen zu jedem KMS-Schlüssel anzuzeigen:

- Schlüssel-ID
- Status
- Erstellungsdatum
- Ablaufdatum (für KMS-Schlüssel mit importiertem Schlüsselmaterial)
- Urspung
- Benutzerdefinierte Schlüsselspeicher-ID (für KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern)

Das CloudWatch -Konsolen-Dashboard zeigt Folgendes an:

- Aktuelle Alarme und Status
- Diagramme mit Alarmen und Ressourcen
- Servicestatus

Darüber hinaus können CloudWatch Sie Folgendes verwenden:

- Erstellen angepasster Dashboards zur Überwachung der gewünschten Services.
- Aufzeichnen von Metrikdaten, um Probleme zu beheben und Trends zu erkennen

- Suchen und durchsuchen Sie alle Ihre AWS Ressourcenmetriken
- Erstellen und Bearbeiten von Alarmen, um über Probleme benachrichtigt zu werden

AWS Trusted Advisor kann Ihnen helfen, Ihre AWS Ressourcen zu überwachen, um Leistung, Zuverlässigkeit, Sicherheit und Wirtschaftlichkeit zu verbessern. Vier Trusted Advisor Checks stehen allen Benutzern zur Verfügung; mehr als 50 Checks stehen Benutzern mit einem Business- oder Enterprise-Supportplan zur Verfügung. Weitere Informationen finden Sie unter AWS Trusted Advisor.

AWS KMS API-Aufrufe protokollieren mit AWS CloudTrail

AWS KMS ist in einen Dienst integriert AWS CloudTrail, der alle Aufrufe AWS KMS von Benutzern, Rollen und anderen AWS Diensten aufzeichnet. CloudTrail erfasst alle API-Aufrufe AWS KMS als Ereignisse, einschließlich Aufrufe von der AWS KMS Konsole AWS KMS APIs, AWS CloudFormation Vorlagen, AWS Command Line Interface (AWS CLI) und AWS -Tools für PowerShell.

CloudTrail protokolliert alle AWS KMS Operationen, einschließlich schreibgeschützter Operationen wie ListAliasesund GetKeyRotationStatus, Operationen zur Verwaltung von KMS-Schlüsseln wie CreateKeyund und PutKeyPolicykryptografischer Operationen wie GenerateDataKeyund Decrypt. Außerdem werden interne Vorgänge protokolliert, die für AWS KMS Sie erforderlich sind, z. B., DeleteExpiredKeyMaterial, und DeleteKey. SynchronizeMultiRegionKeyRotateKey

CloudTrail protokolliert alle erfolgreichen Operationen und in einigen Szenarien fehlgeschlagene Aufrufversuche, z. B. wenn dem Anrufer der Zugriff auf eine Ressource verweigert wird. Kontoübergreifende Vorgänge auf KMS-Schlüssel in anderen Konten werden sowohl im Konto des Anrufers als auch im Konto des KMS-Schlüsselbesitzers protokolliert. Kontoübergreifende AWS KMS Anfragen, die abgelehnt werden, weil der Zugriff verweigert wurde, werden jedoch nur im Konto des Anrufers protokolliert.

Aus Sicherheitsgründen werden einige Felder in den AWS KMS Protokolleinträgen weggelassen, z. B. der `Plaintext` Parameter einer Verschlüsselungsanforderung und die Antwort auf GetKeyPolicyoder andere kryptografische Operationen. Um die Suche nach CloudTrail Protokolleinträgen für bestimmte KMS-Schlüssel zu vereinfachen, AWS KMS fügt der Schlüssel-ARN des betroffenen KMS-Schlüssels dem `responseElements` Feld in den Protokolleinträgen für einige AWS KMS Schlüsselverwaltungsvorgänge hinzu, auch wenn der API-Vorgang den Schlüssel-ARN nicht zurückgibt.

Standardmäßig werden zwar alle AWS KMS Aktionen als CloudTrail Ereignisse protokolliert, Sie können jedoch AWS KMS Aktionen von einem CloudTrail Trail ausschließen. Details hierzu finden Sie unter AWS KMS Ereignisse aus einer Spur ausschließen.

Weitere Informationen:

- CloudTrail Protokollbeispiele für AWS KMS Operationen in einer AWS Nitro-Enklave finden Sie unter. Überwachung von Anfragen für Nitro-Enklaven

Themen

- Suchen nach AWS KMS Protokolleinträgen in CloudTrail
- AWS KMS Ereignisse aus einer Spur ausschließen
- Beispiele für AWS KMS Protokolleinträge

Suchen nach AWS KMS Protokolleinträgen in CloudTrail

Verwenden Sie die CloudTrail Konsole oder den CloudTrail LookupEventsVorgang, um nach CloudTrail Protokolleinträgen zu suchen. CloudTrail unterstützt zahlreiche Attributwerte zum Filtern Ihrer Suche, einschließlich Ereignisname, Benutzername und Ereignisquelle.

Füllt die folgenden AWS KMS Protokolleintragsfelder aus CloudTrail, AWS KMS um Ihnen die Suche nach CloudTrail Protokolleinträgen zu erleichtern.

> **Note**
>
> Füllt ab Dezember 2022 die Attribute Ressourcentyp und Ressourcenname in allen Verwaltungsvorgängen aus, die einen bestimmten KMS-Schlüssel ändern. AWS KMS Diese Attributwerte können in älteren CloudTrail Einträgen für die folgenden Operationen Null sein: CreateAliasCreateGrantDeleteAlias, DeleteImportedKeyMaterial, ImportKeyMaterial, ReplicateKey, RetireGrant, RevokeGrantUpdateAlias, und UpdatePrimaryRegion.

Attribut	Wert	Protokolleinträge
Ereignisquelle (`EventSource`)	`kms.amazonaws.com`	Alle Vorgänge.

Attribut	Wert	Protokolleinträge
Ressourcentyp (`ResourceType`)	`AWS::KMS::Key`	Verwaltungsvorgänge, die einen bestimmten KMS-Schlüssel ändern, z. B. `CreateKey` und `EnableKey`, aber nicht `ListKeys`.
Ressourcenname (`ResourceName`)	Schlüssel-ARN (oder Schlüssel-ID und Schlüssel-ARN)	Verwaltungsvorgänge, die einen bestimmten KMS-Schlüssel ändern, z. B. `CreateKey` und `EnableKey`, aber nicht `ListKeys`.

Um Ihnen das Auffinden von Protokolleinträgen für Verwaltungsvorgänge mit bestimmten KMS-Schlüsseln zu erleichtern, AWS KMS zeichnet es den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Element des Protokolleintrags auf, auch wenn der AWS KMS API-Vorgang den Schlüssel-ARN nicht zurückgibt.

Beispielsweise gibt ein erfolgreicher Aufruf des DisableKeyVorgangs keine Werte in der Antwort zurück, aber statt eines Nullwerts enthält der `responseElements.keyId` Wert im DisableKey Protokolleintrag den Schlüssel ARN des deaktivierten KMS-Schlüssels.

Diese Funktion wurde im Dezember 2022 hinzugefügt und wirkt sich auf die folgenden CloudTrail Protokolleinträge aus: CreateAliasCreateGrantDeleteAlias, DeleteKey, DisableKey, EnableKey, EnableKeyRotation, ImportKeyMaterial, RotateKey, SynchronizeMultiRegionKey, TagResource, UntagResource, UpdateAlias, und UpdatePrimaryRegion.

AWS KMS Ereignisse aus einer Spur ausschließen

Um die Nutzung und Verwaltung ihrer AWS KMS Ressourcen aufzuzeichnen, verlassen sich die meisten AWS KMS Benutzer auf die Ereignisse in einem CloudTrail Trail. Der Trail kann eine wertvolle Datenquelle für die Überwachung kritischer Ereignisse sein, wie z. B. das Erstellen, Deaktivieren und Löschen AWS KMS keys, Ändern von Schlüsselrichtlinien und die Verwendung Ihrer KMS-Schlüssel durch AWS Dienste in Ihrem Namen. In einigen Fällen können Ihnen die Metadaten in einem CloudTrail Protokolleintrag, z. B. der Verschlüsselungskontext bei einem Verschlüsselungsvorgang, dabei helfen, Fehler zu vermeiden oder zu beheben.

Da jedoch eine große Anzahl von Ereignissen generiert werden AWS KMS kann, AWS CloudTrail können Sie AWS KMS Ereignisse aus einer Spur ausschließen. Diese Einstellung pro Trail schließt alle AWS KMS Ereignisse aus. Sie können bestimmte AWS KMS Ereignisse nicht ausschließen.

> **Warning**
>
> Durch das Ausschließen von AWS KMS Ereignissen aus einem CloudTrail Protokoll können Aktionen, die Ihre KMS-Schlüssel verwenden, verschleiert werden. Seien Sie vorsichtig, wenn Sie Prinzipalen die `cloudtrail:PutEventSelectors`-Berechtigung erteilen, die zum Ausführen dieser Operation erforderlich ist.

So schließen Sie AWS KMS Ereignisse aus einem Trail aus:

- Verwenden Sie in der CloudTrail Konsole die Einstellung Ereignisse des Schlüsselverwaltungsdienstes protokollieren, wenn Sie einen Trail erstellen oder einen Trail aktualisieren. Anweisungen finden Sie AWS Management Console im AWS CloudTrail Benutzerhandbuch unter Protokollieren von Management-Ereignissen mit dem.
- Verwenden Sie in der CloudTrail API den PutEventSelectorsVorgang. Fügen Sie das Attribut `ExcludeManagementEventSources` mit dem Wert `kms.amazonaws.com` zu Ihren Ereignisselektoren hinzu. Ein Beispiel finden Sie im AWS CloudTrail Benutzerhandbuch unter Beispiel: Ein Trail, der keine AWS Key Management Service Ereignisse protokolliert.

Sie können diesen Ausschluss jederzeit deaktivieren, indem Sie die Konsoleneinstellung oder die Ereignisselektoren für einen Trail ändern. Der Trail beginnt dann mit der Aufzeichnung von AWS KMS Ereignissen. AWS KMS Ereignisse, die während der Gültigkeit des Ausschlusses eingetreten sind, können jedoch nicht wiederhergestellt werden.

Wenn Sie AWS KMS Ereignisse mithilfe der Konsole oder der API ausschließen, wird der daraus resultierende CloudTrail `PutEventSelectors` API-Vorgang auch in Ihren CloudTrail Protokollen protokolliert. Wenn AWS KMS Ereignisse nicht in Ihren CloudTrail Protokollen erscheinen, suchen Sie nach einem `PutEventSelectors` Ereignis, bei dem das `ExcludeManagementEventSources` Attribut auf gesetzt ist`kms.amazonaws.com`.

Beispiele für AWS KMS Protokolleinträge

AWS KMS schreibt Einträge in Ihr CloudTrail Protokoll, wenn Sie einen AWS KMS Vorgang aufrufen und wenn ein AWS Dienst einen Vorgang in Ihrem Namen aufruft. AWS KMS schreibt auch einen

Eintrag, wenn er eine Operation für Sie aufruft. Zum Beispiel schreibt es einen Eintrag, wenn es einen KMS-Schlüssel löscht, den Sie zum Löschen vorgesehen haben.

Die folgenden Themen enthalten Beispiele für CloudTrail Protokolleinträge für AWS KMS Operationen.

Beispiele für CloudTrail Protokolleinträge von Anfragen an AWS KMS von AWS Nitro Enclaves finden Sie unter. Überwachung von Anfragen für Nitro-Enklaven

Themen

- CancelKeyDeletion
- ConnectCustomKeyStore
- CreateAlias
- CreateCustomKeyStore
- CreateGrant
- CreateKey
- Decrypt
- DeleteAlias
- DeleteCustomKeyStore
- DeleteExpiredKeyMaterial
- DeleteImportedKeyMaterial
- DeleteKey
- DescribeCustomKeyStores
- DescribeKey
- DisableKey
- DisableKeyRotation
- DisconnectCustomKeyStore
- EnableKey
- EnableKeyRotation
- Encrypt
- GenerateDataKey

- GenerateDataKeyPair
- GenerateDataKeyPairWithoutPlaintext
- GenerateDataKeyWithoutPlaintext
- GenerateMac
- GenerateRandom
- GetKeyPolicy
- GetKeyRotationStatus
- GetParametersForImport
- ImportKeyMaterial
- ListAliases
- ListGrants
- ListKeyRotations
- PutKeyPolicy
- ReEncrypt
- ReplicateKey
- RetireGrant
- RevokeGrant
- RotateKey
- RotateKeyOnDemand
- ScheduleKeyDeletion
- Sign
- SynchronizeMultiRegionKey
- TagResource
- UntagResource
- UpdateAlias
- UpdateCustomKeyStore
- UpdateKeyDescription
- UpdatePrimaryRegion

- VerifyMac
- Verify
- EC2 Amazon-Beispiel eins
- EC2 Amazon-Beispiel zwei

CancelKeyDeletion

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf der CancelKeyDeletionOperation generiert wurde. Hinweise zum Löschen finden AWS KMS keys Sie unterLösche eine AWS KMS key.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-27T21:53:17Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CancelKeyDeletion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "e3452e68-d4b0-4ec7-a768-7ae96c23764f",
    "eventID": "d818bf03-6655-48e9-8b26-f279a07075fd",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

ConnectCustomKeyStore

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf der `ConnectCustomKeyStore`Operation generiert wurde. Hinweise zum Verbinden von benutzerdefinierten Schlüsselspeichern finden Sie unter Verbindung zu einem AWS CloudHSM Schlüsselspeicher trennen.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ConnectCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```

CreateAlias

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den CreateAliasVorgang. Das `resources`-Element enthält Felder für den Alias und KMS-Schlüsselressourcen. Hinweise zum Erstellen von Aliasen in finden Sie AWS KMS unterAliase erstellen.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-08-14T23:08:31Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateAlias",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "aliasName": "alias/ExampleAlias",
        "targetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "caec1e0c-ce03-419e-bdab-6ab1f7c57c01",
    "eventID": "2dd6e784-8286-46a6-befd-d64e5a02fb28",
    "readOnly": false,
    "resources": [
        {
```

```
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

CreateCustomKeyStore

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf des `CreateCustomKeyStore`Vorgangs in einem AWS CloudHSM Schlüsselspeicher generiert wurde. Hinweise zum Erstellen von benutzerdefinierten Schlüsselspeichern finden Sie unter Erstellen Sie einen AWS CloudHSM Schlüsselspeicher.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
```

```
    },
    "responseElements": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```

CreateGrant

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den CreateGrantVorgang. Informationen zum Erstellen von Zuschüssen in AWS KMS finden Sie unterZuschüsse in AWS KMS.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

```
{
  "eventVersion": "1.02",
  "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::111122223333:user/Alice",
      "accountId": "111122223333",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
  },
  "eventTime": "2014-11-04T00:53:12Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateGrant",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
      "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "constraints": {
          "encryptionContextSubset": {
              "ContextKey1": "Value1"
```

```
            }
        },
        "operations": ["Encrypt",
        "RetireGrant"],
        "granteePrincipal": "EX_PRINCIPAL_ID"
    },
    "responseElements": {
        "grantId": "f020fe75197b93991dc8491d6f19dd3cebb24ee62277a05914386724f3d48758",
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "f3c08808-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "5d529779-2d27-42b5-92da-91aaea1fc4b5",
    "readOnly": false,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

CreateKey

Diese Beispiele zeigen AWS CloudTrail Protokolleinträge für den CreateKeyVorgang.

Ein `CreateKey` Protokolleintrag kann aus einer `CreateKey` Anfrage oder dem `CreateKey` Vorgang für eine ReplicateKeyAnfrage resultieren.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für einen CreateKeyVorgang, der einen KMS-Schlüssel mit symmetrischer Verschlüsselung erstellt. Weitere Informationen zum Erstellen und Verwalten von KMS-Schlüsseln finden Sie unter Erste Schritte.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
```

```
    },
    "eventTime": "2022-08-10T22:38:27Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "description": "",
        "origin": "EXTERNAL",
        "bypassPolicyLockoutSafetyCheck": false,
        "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "keySpec": "SYMMETRIC_DEFAULT",
        "keyUsage": "ENCRYPT_DECRYPT"
    },
    "responseElements": {
        "keyMetadata": {
            "AWSAccountId": "111122223333",
            "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "creationDate": "Aug 10, 2022, 10:38:27 PM",
            "enabled": false,
            "description": "",
            "keyUsage": "ENCRYPT_DECRYPT",
            "keyState": "PendingImport",
            "origin": "EXTERNAL",
            "keyManager": "CUSTOMER",
            "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
            "keySpec": "SYMMETRIC_DEFAULT",
            "encryptionAlgorithms": [
                "SYMMETRIC_DEFAULT"
            ],
            "multiRegion": false
        }
    },
    "requestID": "1aef6713-0223-4ff7-9a6d-781360521930",
    "eventID": "36327b37-f4f6-40a9-92ab-48064ec905a2",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Das folgende Beispiel zeigt das CloudTrail Protokoll eines `CreateKey` Vorgangs, bei dem ein KMS-Schlüssel mit symmetrischer Verschlüsselung in einem AWS CloudHSM Schlüsselspeicher erstellt wird.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-10-14T17:39:50Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyUsage": "ENCRYPT_DECRYPT",
        "bypassPolicyLockoutSafetyCheck": false,
        "origin": "AWS_CLOUDHSM",
        "keySpec": "SYMMETRIC_DEFAULT",
        "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "customKeyStoreId": "cks-1234567890abcdef0",
        "description": ""
    },
    "responseElements": {
        "keyMetadata": {
            "aWSAccountId": "111122223333",
            "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
```

```
            "arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
            "creationDate": "Oct 14, 2021, 5:39:50 PM",
            "enabled": true,
            "description": "",
            "keyUsage": "ENCRYPT_DECRYPT",
            "keyState": "Enabled",
            "origin": "AWS_CLOUDHSM",
            "customKeyStoreId": "cks-1234567890abcdef0",
            "cloudHsmClusterId": "cluster-1a23b4cdefg",
            "keyManager": "CUSTOMER",
            "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
            "keySpec": "SYMMETRIC_DEFAULT",
            "encryptionAlgorithms": [
                "SYMMETRIC_DEFAULT"
            ],
            "multiRegion": false
        }
    },
    "additionalEventData": {
        "backingKey": "{\"backingKeyId\":\"backing-key-id\"}"
    },
    "requestID": "4f0b185c-588c-4767-9e90-c618f7e13cad",
    "eventID": "c73964b8-703d-49e4-bd9e-f773d0ee1e65",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Das folgende Beispiel zeigt das CloudTrail Protokoll eines `CreateKey` Vorgangs, bei dem ein KMS-Schlüssel mit symmetrischer Verschlüsselung in einem externen Schlüsselspeicher erstellt wird.

```
{
```

```
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2022-09-07T22:37:45Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "tags": [],
        "keyUsage": "ENCRYPT_DECRYPT",
        "description": "",
        "origin": "EXTERNAL_KEY_STORE",
        "multiRegion": false,
        "keySpec": "SYMMETRIC_DEFAULT",
        "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "bypassPolicyLockoutSafetyCheck": false,
        "customKeyStoreId": "cks-1234567890abcdef0",
        "xksKeyId": "bb8562717f809024"
    },
    "responseElements": {
        "keyMetadata": {
            "aWSAccountId": "111122223333",
            "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "creationDate": "Dec 7, 2022, 10:37:45 PM",
            "enabled": true,
            "description": "",
            "keyUsage": "ENCRYPT_DECRYPT",
            "keyState": "Enabled",
            "origin": "EXTERNAL_KEY_STORE",
            "customKeyStoreId": "cks-1234567890abcdef0",
            "keyManager": "CUSTOMER",
            "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
            "keySpec": "SYMMETRIC_DEFAULT",
            "encryptionAlgorithms": [
```

```
                "SYMMETRIC_DEFAULT"
            ],
            "multiRegion": false,
            "xksKeyConfiguration": {
                "id": "bb8562717f809024"
            }
        }
    },
    "requestID": "ba197c82-3ac7-487a-8ff4-7736bbeb1316",
    "eventID": "838ad5f4-5fdd-4044-afd7-4dbd88c6af56",
    "readOnly": false,
    "resources": [
        {
            "accountId": "227179770375",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:227179770375:key/39c5eb22-
f37c-4956-92ca-89e8f8b57ab2"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Decrypt

Diese Beispiele zeigen AWS CloudTrail Protokolleinträge für den Decrypt-Vorgang.

Der CloudTrail Protokolleintrag für einen `Decrypt` Vorgang enthält immer das`encryptionAlgorithm`, `requestParameters` auch wenn der Verschlüsselungsalgorithmus in der Anforderung nicht angegeben wurde. Der Chiffretext in der Anforderung und der Klartext in der Antwort werden weggelassen.

Themen

- Entschlüsselung mit einem symmetrischen Standardverschlüsselungsschlüssel
- Fehler bei der Entschlüsselung mit einem symmetrischen Standardverschlüsselungsschlüssel
- Mit einem KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher entschlüsseln
- Entschlüsselung mit einem KMS-Schlüssel in einem externen Schlüsselspeicher
- Fehler bei der Entschlüsselung mit einem KMS-Schlüssel in einem externen Schlüsselspeicher

Entschlüsselung mit einem symmetrischen Standardverschlüsselungsschlüssel

Im Folgenden finden Sie ein Beispiel für einen CloudTrail Protokolleintrag für einen `Decrypt` Vorgang mit einem standardmäßigen symmetrischen Verschlüsselungsschlüssel.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:45:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "Department": "Engineering",
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

Fehler bei der Entschlüsselung mit einem symmetrischen Standardverschlüsselungsschlüssel

Der folgende CloudTrail Beispielprotokolleintrag zeichnet einen fehlgeschlagenen `Decrypt` Vorgang mit einem KMS-Schlüssel für die symmetrische Standardverschlüsselung auf. Die Ausnahme (`errorCode`) und die Fehlermeldung (`errorMessage`) sind enthalten und helfen Ihnen, den Fehler zu beheben.

In diesem Fall war der in der `Decrypt`-Anfrage angegebene symmetrische KMS-Verschlüsselungsschlüssel nicht der symmetrische KMS-Verschlüsselungsschlüssel, der zum Verschlüsseln der Daten verwendet wurde.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-11-24T18:57:43Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "errorCode": "IncorrectKeyException"
    "errorMessage": "The key ID in the request does not identify a CMK that can perform
 this operation.",
```

```
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "Department": "Engineering",
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "requestID": "22345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Mit einem KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher entschlüsseln

Der folgende CloudTrail Beispielprotokolleintrag zeichnet einen `Decrypt` Vorgang mit einem KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher auf. Alle Protokolleinträge für kryptografische Operationen mit einem KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher enthalten ein `additionalEventData` Feld mit dem `customKeyStoreId` und`backingKeyId`. Der im `backingKeyId` Feld zurückgegebene Wert ist das `id` CloudHSM-Schlüsselattribut. Die `additionalEventData` ist nicht in der Anfrage angegeben.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
```

```
    },
    "eventTime": "2021-10-26T23:41:27Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "Department": "Development",
            "Purpose": "Test"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "requestID": "e1b881f8-2048-41f8-b6cc-382b7857ec61",
    "eventID": "a79603d5-4cde-46fc-819c-a7cf547b9df4",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Entschlüsselung mit einem KMS-Schlüssel in einem externen Schlüsselspeicher

Der folgende CloudTrail Beispielprotokolleintrag zeichnet einen `Decrypt` Vorgang mit einem KMS-Schlüssel in einem externen Schlüsselspeicher auf. Zusätzlich zur `customKeyStoreId` enthält das `additionalEventData`-Feld die externe Schlüssel-ID (`XksKeyId`). Die `additionalEventData` ist nicht in der Anfrage angegeben.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-11-24T00:26:58Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "encryptionContext": {
            "Department": "Engineering",
            "Purpose": "Test"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreId": "cks-9876543210fedcba9",
        "xksKeyId": "abc01234567890fe"
    },
    "requestID": "f1b881f8-2048-41f8-b6cc-382b7857ec61",
    "eventID": "b79603d5-4cde-46fc-819c-a7cf547b9df4",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
```

```
    "eventCategory": "Management"
}
```

Fehler bei der Entschlüsselung mit einem KMS-Schlüssel in einem externen Schlüsselspeicher

Der folgende CloudTrail Beispielprotokolleintrag zeichnet eine fehlgeschlagene Anforderung für einen `Decrypt` Vorgang mit einem KMS-Schlüssel in einem externen Schlüsselspeicher auf. CloudWatch protokolliert zusätzlich zu erfolgreichen Anfragen auch fehlgeschlagene Anfragen. Bei der Aufzeichnung eines Fehlers enthält der CloudTrail Protokolleintrag die Ausnahme (errorCode) und die zugehörige Fehlermeldung (errorMessage).

Wenn die fehlgeschlagene Anforderung Ihren externen Schlüsselspeicher-Proxy erreicht hat, wie in diesem Beispiel, können Sie den `requestId`-Wert verwenden, um die fehlgeschlagene Anforderung einer entsprechenden Anfrage zuzuordnen, die Ihr externer Schlüsselspeicher-Proxy protokolliert, sofern Ihr Proxy sie bereitstellt.

Hilfe zu `Decrypt`-Anfragen in externen Schlüsselspeichern finden Sie unter Entschlüsselungsfehler.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-11-24T00:26:58Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "errorCode": "KMSInvalidStateException",
    "errorMessage": "The external key store proxy rejected the request because the
 specified ciphertext or additional authenticated data is corrupted, missing, or
 otherwise invalid.",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
```

```
        "encryptionContext": {
            "Department": "Engineering",
            "Purpose": "Test"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreId": "cks-9876543210fedcba9",
        "xksKeyId": "abc01234567890fe"
    },
    "requestID": "f1b881f8-2048-41f8-b6cc-382b7857ec61",
    "eventID": "b79603d5-4cde-46fc-819c-a7cf547b9df4",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

DeleteAlias

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den DeleteAliasVorgang. Weitere Informationen zum Löschen von Aliassen finden Sie unter Löscht einen Alias.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
```

```
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2014-11-04T00:52:27Z"
            }
        }
    },
    "eventTime": "2014-11-04T00:52:27Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteAlias",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "aliasName": "alias/my_alias"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "d9542792-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "12f48554-bb04-4991-9cfc-e7e85f68eda0",
    "readOnly": false,
    "resources": [{
        "ARN": "arn:aws:kms:us-east-1:111122223333:alias/my_alias",
        "accountId": "111122223333"
    },
    {
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

DeleteCustomKeyStore

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf der `DeleteCustomKeyStore`Operation generiert wurde. Hinweise zum Erstellen von

benutzerdefinierten Schlüsselspeichern finden Sie unter Einen AWS CloudHSM Schlüsselspeicher löschen.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```

DeleteExpiredKeyMaterial

Wenn Sie Schlüsselmaterial in einen AWS KMS key (KMS-Schlüssel) importieren, können Sie ein Ablaufdatum und eine Ablaufzeit für dieses Schlüsselmaterial festlegen. AWS KMS zeichnet einen Eintrag in Ihrem CloudTrail Protokoll auf, wenn Sie das Schlüsselmaterial importieren (mit den Ablaufeinstellungen) und wenn das abgelaufene Schlüsselmaterial AWS KMS gelöscht wird. Weitere

Informationen zum Erstellen eines KMS-Schlüssels mit importiertem Schlüsselmaterial finden Sie unter Schlüsselmaterial für AWS KMS Schlüssel importieren.

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der generiert wird, wenn das abgelaufene Schlüsselmaterial AWS KMS gelöscht wird.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-05-22T19:55:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteExpiredKeyMaterial",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "cfa932fd-0d3a-4a76-a8b8-616863a2b547",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "eventCategory": "Management"
}
```

DeleteImportedKeyMaterial

Wenn Sie Schlüsselmaterial in einen KMS-Schlüssel importieren, können Sie das importierte Schlüsselmaterial mithilfe der DeleteImportedKeyMaterialOperation jederzeit löschen. Wenn Sie importiertes Schlüsselmaterial von einem KMS-Schlüssel löschen, ändert sich der Schlüsselstatus des KMS-Schlüssels auf `PendingImport`, und der KMS-Schlüssel kann nicht in kryptografischen Vorgängen verwendet werden. Details hierzu finden Sie unter Importiertes Schlüsselmaterial löschen.

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der für den `DeleteImportedKeyMaterial` Vorgang generiert wurde.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:45:08Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteImportedKeyMaterial",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "dcf0e82f-dad0-4622-a378-a5b964ad42c1",
    "eventID": "2afbb991-c668-4641-8a00-67d62e1fecbd",
    "readOnly": false,
    "resources": [
        {
```

```
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

DeleteKey

Diese Beispiele zeigen den AWS CloudTrail Protokolleintrag, der generiert wird, wenn ein KMS-Schlüssel gelöscht wird. Um einen KMS-Schlüssel zu löschen, verwenden Sie den ScheduleKeyDeletionVorgang. AWS KMS Löscht nach Ablauf der angegebenen Wartezeit den KMS-Schlüssel und zeichnet einen Eintrag wie den folgenden in Ihrem CloudTrail Protokoll auf, um dieses Ereignis aufzuzeichnen.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

Ein Beispiel für den CloudTrail Protokolleintrag für den `ScheduleKeyDeletion` Vorgang finden Sie unterScheduleKeyDeletion. Weitere Informationen zum Erstellen und Verwalten von KMS-Schlüsseln finden Sie unter Erste Schritte.

Der folgende CloudTrail Beispielprotokolleintrag zeichnet einen `DeleteKey` Vorgang eines KMS-Schlüssels auf, in dem Schlüsselmaterial enthalten ist AWS KMS.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
```

```
    "eventTime": "2020-07-31T00:07:00Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "b25f9cda-74e1-4458-847b-4972a0bf9668",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
     "managementEvent": true,
    "eventCategory": "Management"
}
```

Der folgende CloudTrail Protokolleintrag zeichnet einen `DeleteKey` Vorgang eines KMS-Schlüssels in einem AWS CloudHSM benutzerdefinierten Schlüsselspeicher auf.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-10-26T23:41:27Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
    },
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0",
        "clusterId": "cluster-1a23b4cdefg",
        "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]",
        "backingKeysDeletionStatus": "[{\"backingKeyId\":\"backing-key-id\",
\"deletionStatus\":\"SUCCESS\"}]"
    },
    "eventID": "1234585c-4b0c-4340-ab11-662414b79239",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "managementEvent": true,
    "eventCategory": "Management"
}
```

DescribeCustomKeyStores

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf der `DescribeCustomKeyStores`Operation generiert wurde. Hinweise zum Anzeigen von benutzerdefinierten Schlüsselspeichern finden Sie unter Sehen Sie sich einen AWS CloudHSM Schlüsselspeicher an.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
```

```
    "eventName": "DescribeCustomKeyStores",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0"
    },
    "responseElements": null,
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "2ea1735f-628d-43e3-b2ee-486d02913a78",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```

DescribeKey

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den DescribeKeyVorgang. AWS KMS zeichnet einen Eintrag wie den folgenden auf, wenn Sie den `DescribeKey` Vorgang aufrufen oder KMS-Schlüssel in der AWS KMS Konsole anzeigen. Dieser Aufruf ist das Ergebnis der Anzeige eines Schlüssels in der AWS KMS Verwaltungskonsole.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-26T18:01:36Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
```

```
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

DisableKey

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den DisableKeyVorgang. Hinweise zum Aktivieren und Deaktivieren von AWS KMS keys in finden Sie AWS KMS unterTasten aktivieren und deaktivieren.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:43Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DisableKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": false,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

DisableKeyRotation

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf des DisableKeyRotationVorgangs generiert wurde. Weitere Informationen zur automatischen Schlüsselrotation finden Sie unter Rotieren AWS KMS keys.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T19:31:39Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DisableKeyRotation",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
```

```
    "responseElements": null,
    "requestID": "d6a9351a-ed6e-4581-88d1-2a9a8a538497",
    "eventID": "6313164c-83aa-4cc3-9e1a-b7c426f7a5b1",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

DisconnectCustomKeyStore

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf der `DisconnectCustomKeyStore`Operation generiert wurde. Hinweise zum Trennen von benutzerdefinierten Schlüsselspeichern finden Sie unter Verbindung zu einem AWS CloudHSM Schlüsselspeicher trennen.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DisconnectCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0"
```

```
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```

EnableKey

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den EnableKeyVorgang. Informationen zum Aktivieren und Deaktivieren von AWS KMS keys in finden Sie AWS KMS unter.. Tasten aktivieren und deaktivieren

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:20Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "EnableKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
```

```
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "d528a6fb-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "be393928-3629-4370-9634-567f9274d52e",
    "readOnly": false,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

EnableKeyRotation

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag eines Aufrufs der EnableKeyRotationOperation. Ein Beispiel für den CloudTrail Protokolleintrag, der geschrieben wird, wenn der Schlüssel gedreht wird, finden Sie unterRotateKey. Weitere Informationen zur Drehung von AWS KMS keys finden Sie unter Rotieren AWS KMS keys.

> **Note**
>
> Das rotation-period ist ein optionaler Anforderungsparameter. Wenn Sie bei der Aktivierung der automatischen Schlüsselrotation keinen Rotationszeitraum angeben, ist der Standardwert 365 Tage.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
```

```
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-25T23:41:56Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "EnableKeyRotation",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "rotationPeriodInDays": 180
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "81f5b794-452b-4d6a-932b-68c188165273",
    "eventID": "fefc43a7-8e06-419f-bcab-b3bf18d6a401",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Encrypt

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den Vorgang Encrypt.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
```

```
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Encrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionContext": {
            "Department": "Engineering"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "f3423043-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "91235988-eb87-476a-ac2c-0cdc244e6dca",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

GenerateDataKey

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den GenerateDataKeyVorgang.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "keySpec": "AES_256",
        "encryptionContext": {
            "Department": "Engineering",
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

GenerateDataKeyPair

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den GenerateDataKeyPairVorgang. In diesem Beispiel wird eine Operation aufgezeichnet, die ein RSA-Schlüsselpaar generiert, das mit einer symmetrischen Verschlüsselung AWS KMS key verschlüsselt ist.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyPair",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyPairSpec": "RSA_3072",
        "encryptionContext": {
            "Project": "Alpha"
        },
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
```

```
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
    "eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

GenerateDataKeyPairWithoutPlaintext

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den GenerateDataKeyPairWithoutPlaintextVorgang. In diesem Beispiel wird eine Operation aufgezeichnet, die ein RSA-Schlüsselpaar generiert, das mit einer symmetrischen Verschlüsselung AWS KMS key verschlüsselt ist.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
```

```
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyPairWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyPairSpec": "RSA_4096",
        "encryptionContext": {
            "Index": "5"
        },
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
    "eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```


GenerateDataKeyWithoutPlaintext

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den GenerateDataKeyWithoutPlaintextVorgang.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "keySpec": "AES_256",
        "encryptionContext": {
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "d6b8e411-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "f7734272-9ec5-4c80-9f36-528ebbe35e4a",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
```

```
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

GenerateMac

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den GenerateMacVorgang.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
     },
    "eventTime": "2022-12-23T19:26:54Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateMac",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "macAlgorithm": "HMAC_SHA_512",
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateRandom

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den GenerateRandomVorgang. Da dieser Vorgang kein verwendet AWS KMS key, ist das `resources` Feld leer.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateRandom",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "df1e3de6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "239cb9f7-ae05-4c94-9221-6ea30eef0442",
    "readOnly": true,
    "resources": [],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GetKeyPolicy

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den GetKeyPolicyVorgang. Informationen zum Anzeigen der Schlüsselrichtlinie für einen KMS-Schlüssel finden Sie unter Wichtige Richtlinien anzeigen.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
```

```
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:50:30Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GetKeyPolicy",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "policyName": "default"
    },
    "responseElements": null,
    "requestID": "93746dd6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "4aa7e4d5-d047-452a-a5a6-2cce282a7e82",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GetKeyRotationStatus

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den GetKeyRotationStatusVorgang. Informationen zur automatischen und bedarfsgesteuerten Rotation von Schlüsselmaterial für einen KMS-Schlüssel finden Sie unterRotieren AWS KMS keys.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
```

```
        "userName": "Alice"
    },
    "eventTime": "2024-02-20T19:16:45Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GetKeyRotationStatus",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "12f9b7e8-49b9-4c1c-a7e3-34ac0cdf0467",
    "eventID": "3d082126-9e7d-4167-8372-a6cfcbed4be6",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```


GetParametersForImport

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der generiert wurde, wenn Sie den GetParametersForImportVorgang verwenden. Dieser Vorgang gibt den öffentlichen Schlüssel und Import-Token zurück, den Sie beim Importieren von Schlüsselmaterial in einen KMS-Schlüssel verwenden. Derselbe CloudTrail Eintrag wird aufgezeichnet, wenn Sie den `GetParametersForImport` Vorgang verwenden oder die AWS KMS Konsole verwenden, um den öffentlichen Schlüssel herunterzuladen und das Token zu importieren.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2020-07-25T23:58:23Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GetParametersForImport",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "wrappingAlgorithm": "RSAES_OAEP_SHA_256",
        "wrappingKeySpec": "RSA_2048"
    },
    "responseElements": null,
    "requestID": "b5786406-e3c7-43d6-8d3c-6d5ef96e2278",
    "eventID": "4023e622-0c3e-4324-bdef-7f58193bba87",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


ImportKeyMaterial

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der generiert wurde, wenn Sie den ImportKeyMaterialVorgang verwenden. Derselbe CloudTrail Eintrag wird aufgezeichnet, wenn

Sie den `ImportKeyMaterial` Vorgang verwenden oder die AWS KMS Konsole verwenden, um Schlüsselmaterial in eine zu importieren AWS KMS key.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.


```
{
    "eventVersion": "1.11",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2025-05-21T05:42:31Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ImportKeyMaterial",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "validTo": "May 21, 2025, 5:47:45 AM",
        "expirationModel": "KEY_MATERIAL_EXPIRES",
        "importType": "NEW_KEY_MATERIAL",
        "keyMaterialDescription": "ExampleKeyMaterialA"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "89e10ee7-a612-414d-95a2-a128346969fd",
    "eventID": "c7abd205-a5a2-4430-bbfa-fc10f3e2d79f",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

ListAliases

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den ListAliasesVorgang. Da dieser Vorgang keinen bestimmten Alias oder verwendet AWS KMS key, ist das `resources` Feld leer. Hinweise zum Anzeigen von Aliasnamen in finden Sie AWS KMS unterSuchen Sie den Aliasnamen und den Alias-ARN für einen KMS-Schlüssel.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:51:45Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ListAliases",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "limit": 5,
        "marker":
 "eyJiIjoiYWxpYXMvZTU0Y2MxOTMtYTMwNC00YzEwLTliZWItYTJjZjA3NjA2OTJhIiwiYSI6ImFsaWFzL2U1NGNjMTkzL
    },
```

```
    "responseElements": null,
    "requestID": "bfe6c190-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a27dda7b-76f1-4ac3-8b40-42dfba77bcd6",
    "readOnly": true,
    "resources": [],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


ListGrants

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den ListGrantVorgang. Informationen zu Zuschüssen AWS KMS finden Sie unterZuschüsse in AWS KMS.


```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:49Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ListGrants",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "marker":
 "eyJncmFudElkIjoiMWY4M2U2ZmM0YTY2NDgxYjQ2Yzc4MTdhM2Y4YmQwMDFkZDNiYmQ1MGVlYTMyY2RmOWFiNWY1Nzc1N
\u003d\u003d",
        "limit": 10
    },
    "responseElements": null,
    "requestID": "e5c23960-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "d24380f5-1b20-4253-8e92-dd0492b3bd3d",
    "readOnly": true,
    "resources": [{
```

```
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

ListKeyRotations

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den ListKeyRotationsVorgang. Informationen zur automatischen und bedarfsgesteuerten Rotation von Schlüsselmaterial für einen KMS-Schlüssel finden Sie unterRotieren AWS KMS keys.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-21T05:42:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ListKeyRotations",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "includeKeyMaterial": "ALL_KEY_MATERIAL"
    },
    "responseElements": null,
    "requestID": "99c88d32-f2db-455e-8a9a-23855258a452",
    "eventID": "8ce0e74b-b9c7-45a2-96ef-83136d38068e",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

PutKeyPolicy

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf des PutKeyPolicyVorgangs generiert wurde. Informationen zur Aktualisierung einer Schlüsselrichtlinie finden Sie unter Ändern Sie eine wichtige Richtlinie.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T20:06:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "PutKeyPolicy",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "policyName": "default",
        "policy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Id\" : \"key-default-1\",\n
 \"Statement\" : [ {\n    \"Sid\" : \"Enable IAM User Permissions\",\n    \"Effect\" :
```

```
 \"Allow\",\n    \"Principal\" : {\n      \"AWS\" : \"arn:aws:iam::111122223333:root
\"\n    },\n    \"Action\" : \"kms:*\",\n    \"Resource\" : \"*\"\n  } ]\n}",
        "bypassPolicyLockoutSafetyCheck": false
    },
    "responseElements": null,
    "requestID": "7bb906fa-dc21-4350-b65c-808ff0f72f55",
    "eventID": "c217db1f-903f-4a2f-8f88-9580182d6313",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

ReEncrypt

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den ReEncryptVorgang. Das `resources` Feld in diesem Protokolleintrag gibt zwei an AWS KMS keys, den KMS-Quellschlüssel und den KMS-Zielschlüssel, in dieser Reihenfolge.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-22T19:34:55Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ReEncrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
```

```
    "userAgent": "AWS Internal",
    "requestParameters": {
        "sourceEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "sourceEncryptionContext": {
            "Project": "Alpha",
            "Department": "Engineering"
        },
        "destinationKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "destinationEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "destinationEncryptionContext": {
            "Level": "3A"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "destinationKeyMaterialId":
 "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
        "sourceKeyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "03769fd4-acf9-4b33-adf3-2ab8ca73aadf",
    "eventID": "542d9e04-0e8d-4e05-bf4b-4bdeb032e6ec",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
```

```
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

ReplicateKey

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf des ReplicateKeyVorgangs generiert wurde. Eine `ReplicateKey` Anfrage führt zu einer `ReplicateKey` Operation und einer CreateKeyOperation.

Informationen zum Replizieren von multiregionalen Schlüsseln finden Sie unter Erstellen Sie Replikatschlüssel für mehrere Regionen.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-11-18T01:29:18Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ReplicateKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "replicaRegion": "us-west-2",
        "bypassPolicyLockoutSafetyCheck": false,
        "description": ""
    },
    "responseElements": {
        "replicaKeyMetadata": {
            "aWSAccountId": "111122223333",
            "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "creationDate": "Nov 18, 2020, 1:29:18 AM",
            "enabled": false,
```

```
            "description": "",
            "keyUsage": "ENCRYPT_DECRYPT",
            "keyState": "Creating",
            "origin": "AWS_KMS",
            "keyManager": "CUSTOMER",
            "keySpec": "SYMMETRIC_DEFAULT",
            "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
            "encryptionAlgorithms": [
                "SYMMETRIC_DEFAULT"
            ],
            "multiRegion": true,
            "multiRegionConfiguration": {
                "multiRegionKeyType": "REPLICA",
                "primaryKey": {
                    "arn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                    "region": "us-east-1"
                },
                "replicaKeys": [
                    {
                        "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                        "region": "us-west-2"
                    }
                ]
            }
        },
        "replicaPolicy": "{\n  \"Version\":\"2012-10-17\",\n  \"Statement\":[{\n
 \"Effect\":\"Allow\",\n    \"Principal\":{\"AWS\":\"arn:aws:iam::123456789012:user/
Alice\"},\n    \"Action\":\"kms:*\",\n    \"Resource\":\"*\"\n  }, {\n    \"Effect
\":\"Allow\",\n    \"Principal\":{\"AWS\":\"arn:aws:iam::012345678901:user/Bob\"},\n
    \"Action\":\"kms:CreateGrant\",\n    \"Resource\":\"*\"\n  }, {\n    \"Effect\":
\"Allow\",\n    \"Principal\":{\"AWS\":\"arn:aws:iam::012345678901:user/Charlie\"},\n
  \"Action\":\"kms:Encrypt\",\n    \"Resource\":\"*\"\n}]\n}",
    },
    "requestID": "abcdef68-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "fedcba44-6773-4f96-8763-1993aec9ae6a",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

```
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

RetireGrant

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf des RetireGrantVorgangs generiert wurde. Informationen zur Außerbetriebnahme von Erteilungen finden Sie unter Außerbetriebnahme und Widerruf von Erteilungen.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T19:39:33Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RetireGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
        "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
    },
    "requestID": "1d274d57-5697-462c-a004-f25fcc29fa26",
    "eventID": "0771bcfb-3e24-4332-9ac8-e1c06563eecf",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
```

```
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

RevokeGrant

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf der Operation generiert wurde. RevokeGrant Informationen zum Widerrufen von Erteilungen finden Sie unter Außerbetriebnahme und Widerruf von Erteilungen.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T19:35:17Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RevokeGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
    },
    "responseElements": null,
    "requestID": "59d94c03-c5b7-428d-ae6e-f2c4b47d2917",
    "eventID": "07a23a39-6526-4ae2-b31e-d35fbe9e24ee",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
```

```
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

RotateKey

Diese Beispiele zeigen die AWS CloudTrail Protokolleinträge für die Operationen, die rotieren. AWS KMS keys Weitere Informationen zur Drehung von KMS-Schlüssel finden Sie unter Rotieren AWS KMS keys.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für den Vorgang, bei dem ein KMS-Schlüssel mit symmetrischer Verschlüsselung rotiert wird, für den die automatische Schlüsselrotation aktiviert ist. Hinweise zur Aktivierung der automatischen Rotation finden Sie unter. Rotieren AWS KMS keys

Ein Beispiel für den CloudTrail Protokolleintrag, der den `EnableKeyRotation` Vorgang aufzeichnet, finden Sie unterEnableKeyRotation.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-05-20T20:44:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RotateKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "a24b3967-ddad-417f-9b22-2332b918db06",
    "readOnly": false,
    "resources": [
```

```
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "rotationType": "AUTOMATIC",
        "keyOrigin": "AWS_KMS",
        "previousKeyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
        "currentKeyMaterialId":
 "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "eventCategory": "Management"
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für eine Rotation auf Anforderung, die durch den RotateKeyOnDemandVorgang initiiert wurde. Hinweise zur bedarfsgerechten Rotation von KMS-Schlüsseln mit symmetrischer Verschlüsselung finden Sie unterFühren Sie die Schlüsselrotation bei Bedarf durch.

Ein Beispiel für den CloudTrail Protokolleintrag, der den `RotateKeyOnDemand` Vorgang aufzeichnet, finden Sie unterRotateKeyOnDemand.

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2025-05-20T20:44:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RotateKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
```

```
    "responseElements": null,
    "eventID": "a24b3967-ddad-417f-9b22-2332b918db06",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "rotationType": "ON_DEMAND",
        "keyOrigin": "EXTERNAL",
        "previousKeyMaterialId":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
        "currentKeyMaterialId":
 "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "eventCategory": "Management"
}
```

RotateKeyOnDemand

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den RotateKeyOnDemandVorgang. Ein Beispiel für den CloudTrail Protokolleintrag, der geschrieben wird, wenn der Schlüssel gedreht wird, finden Sie unterRotateKey. Weitere Informationen zur On-Demand-Rotation von Schlüsselmaterial für einen KMS-Schlüssel finden Sie unterFühren Sie die Schlüsselrotation bei Bedarf durch.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
```

```
    },
    "eventTime": "2024-02-20T17:41:57Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RotateKeyOnDemand",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "9e1dee86-eb84-42fd-8f25-e3fc7dbb32c8",
    "eventID": "00a09fbc-20d6-4a58-9b92-7da85984ab77",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
}
```

ScheduleKeyDeletion

Diese Beispiele zeigen AWS CloudTrail Protokolleinträge für den ScheduleKeyDeletionVorgang.

Ein Beispiel für den CloudTrail Protokolleintrag, der beim Löschen des Schlüssels geschrieben wird, finden Sie unterDeleteKey. Informationen zum Löschen von AWS KMS keys finden Sie unter Lösche eine AWS KMS key.

Im folgenden Beispiel wird eine `ScheduleKeyDeletion`-Anforderung für einen einzelregionalen KMS-Schlüssel aufgezeichnet.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-03-23T18:58:30Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "pendingWindowInDays": 20,
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyState": "PendingDeletion",
        "deletionDate": "Apr 12, 2021 18:58:30 PM"
    },
    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "3c4226b0-1e81-48a8-a333-7fa5f3cbd118",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Im folgenden Beispiel wird eine `ScheduleKeyDeletion`-Anforderung für einen multiregionalen KMS-Schlüssel mit Replikatschlüsseln aufgezeichnet.

Denn ein Schlüssel mit mehreren Regionen AWS KMS wird erst gelöscht, wenn alle zugehörigen Replikatschlüssel gelöscht wurden. In dem `responseElements` Feld `keyState` wird das ist `PendingReplicaDeletion` und das `deletionDate` Feld weggelassen.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-10-28T17:59:05Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "pendingWindowInDays": 30,
        "keyId": "mrk-1234abcd12ab34cd56ef1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "keyState": "PendingReplicaDeletion",
        "pendingWindowInDays": 30
    },
    "requestID": "12341411-d846-42a6-a476-b1cbe3011f89",
    "eventID": "abcda5f-396d-494c-9380-0c47860df5f1",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Im folgenden Beispiel wird eine `ScheduleKeyDeletion` Anforderung für einen KMS-Schlüssel in einem AWS CloudHSM benutzerdefinierten Schlüsselspeicher aufgezeichnet.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-10-26T23:25:25Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "pendingWindowInDays": 30
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321",
        "deletionDate": "Nov 2, 2021, 11:25:25 PM",
        "keyState": "PendingDeletion",
        "pendingWindowInDays": 30
    },
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0",
        "clusterId": "cluster-1a23b4cdefg",
        "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]"
    },
```

```
    "requestID": "abcd9f60-2c9c-4a0b-a456-d5d998f7f321",
    "eventID": "ca01996a-01b0-4edd-bbbb-25d7b6d1a6fa",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Sign

Diese Beispiele zeigen AWS CloudTrail Protokolleinträge für den Vorgang Signieren.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für einen Sign-Vorgang, der einen asymmetrischen RSA-KMS-Schlüssel verwendet, um eine digitale Signatur für eine Datei zu generieren.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-03-07T22:36:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Sign",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "messageType": "RAW",
```

```
        "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "signingAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
    },
    "responseElements": null,
    "requestID": "8d0b35e0-46cf-48b9-be99-bf2ebc9ab9fb",
    "eventID": "107b3cac-b125-4556-9702-12a2b9afcff7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

SynchronizeMultiRegionKey

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der generiert wird, wenn ein Schlüssel mit mehreren Regionen AWS KMS synchronisiert wird. Die Synchronisierung beinhaltet regionsübergreifende Aufrufe, um die gemeinsamen Eigenschaften eines Primärschlüssels mit mehreren Regionen in seine Replikatschlüssel zu kopieren. AWS KMS synchronisiert Schlüssel mit mehreren Regionen in regelmäßigen Abständen, um sicherzustellen, dass alle zugehörigen Schlüssel für mehrere Regionen dasselbe Schlüsselmaterial haben.

Das `resources` Element des CloudTrail Protokolleintrags enthält den Schlüssel-ARN des Primärschlüssels für mehrere Regionen, einschließlich seines AWS-Region. Die zugehörigen multiregionalen Replikatschlüssel und ihre Regionen werden in diesem Protokolleintrag nicht aufgeführt.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
```

```
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-11-18T02:04:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "SynchronizeMultiRegionKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "12345681-de97-42e9-bed0-b02ae1abd8dc",
    "eventID": "abcdec99-2b5c-4670-9521-ddb8f031e146",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

TagResource

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag eines Aufrufs der TagResourceOperation zum Hinzufügen eines Tags mit dem Tag-Schlüssel `Department` und dem Tag-Wert von`IT`.

Ein Beispiel für einen `UntagResource` CloudTrail Protokolleintrag, der geschrieben wird, wenn der Schlüssel gedreht wird, finden Sie unterUntagResource. Hinweise zum Markieren finden Sie AWS KMS keys unterSchlagworte in AWS KMS.

```
{
    "eventVersion": "1.05",
```

```
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-01T21:19:25Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "TagResource",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "tags": [
            {
                "tagKey": "Department",
                "tagValue": "IT"
            }
        ]
    },
    "responseElements": null,
    "requestID": "b942584a-f77d-4787-9feb-b9c5be6e746d",
    "eventID": "0a091b9b-0df5-4cf9-b667-6f2879532b8f",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

UntagResource

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag eines Aufrufs der UntagResourceOperation zum Löschen eines Tags mit dem Tag-Schlüssel von`Dept`.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

Ein Beispiel für einen `TagResource` CloudTrail Protokolleintrag finden Sie unterTagResource. Weitere Informationen über das Markieren von AWS KMS keys finden Sie unter Schlagworte in AWS KMS.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-01T21:19:19Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UntagResource",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "tagKeys": [
            "Dept"
        ]
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "cb1d507b-6015-47f4-812b-179713af8068",
    "eventID": "0b00f4b0-036e-411d-aa75-87eb4a35a4b3",
    "readOnly": false,
```

```
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

UpdateAlias

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den UpdateAliasVorgang. Das `resources`-Element enthält Felder für den Alias und KMS-Schlüsselressourcen. Hinweise zum Erstellen von Aliasen in finden Sie AWS KMS unterAliase erstellen.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
         "principalId": "EX_PRINCIPAL_ID",
         "arn": "arn:aws:iam::111122223333:user/Alice",
         "accountId": "111122223333",
         "accessKeyId": "EXAMPLE_KEY_ID",
         "userName": "Alice"
    },
    "eventTime": "2020-11-13T23:18:15Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UpdateAlias",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "aliasName": "alias/my_alias",
        "targetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
```

```
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "d9472f40-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "f72d3993-864f-48d6-8f16-e26e1ae8dff0",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:alias/my_alias"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

UpdateCustomKeyStore

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf des `UpdateCustomKeyStore`Vorgangs zur Aktualisierung der Cluster-ID für einen benutzerdefinierten Schlüsselspeicher generiert wurde. Hinweise zum Bearbeiten von benutzerdefinierten Schlüsselspeichern finden Sie unter Einstellungen AWS CloudHSM für den Schlüsselspeicher bearbeiten.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
```

```
    "eventName": "UpdateCustomKeyStore",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "customKeyStoreId": "cks-1234567890abcdef0",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "responseElements": null,
    "additionalEventData": {
        "customKeyStoreName": "ExampleKeyStore",
        "clusterId": "cluster-1a23b4cdefg"
    },
    "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
    "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
}
```

UpdateKeyDescription

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag, der durch den Aufruf des UpdateKeyDescriptionVorgangs generiert wurde.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-09-01T19:22:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UpdateKeyDescription",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
```

```
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "description": "New key description"
    },
    "responseElements": null,
    "requestID": "8c3c1f8b-336d-4896-b034-4eb9916bc9b3",
    "eventID": "f5f3d548-2e9e-4658-8427-9dcb5b1ea791",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

UpdatePrimaryRegion

Das folgende Beispiel zeigt die AWS CloudTrail Protokolleinträge, die generiert werden, wenn der UpdatePrimaryRegionVorgang für einen Schlüssel mit mehreren Regionen aufgerufen wird.

Der `UpdatePrimaryRegion` Vorgang schreibt zwei CloudTrail Protokolleinträge: einen in der Region mit dem Primärschlüssel für mehrere Regionen, der in einen Replikatschlüssel konvertiert wird, und einen in der Region mit einem Replikatschlüssel für mehrere Regionen, der in einen Primärschlüssel umgewandelt wird.

CloudTrail Protokolleinträge für diesen Vorgang, die am oder nach Dezember 2022 aufgezeichnet wurden, enthalten den Schlüssel-ARN des betroffenen KMS-Schlüssels im `responseElements.keyId` Wert, obwohl dieser Vorgang den Schlüssel-ARN nicht zurückgibt.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für die Region, `UpdatePrimaryRegion` in der der Schlüssel für mehrere Regionen von einem Primärschlüssel zu einem Replikatschlüssel (us-west-2) geändert wurde. Das `primaryRegion`-Feld zeigt die Region an, die nun den Primärschlüssel hostet (ap-northeast-1).

```
{
    "eventVersion": "1.08",
```

```
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2021-03-10T20:23:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UpdatePrimaryRegion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "primaryRegion": "ap-northeast-1"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "3c4226b0-1e81-48a8-a333-7fa5f3cbd118",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

Das folgende Beispiel stellt den CloudTrail Protokolleintrag für die Region dar, `UpdatePrimaryRegion` in der der Schlüssel für mehrere Regionen von einem Replikatschlüssel in einen Primärschlüssel (ap-northeast-1) geändert wurde. Dieser Protokolleintrag identifiziert die vorherige primäre Region nicht.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice",
            "invokedBy": "kms.amazonaws.com"
    },
    "eventTime": "2021-03-10T20:23:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UpdatePrimaryRegion",
    "awsRegion": "ap-northeast-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "primaryRegion": "ap-northeast-1"
    },
    "responseElements": {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "091e6be5-737f-43c6-8431-e3679d6d0619",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

VerifyMac

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag für den VerifyMacVorgang.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
```

```
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
     },
    "eventTime": "2022-03-31T19:25:54Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "VerifyMac",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "macAlgorithm": "HMAC_SHA_384",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "f35da560-edff-4d6e-9b40-fb306fa9ef1e",
    "eventID": "6b464487-6dea-44cd-84ad-225d7450c975",
    "readOnly": true,
    "resources": [
        {
           "accountId": "111122223333",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Verify

Diese Beispiele zeigen AWS CloudTrail Protokolleinträge für den Vorgang Verify.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag für einen Verify-Vorgang, bei dem ein asymmetrischer RSA-KMS-Schlüssel zur Überprüfung einer digitalen Signatur verwendet wird.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2022-03-07T22:50:41Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Verify",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "signingAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256",
        "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "messageType": "RAW"
    },
    "responseElements": null,
    "requestID": "c73ab82a-af82-4750-ae2c-b6bb790e9c28",
    "eventID": "3b4331cd-5b7b-4de5-bf5f-82ec22f0dac0",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


EC2 Amazon-Beispiel eins

Das folgende Beispiel zeichnet einen IAM-Prinzipal auf, der ein verschlüsseltes Volume mithilfe des Standard-Volumeschlüssels in der EC2 Amazon-Managementkonsole erstellt.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, in dem die Benutzerin Alice in der EC2 Amazon-Managementkonsole ein verschlüsseltes Volume mit einem Standard-Volume-Schlüssel

erstellt. Der EC2 Protokolldateisatz enthält ein `volumeId` Feld mit dem Wert`"vol-13439757"`. Der AWS KMS Datensatz enthält ein `encryptionContext` Feld mit dem Wert`"aws:ebs:id": "vol-13439757"`. Auch die `principalId` und `accountId` zwischen den beiden Datensätzen stimmen überein. Die Datensätze zeigen, dass bei der Erstellung eines verschlüsselten Volumes ein Datenschlüssel für die Verschlüsselung der Volume-Inhalte generiert wird.

```
{
  "Records": [
     {
      "eventVersion": "1.02",
      "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
      "eventTime": "2014-11-05T20:50:18Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "CreateVolume",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "AWS Internal",
      "requestParameters": {
        "size": "10",
        "zone": "us-east-1a",
        "volumeType": "gp2",
        "encrypted": true
      },
      "responseElements": {
        "volumeId": "vol-13439757",
        "size": "10",
        "zone": "us-east-1a",
        "status": "creating",
        "createTime": 1415220618876,
        "volumeType": "gp2",
        "iops": 30,
        "encrypted": true
      },
      "requestID": "1565210e-73d0-4912-854c-b15ed349e526",
      "eventID": "a3447186-135f-4b00-8424-bc41f1a93b4f",
      "eventType": "AwsApiCall",
```

```
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
      },
      "eventTime": "2014-11-05T20:50:19Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "GenerateDataKeyWithoutPlaintext",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "&AWS; Internal",
      "requestParameters": {
        "encryptionContext": {
          "aws:ebs:id": "vol-13439757"
        },
        "numberOfBytes": 64,
        "keyId": "alias/aws/ebs"
      },
      "responseElements": null,
      "requestID": "create-123456789012-758241111-1415220618",
      "eventID": "4bd2a696-d833-48cc-b72c-05e61b608399",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "accountId": "111122223333"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```

EC2 Amazon-Beispiel zwei

Im folgenden Beispiel erstellt und mountet ein IAM-Principal, der eine EC2 Amazon-Instance ausführt, ein Datenvolume, das unter einem KMS-Schlüssel verschlüsselt ist. Diese Aktion generiert mehrere CloudTrail Protokolldatensätze. Weitere Informationen zu Amazon EBS und Verschlüsselung finden Sie unter Anforderungen für die Amazon EBS-Verschlüsselung.

Wenn das Volume erstellt wird EC2, erhält Amazon im Namen des Kunden einen verschlüsselten Datenschlüssel von AWS KMS (`GenerateDataKeyWithoutPlaintext`). Dann wird eine Berechtigungserteilung (`CreateGrant`) erstellt, über die der Datenschlüssel entschlüsselt werden kann. Wenn das Volume bereitgestellt ist, EC2 ruft Amazon auf, AWS KMS um den Datenschlüssel zu entschlüsseln (`Decrypt`).

Die `instanceId` der EC2 Amazon-Instance,`"i-81e2f56c"`, wird im `RunInstances` Ereignis angezeigt. Dieselbe Instance-ID qualifiziert den `granteePrincipal` der erstellten Berechtigungserteilung (`"111122223333:aws:ec2-infrastructure:i-81e2f56c"`) und die angenommene Rolle, die der Prinzipal im `Decrypt`-Aufruf ist (`"arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/i-81e2f56c"`).

Der Schlüssel-ARN des KMS-Schlüssels, der das Datenvolumen schützt`arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`, erscheint in allen drei AWS KMS Aufrufen (`CreateGrant``GenerateDataKeyWithoutPlaintext`, und`Decrypt`).

```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
      },
      "eventTime": "2014-11-05T21:35:27Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "RunInstances",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
```

```
        "userAgent": "AWS Internal",
        "requestParameters": {
          "instancesSet": {
            "items": [
              {
                "imageId": "ami-b66ed3de",
                "minCount": 1,
                "maxCount": 1
              }
            ]
          },
          "groupSet": {
            "items": [
              {
                "groupId": "sg-98b6e0f2"
              }
            ]
          },
          "instanceType": "m3.medium",
          "blockDeviceMapping": {
            "items": [
              {
                "deviceName": "/dev/xvda",
                "ebs": {
                  "volumeSize": 8,
                  "deleteOnTermination": true,
                  "volumeType": "gp2"
                }
              },
              {
                "deviceName": "/dev/sdb",
                "ebs": {
                  "volumeSize": 8,
                  "deleteOnTermination": false,
                  "volumeType": "gp2",
                  "encrypted": true
                }
              }
            ]
          },
          "monitoring": {
            "enabled": false
          },
          "disableApiTermination": false,
```

```
        "instanceInitiatedShutdownBehavior": "stop",
        "clientToken": "XdKUT141516171819",
        "ebsOptimized": false
      },
      "responseElements": {
        "reservationId": "r-5ebc9f74",
        "ownerId": "111122223333",
        "groupSet": {
          "items": [
            {
              "groupId": "sg-98b6e0f2",
              "groupName": "launch-wizard-2"
            }
          ]
        },
        "instancesSet": {
          "items": [
            {
              "instanceId": "i-81e2f56c",
              "imageId": "ami-b66ed3de",
              "instanceState": {
                "code": 0,
                "name": "pending"
              },
              "amiLaunchIndex": 0,
              "productCodes": {

              },
              "instanceType": "m3.medium",
              "launchTime": 1415223328000,
              "placement": {
                "availabilityZone": "us-east-1a",
                "tenancy": "default"
              },
              "monitoring": {
                "state": "disabled"
              },
              "stateReason": {
                "code": "pending",
                "message": "pending"
              },
              "architecture": "x86_64",
              "rootDeviceType": "ebs",
              "rootDeviceName": "/dev/xvda",
```

```
              "blockDeviceMapping": {

              },
              "virtualizationType": "hvm",
              "hypervisor": "xen",
              "clientToken": "XdKUT1415223327917",
              "groupSet": {
                "items": [
                  {
                    "groupId": "sg-98b6e0f2",
                    "groupName": "launch-wizard-2"
                  }
                ]
              },
              "networkInterfaceSet": {

              },
              "ebsOptimized": false
            }
          ]
        }
      },
      "requestID": "41c4b4f7-8bce-4773-bf0e-5ae3bb5cbce2",
      "eventID": "cd75a605-2fee-4fda-b847-9c3d330ebaae",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
      },
      "eventTime": "2014-11-05T21:35:35Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "CreateGrant",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "AWS Internal",
      "requestParameters": {
```

```
        "constraints": {
          "encryptionContextSubset": {
            "aws:ebs:id": "vol-f67bafb2"
          }
        },
        "granteePrincipal": "111122223333:aws:ec2-infrastructure:i-81e2f56c",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      },
      "responseElements": {
        "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
      },
      "requestID": "41c4b4f7-8bce-4773-bf0e-5ae3bb5cbce2",
      "eventID": "c1ad79e3-0d3f-402a-b119-d5c31d7c6a6c",
      "readOnly": false,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "accountId": "111122223333"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
          "type": "IAMUser",
          "principalId": "EX_PRINCIPAL_ID",
          "arn": "arn:aws:iam::111122223333:user/Alice",
          "accountId": "111122223333",
          "accessKeyId": "EXAMPLE_KEY_ID",
          "userName": "Alice"
      },
      "eventTime": "2014-11-05T21:35:32Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "GenerateDataKeyWithoutPlaintext",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "AWS Internal",
      "requestParameters": {
        "encryptionContext": {
          "aws:ebs:id": "vol-f67bafb2"
```

```
        },
        "numberOfBytes": 64,
        "keyId": "alias/aws/ebs"
      },
      "responseElements": null,
      "requestID": "create-111122223333-758247346-1415223332",
      "eventID": "ac3cab10-ce93-4953-9d62-0b6e5cba651d",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "accountId": "111122223333"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "111122223333:aws:ec2-infrastructure:i-81e2f56c",
        "arn": "arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/
i-81e2f56c",
        "accountId": "111122223333",
        "accessKeyId": "",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2014-11-05T21:35:38Z"
          },
          "sessionIssuer": {
            "type": "Role",
            "principalId": "111122223333:aws:ec2-infrastructure",
            "arn": "arn:aws:iam::111122223333:role/aws:ec2-infrastructure",
            "accountId": "111122223333",
            "userName": "aws:ec2-infrastructure"
          }
        }
      },
      "eventTime": "2014-11-05T21:35:47Z",
      "eventSource": "kms.amazonaws.com",
      "eventName": "Decrypt",
```

```
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "requestParameters": {
        "encryptionContext": {
          "aws:ebs:id": "vol-f67bafb2"
        }
      },
      "responseElements": null,
      "requestID": "b4b27883-6533-11e4-b4d9-751f1761e9e5",
      "eventID": "edb65380-0a3e-4123-bbc8-3d1b7cff49b0",
      "readOnly": true,
      "resources": [
        {
          "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "accountId": "111122223333"
        }
      ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333"
    }
  ]
}
```


Überwachen Sie KMS-Schlüssel mit Amazon CloudWatch

Sie können Ihre AWS KMS keys Nutzung von Amazon überwachen CloudWatch, einem AWS Service, der Rohdaten sammelt und AWS KMS in lesbare, nahezu in Echtzeit verfügbare Metriken verarbeitet. Diese Daten werden für einen Zeitraum von zwei Wochen aufgezeichnet, damit Sie auf historische Daten zugreifen und die Nutzung Ihrer KMS-Schlüssel sowie deren Änderungen besser nachvollziehen können.

Sie können Amazon verwenden CloudWatch , um Sie auf wichtige Ereignisse wie die folgenden aufmerksam zu machen.

- Das importierte Schlüsselmaterial in einem KMS-Schlüssel befindet kurz vor dem Ablaufdatum.
- Ein KMS-Schlüssel, dessen Löschung ansteht, wird immer noch verwendet.
- Das Schlüsselmaterial in einem KMS-Schlüssel wurde automatisch rotiert.
- Ein KMS-Schlüssel wurde gelöscht.

Sie können auch einen CloudWatchAmazon-Alarm einrichten, der Sie benachrichtigt, wenn Ihre Anforderungsrate einen bestimmten Prozentsatz eines Kontingents erreicht. Einzelheiten finden Sie im AWS Sicherheitsblog unter Verwalten Sie Ihre AWS KMS API-Anforderungsraten mithilfe von Service Quotas und Amazon CloudWatch.

AWS KMS Metriken und Dimensionen

AWS KMS definiert CloudWatch Amazon-Metriken, um Ihnen die Überwachung kritischer Daten und die Erstellung von Alarmen zu erleichtern. Sie können die AWS KMS Metriken mithilfe der AWS Management Console und der CloudWatch Amazon-API anzeigen.

In diesem Abschnitt werden die einzelnen AWS KMS Metriken und die Dimensionen für jede Metrik aufgeführt und einige grundlegende Anleitungen zur Erstellung von CloudWatch Alarmen auf der Grundlage dieser Metriken und Dimensionen bereitgestellt.

> **Note**
>
> Name der Dimensionsgruppe:
> Um eine Metrik in der CloudWatch Amazon-Konsole anzuzeigen, wählen Sie im Abschnitt Metriken den Namen der Dimensionsgruppe aus. Dann können Sie nach dem Metriknamen filtern. Dieses Thema enthält den Metriknamen und den Dimensionsgruppennamen für jede AWS KMS -Metrik.

Sie können AWS KMS Metriken mithilfe der AWS Management Console und der CloudWatch Amazon-API anzeigen. Weitere Informationen finden Sie unter Verfügbare Messwerte anzeigen im CloudWatch Amazon-Benutzerhandbuch.

Themen

- SuccessfulRequest
- SecondsUntilKeyMaterialExpiration
- Wolke HSMKey StoreThrottle
- ExternalKeyStoreThrottle
- XksProxyCertificateDaysToExpire
- XksProxyCredentialAge
- XksProxyErrors
- XksExternalKeyManagerStates

- XksProxyLatency

SuccessfulRequest

Die Anzahl erfolgreicher Anfragen für kryptografische Operationen mit einem bestimmten KMS-Schlüssel. Mithilfe der `SuccessfulRequest` Metrik können Sie Filter auf Schlüsselebene auf die AWS KMS API-Nutzung in anwenden. CloudWatch Die `Sum` Statistik für diese Metrik definiert die Gesamtzahl der erfolgreichen Anfragen während des Zeitraums.

Verwenden Sie diese Metrik, um zu ermitteln, welche KMS-Schlüssel den größten Teil Ihres Anforderungskontingents verbrauchen oder am meisten zu den API-Gebühren beitragen. Sie können auch einen CloudWatch Alarm auf der Grundlage der `SuccesfulRequest` Metrik erstellen, um Sie über ungewöhnliche AWS KMS API-Nutzungsmuster zu informieren. Diese Benachrichtigungen können dabei helfen, ineffiziente Workflows zu identifizieren, die Ihre Anforderungsquoten unbeabsichtigt überschreiten oder unerwartete Gebühren verursachen könnten.

Abmessungen für **`SuccessfulRequest`**

Dimension	Beschreibung
KeyArn	Wert für jeden KMS-Schlüssel.
Operation	Wert für jeden AWS KMS API-Vorgang. Diese Metrik gilt nur für kryptografische Operationen.

Bei ReEncryptVorgängen umfasst die `SuccessfulRequest` Metrik Dimensionen sowohl für den Quell- als auch für den Ziel-KMS-Schlüssel.

Dimension	Beschreibung
SourceKeyArn	Wert für den KMS-Schlüssel, der den Chiffretext entschlüsselt hat.
DestinationKeyArn	Wert für den KMS-Schlüssel, der die Daten erneut verschlüsselt hat.
Operation	Wert für jeden AWS KMS API-Vorgang, in diesem Fall ReEncrypt.

SecondsUntilKeyMaterialExpiration

Die Anzahl der Sekunden, die bis zum frühesten ablaufenden importierten Schlüsselmaterial in einem KMS-Schlüssel verbleiben. Diese Metrik gilt nur für KMS-Schlüssel mit importiertem Schlüsselmaterial (Herkunft des Schlüsselmaterials `EXTERNAL`) und einem Ablaufdatum.

Verwenden Sie diese Metrik, um zu verfolgen, wie viel Zeit noch verbleibt, bis Ihr importiertes Schlüsselmaterial abläuft, das am frühesten abläuft. Wenn diese Zeit unter einen von Ihnen definierten Schwellenwert fällt, sollten Sie das Schlüsselmaterial mit einem neuen Ablaufdatum erneut importieren, damit der KMS-Schlüssel weiterhin verwendet werden kann. Die `SecondsUntilKeyMaterialExpiration`-Metrik ist spezifisch für einen KMS-Schlüssel. Sie können diese Metrik nicht verwenden, um mehrere KMS-Schlüssel oder KMS-Schlüssel, die Sie möglicherweise in Zukunft erstellen werden, zu überwachen. Hilfe zum Erstellen eines CloudWatch Alarms zur Überwachung dieser Metrik finden Sie unterEinen CloudWatch Alarm für den Ablauf von importiertem Schlüsselmaterial erstellen.

Die nützlichste Statistik für diese Metrik ist `Minimum`, die Ihnen die kleinste verbleibende Zeit für alle Datenpunkte im angegebenen Statistikzeitraum angibt. Die einzige gültige Einheit für diese Metrik ist `Seconds`.

Name der Dimensionsgruppe: Per-Key Metrics (Metriken pro Schlüssel)

Dimensionen für **`SecondsUntilKeyMaterialExpiration`**

Dimension	Beschreibung; bezieht sich auf AWS
KeyId	Wert für jeden KMS-Schlüssel.

Wenn Sie für einen KMS-Schlüssel das Löschen planen, erzwingt AWS KMS vor dem Löschen des KMS-Schlüssels eine Wartezeit. In der Wartezeit können Sie sicherzustellen, dass Sie den KMS-Schlüssel jetzt oder in der Zukunft nicht mehr benötigen. Sie können auch einen CloudWatch Alarm konfigurieren, der Sie warnt, wenn eine Person oder Anwendung während der Wartezeit versucht, den KMS-Schlüssel in einem kryptografischen Vorgang zu verwenden. Wenn Sie eine Benachrichtigung von einem solchen Alarm erhalten, können Sie das Löschen des KMS-Schlüssel abbrechen.

Detaillierte Anweisungen finden Sie unter Erstellen Sie einen Alarm, der die Verwendung eines KMS-Schlüssels erkennt, dessen Löschung noch aussteht.

Wolke HSMKey StoreThrottle

Die Anzahl der Anfragen für kryptografische Operationen mit KMS-Schlüsseln in jedem AWS CloudHSM Schlüsselspeicher, der AWS KMS gedrosselt wird (mit einem antwortet). `ThrottlingException` Diese Metrik gilt nur für Schlüsselspeicher. AWS CloudHSM

Die `CloudHSMKeyStoreThrottle` Metrik gilt nur für KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher und nur für Anfragen für kryptografische Operationen. AWS KMS drosselt diese Anfragen, wenn die Anforderungsrate das benutzerdefinierte Schlüsselspeicher-Anforderungskontingent für Ihren AWS CloudHSM Schlüsselspeicher überschreitet. Diese Metrik beinhaltet auch die Drosselung durch den Cluster. AWS CloudHSM

Name der Dimensionsgruppe: Keystore Throttle Metrics (Keystore-Drossel-Metriken)

Dimension	Beschreibung
CustomKeyStoreId	Wert für jeden AWS CloudHSM Schlüsselspeicher.
KmsOperation	Wert für jeden AWS KMS API-Vorgang. Diese Metrik gilt nur für kryptografische Operationen mit KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher.
KeySpec	Wert für jeden Typ von KMS-Schlüssel. Die einzige unterstützte Schlüsselspezifikation für KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher ist SYMMETRIC_DEFAULT.

ExternalKeyStoreThrottle

Die Anzahl der Anfragen für kryptografische Operationen mit KMS-Schlüsseln in jedem externen Schlüsselspeicher, der AWS KMS gedrosselt wird (mit einem antwortet). `ThrottlingException` Diese Metrik gilt nur für externe Schlüsselspeicher.

Die `ExternalKeyStoreThrottle` Metrik gilt nur für KMS-Schlüssel in einem externen Schlüsselspeicher und nur für Anfragen für kryptografische Operationen. AWS KMS drosselt diese Anfragen, wenn die Anforderungsrate das Anforderungskontingent für benutzerdefinierte Schlüsselspeicher für Ihren externen Schlüsselspeicher überschreitet. Diese Metrik berücksichtigt nicht die Drosselung durch Ihren externen Schlüsselspeicher-Proxy oder externen Schlüsselmanager.

Verwenden Sie diese Metrik, um den Wert Ihres Anforderungskontingents für benutzerdefinierte Schlüsselspeicher zu überprüfen und anzupassen. Wenn diese Metrik darauf hindeutet, AWS KMS dass Ihre Anfragen für diese KMS-Schlüssel häufig gedrosselt werden, sollten Sie erwägen, eine Erhöhung Ihres Kontingents für benutzerdefinierte Schlüsselspeicher-Anfragen zu beantragen. Hilfe finden Sie unter Anfordern einer Kontingenterhöhung im Service-Quotas-Benutzerhandbuch.

Wenn Sie sehr häufig `KMSInvalidStateException`-Fehler mit einer Meldung erhalten, in der erklärt wird, dass die Anforderung „aufgrund einer sehr hohen Anforderungsrate“ abgelehnt wurde oder die Anforderung abgelehnt wurde, „weil der externe Schlüsselspeicher-Proxy nicht rechtzeitig geantwortet hat“, könnte dies darauf hinweisen, dass Ihr externer Schlüsselmanager oder externer Schlüsselspeicher-Proxy mit der aktuellen Anforderungsrate nicht Schritt halten kann. Verringen Sie wenn möglich Ihre Anforderungsrate. Sie können auch eine Verringerung des Anforderungskontingentwerts für benutzerdefinierte Schlüsselspeicher anfordern. Eine Verringerung dieses Kontingentwerts kann die Drosselung (und den `ExternalKeyStoreThrottle` Metrikwert) erhöhen, bedeutet aber, dass AWS KMS überschüssige Anfragen schnell zurückgewiesen werden, bevor sie an Ihren externen Schlüsselspeicher-Proxy oder externen Schlüsselmanager gesendet werden. Um eine Reduzierung des Kontingents zu beantragen, besuchen Sie bitte das AWS -Support Center und erstellen Sie einen Fall.

Name der Dimensionsgruppe: Keystore Throttle Metrics (Keystore-Drossel-Metriken)

Dimension	Beschreibung
CustomKeyStoreId	Wert für jeden externen Schlüsselspeicher.
KmsOperation	Wert für jeden API-Vorgang AWS KMS . Diese Metrik gilt nur für kryptografische Operationen mit KMS-Schlüsseln in einem externen Schlüsselspeicher.
KeySpec	Wert für jeden Typ von KMS-Schlüssel. Die einzige unterstützte Schlüsselspezifikation für KMS-Schlüssel in einem externen Schlüsselspeicher ist SYMMETRIC_DEFAULT.

XksProxyCertificateDaysToExpire

Die Anzahl der Tage, bis das TLS-Zertifikat für den Proxy-Endpunkt Ihres externen Schlüsselspeichers (`XksProxyUriEndpoint`) abläuft. Diese Metrik gilt nur für externe Schlüsselspeicher.

Verwenden Sie diese Metrik, um einen CloudWatch Alarm zu erstellen, der Sie über den bevorstehenden Ablauf Ihres TLS-Zertifikats informiert. Wenn das Zertifikat abläuft, AWS KMS kann nicht mit dem externen Schlüsselspeicher-Proxy kommuniziert werden. Der Zugriff auf alle durch KMS-Schlüssel geschützten Daten in Ihrem externen Schlüsselspeicher ist dann erst wieder möglich, wenn Sie das Zertifikat erneuern.

Ein Zertifikatsalarm verhindert, dass ein abgelaufenes Zertifikat den Zugriff auf Ihre verschlüsselten Ressourcen verhindert. Stellen Sie den Alarm so ein, dass Ihre Organisation Zeit hat, das Zertifikat zu erneuern, bevor es abläuft.

Name der Dimensionsgruppe: XKS Proxy Certificate Metrics (XKS-Proxy-Zertifikatsmetriken)

Dimension	Beschreibung
CustomKeyStoreId	Wert für jeden externen Schlüsselspeicher.
CertificateName	Subjektname (CN) im TLS-Zertifikat.

Sie können CloudWatch Alarme auf der Grundlage der Metriken für externe Schlüsselspeicher und KMS-Schlüssel in externen Schlüsselspeichern erstellen. Detaillierte Anweisungen finden Sie unter Überwachen Sie externe Schlüsselspeicher.

XksProxyCredentialAge

Die Anzahl der Tage, seit die aktuelle Anmeldeinformation für die Proxy-Authentifizierung des externen Schlüsselspeichers (`XksProxyAuthenticationCredential`) mit dem externen Schlüsselspeicher verknüpft wurde. Diese Zählung beginnt, wenn Sie die Anmeldeinformation für die Authentifizierung beim Erstellen oder Aktualisieren Ihres externen Schlüsselspeichers eingeben. Diese Metrik gilt nur für externe Schlüsselspeicher.

Dieser Wert soll Sie daran erinnern, wie alt Ihre Anmeldeinformation für die Authentifizierung ist. Da wir jedoch mit der Zählung beginnen, wenn Sie die Anmeldeinformation mit Ihrem externen Schlüsselspeicher verknüpfen, und nicht, wenn Sie Ihre Anmeldeinformation für die Authentifizierung auf dem Proxy des externen Schlüsselspeichers erstellen, ist dies möglicherweise kein genauer Indikator für das Alter der Anmeldeinformation auf dem Proxy.

Verwenden Sie diese Metrik, um einen CloudWatch Alarm zu erstellen, der Sie daran erinnert, Ihre Proxyauthentifizierungsdaten für den externen Schlüsselspeicher zu wechseln.

Name der Dimensionsgruppe: Per-Keystore Metrics (Metriken pro Keystore)

Dimension	Beschreibung
CustomKeyStoreId	Wert für jeden externen Schlüsselspeicher.

Sie können CloudWatch Alarme auf der Grundlage der Metriken für externe Schlüsselspeicher und KMS-Schlüssel in externen Schlüsselspeichern erstellen. Detaillierte Anweisungen finden Sie unter Überwachen Sie externe Schlüsselspeicher.

XksProxyErrors

Die Anzahl der Ausnahmen im Zusammenhang mit AWS KMS Anfragen an Ihren externen Schlüsselspeicher-Proxy. Diese Anzahl umfasst Ausnahmen, zu denen der externe Schlüsselspeicher-Proxy zurückkehrt, AWS KMS und Timeoutfehler, die auftreten, wenn der externe Schlüsselspeicher-Proxy nicht AWS KMS innerhalb des Timeout-Intervalls von 250 Millisekunden reagiert. Diese Metrik gilt nur für externe Schlüsselspeicher.

Mit dieser Metrik können Sie die Fehlerquote von KMS-Schlüsseln in Ihrem externen Schlüsselspeicher verfolgen. Sie gibt Aufschluss über die häufigsten Fehler, sodass Sie Ihre Entwicklungsarbeit nach Prioritäten ordnen können. KMS-Schlüssel, die eine hohe Rate an nicht wiederholbaren Fehlern generieren, könnten beispielsweise auf ein Problem mit der Konfiguration Ihres externen Schlüsselspeichers hinweisen. Informationen zur Konfiguration Ihres externen Schlüsselspeichers finden Sie unter Externe Schlüsselspeicher anzeigen. Informationen zum Bearbeiten der Einstellungen Ihres externen Schlüsselspeichers finden Sie unter Eigenschaften des externen Schlüsselspeichers bearbeiten.

Name der Dimensionsgruppe: XKS Proxy Error Metrics (XKS-Proxy-Fehlermetriken)

Dimension	Beschreibung
CustomKeyStoreId	Wert für jeden externen Schlüsselspeicher.

Dimension	Beschreibung
KmsOperation	Wert für jeden AWS KMS API-Vorgang, der eine Anfrage an den XKS-Proxy generiert hat.
XksOperation	Wert für jeden Proxy-API-Vorgang des externen Schlüsselspeichers.
KeySpec	Wert für jeden Typ von KMS-Schlüssel. Die einzige unterstützte Schlüsselspezifikation für KMS-Schlüssel in einem externen Schlüsselspeicher ist SYMMETRIC_DEFAULT.
ErrorType	Werte: • Wiederholbare Fehler: Wahrscheinlich vorübergehend, z. B. Netzwerkfehler. • Nicht wiederholbare Fehler: Weisen wahrscheinlich auf ein Problem mit der Konfiguration des benutzerdefinierten Schlüsselspeichers oder mit externen Komponenten hin. • N/V: Erfolgreiche Anforderung; keine Fehler
Exception Name	Werte: • Name der Ausnahme • Keine: Erfolgreiche Anforderung; keine Fehler

Sie können CloudWatch Alarme auf der Grundlage der Metriken für externe Schlüsselspeicher und KMS-Schlüssel in externen Schlüsselspeichern erstellen. Detaillierte Anweisungen finden Sie unter Überwachen Sie externe Schlüsselspeicher.

XksExternalKeyManagerStates

Die Anzahl der Instances des externen Schlüsselmanagers in jedem der folgenden Zustände: `Active`, `Degraded` und `Unavailable`. Die Informationen für diese Metrik stammen von dem externen Schlüsselspeicher-Proxy, der jedem externen Schlüsselspeicher zugeordnet ist. Diese Metrik gilt nur für externe Schlüsselspeicher.

Nachfolgend sind die Zustände für die Instances des externen Schlüsselmanagers aufgeführt, die mit einem externen Schlüsselspeicher verbunden sind. Jeder Proxy für einen externen

Schlüsselspeicher verwendet möglicherweise andere Indikatoren zur Messung des Zustands Ihres externen Schlüsselmanagers. Weitere Informationen finden Sie in der Dokumentation Ihres externen Schlüsselspeicher-Proxys.

- `Active`: Der externe Schlüsselmanager ist fehlerfrei.
- `Degraded`: Der externe Schlüsselmanager ist fehlerhaft, kann aber trotzdem Datenverkehr bereitstellen.
- `Unavailable`: Der externe Schlüsselmanager kann keinen Datenverkehr bereitstellen.

Verwenden Sie diese Metrik, um einen CloudWatch Alarm zu erstellen, der Sie auf heruntergestufte und nicht verfügbare externe Key-Manager-Instanzen aufmerksam macht. Sehen Sie in den Protokollen Ihres externen Schlüsselspeicher-Proxys nach, welche Instances des externen Schlüsselmanagers sich in welchem Zustand befinden.

Name der Dimensionsgruppe: XKS External Key Manager Metrics (XKS-Metriken für externe Key Manager)

Dimension	Beschreibung
CustomKeyStoreId	Wert für jeden externen Schlüsselspeicher.
XksExternalKeyManagerState	Wert für jeden Zustand.

Sie können CloudWatch Alarme auf der Grundlage der Metriken für externe Schlüsselspeicher und KMS-Schlüssel in externen Schlüsselspeichern erstellen. Detaillierte Anweisungen finden Sie unter Überwachen Sie externe Schlüsselspeicher.

XksProxyLatency

Die Anzahl der Millisekunden, die ein externer Schlüsselspeicher-Proxy für die Antwort auf eine AWS KMS -Anforderung benötigt. Wenn die Anforderung eine Zeitüberschreitung aufweist, entspricht der aufgezeichnete Wert der Zeitüberschreitungsgrenze von 250 Millisekunden. Diese Metrik gilt nur für externe Schlüsselspeicher.

Verwenden Sie diese Metrik, um die Leistung Ihres externen Schlüsselspeicher-Proxys und Ihres externen Schlüsselmanagers zu bewerten. Wenn der Proxy beispielsweise bei Verschlüsselungs- und Entschlüsselungsvorgängen häufig eine Zeitüberschreitung aufweist, wenden Sie sich an Ihren externen Proxy-Administrator.

Langsame Antworten können auch darauf hindeuten, dass Ihr externer Schlüsselmanager den aktuellen Anforderungsverkehr nicht bewältigen kann. AWS KMS empfiehlt, dass Ihr externer Schlüsselmanager bis zu 1800 Anfragen für kryptografische Operationen pro Sekunde verarbeiten kann. Wenn Ihr externer Schlüsselmanager die Rate von 1 800 Anforderungen pro Sekunde nicht bewältigen kann, sollten Sie eine Verringerung Ihrer Anforderungsquote für KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher anfordern. Anforderungen für kryptografische Vorgänge, die die KMS-Schlüssel in Ihrem externen Schlüsselspeicher verwenden, schlagen schnell mit einer Drosselungsausnahme fehl, anstatt verarbeitet und später von Ihrem externen Schlüsselspeicher-Proxy oder externen Schlüsselmanager abgelehnt zu werden.

Name der Dimensionsgruppe: XKS Proxy Latency Metrics (XKS-Proxy-Latenzmetriken)

Dimension	Beschreibung
CustomKeyStoreId	Wert für jeden externen Schlüsselspeicher.
KmsOperation	Wert für jeden AWS KMS API-Vorgang, der eine Anfrage an den XKS-Proxy generiert hat.
XksOperation	Wert für jeden Proxy-API-Vorgang des externen Schlüsselspeichers.
KeySpec	Wert für jeden Typ von KMS-Schlüssel. Die einzige unterstützte Schlüsselspezifikation für KMS-Schlüssel in einem externen Schlüsselspeicher ist SYMMETRIC_DEFAULT.

Sie können CloudWatch Alarme auf der Grundlage der Metriken für externe Schlüsselspeicher und KMS-Schlüssel in externen Schlüsselspeichern erstellen. Detaillierte Anweisungen finden Sie unter Überwachen Sie externe Schlüsselspeicher.

Einen CloudWatch Alarm für den Ablauf von importiertem Schlüsselmaterial erstellen

Sie können einen CloudWatch Alarm erstellen, der Sie benachrichtigt, wenn sich das in einem KMS-Schlüssel importierte Schlüsselmaterial seiner Ablaufzeit nähert. Der Alarm kann Sie beispielsweise benachrichtigen, wenn die Zeit in weniger als 30 Tagen abläuft.

Wenn Sie Schlüsselmaterial in einen KMS-Schlüssel importieren, können Sie optional festlegen, an welchem Datum und zu welcher Uhrzeit das Schlüsselmaterial abläuft. Wenn das Schlüsselmaterial abläuft, wird das Schlüsselmaterial AWS KMS gelöscht und der KMS-Schlüssel wird unbrauchbar. Wenn Sie den KMS-Schlüssel erneut nutzen möchten, müssen Sie das Schlüsselmaterial erneut importieren. Wenn Sie das Schlüsselmaterial erneut importieren, bevor es abläuft, können Sie jedoch verhindern, dass Prozesse unterbrochen werden, die diesen KMS-Schlüssel verwenden.

Dieser Alarm verwendet die `SecondsUntilKeyMaterialExpiresMetrik`, die AWS KMS veröffentlicht wird, um CloudWatch KMS-Schlüssel mit importiertem Schlüsselmaterial zu veröffentlichen, das abläuft. Jeder Alarm verwendet diese Metrik, um das importierte Schlüsselmaterial für einen bestimmten KMS-Schlüssel zu überwachen. Es ist nicht möglich, einen einzelnen Alarm für alle KMS-Schlüssel mit ablaufendem Schlüsselmaterial oder einen Alarm für KMS-Schlüssel, die Sie möglicherweise in Zukunft erstellen werden, zu erstellen.

Voraussetzungen

Die folgenden Ressourcen sind für einen CloudWatch Alarm erforderlich, der den Ablauf von importiertem Schlüsselmaterial überwacht.

- Ein KMS-Schlüssel mit importiertem Schlüsselmaterial, das abläuft.
- Amazon SNS-Thema. Einzelheiten finden Sie unter Erstellen eines Amazon SNS SNS-Themas im CloudWatch Amazon-Benutzerhandbuch.

Den Alarm erstellen

Folgen Sie den Anweisungen unter Erstellen Sie einen CloudWatch Alarm auf der Grundlage eines statischen Schwellenwerts und verwenden Sie dabei die folgenden erforderlichen Werte. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Value (Wert)
Metrik auswählen	Wählen Sie KMS und dann Per-Key Metrics (Metriken pro Schlüssel) aus. Wählen Sie die Zeile mit dem KMS-Schlüssel und der `SecondsUntilKeyMaterialExpires` -Metrik aus. Wählen Sie dann Select Metric (Metrik auswählen) aus. Die Liste Metrics (Metriken) zeigt die `SecondsUntilKeyMaterialExpires` -Metrik nur für KMS-Schlüssel an, deren importiertes Schlüsselmaterial abläuft. Wenn Sie keine KMS-Schlüssel mit diesen Eigenschaften im Konto und in der Region haben, ist diese Liste leer.
Statistik	Minimum
Intervall	1 Minute
Threshold-Typ	Statisch
Immer, wenn ...	Wann immer größer ***metric-name*** ist als 1

CloudWatch Alarme für externe Schlüsselspeicher erstellen

Sie können CloudWatch Amazon-Alarme auf der Grundlage externer Key Store-Metriken erstellen, um Sie zu benachrichtigen, wenn ein Metrikwert einen von Ihnen angegebenen Schwellenwert überschreitet. Der Alarm kann die Nachricht an ein Amazon Simple Notification Service (Amazon SNS) -Thema oder eine Amazon EC2 Auto Scaling Scaling-Richtlinie senden. Ausführliche Informationen zu CloudWatch Alarmen finden Sie unter Verwenden von CloudWatch Amazon-Alarmen im CloudWatch Amazon-Benutzerhandbuch.

Bevor Sie einen CloudWatch Amazon-Alarm erstellen können, benötigen Sie ein Amazon SNS-Thema. Einzelheiten finden Sie unter Erstellen eines Amazon SNS SNS-Themas im CloudWatch Amazon-Benutzerhandbuch.

Themen

- Erstellen Sie einen Alarm für den Ablauf des Zertifikats
- Erstellen Sie einen Alarm für das Antwort-Timeout
- Erstellen Sie einen Alarm für Fehler, die wiederholt werden können

- Erstellen Sie einen Alarm für Fehler, die nicht wiederholt werden können

Erstellen Sie einen Alarm für den Ablauf des Zertifikats

Dieser Alarm verwendet die XksProxyCertificateDaysToExpire Metrik, die AWS KMS veröffentlicht wird, CloudWatch um den voraussichtlichen Ablauf des TLS-Zertifikats aufzuzeichnen, das Ihrem externen Schlüsselspeicher-Proxyendpunkt zugeordnet ist. Sie können nicht einen einzigen Alarm für alle externen Schlüsselspeicher in Ihrem Konto oder einen Alarm für externe Schlüsselspeicher erstellen, die Sie möglichweise in Zukunft erstellen.

Wir empfehlen, den Alarm so einzustellen, dass Sie 10 Tage vor Ablauf Ihres Zertifikats gewarnt werden, aber Sie sollten den Schwellenwert festlegen, der Ihren Bedürfnissen am besten entspricht.

Den Alarm erstellen

Folgen Sie den Anweisungen unter Erstellen eines CloudWatch Alarms auf der Grundlage eines statischen Schwellenwerts mithilfe der folgenden erforderlichen Werte. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Value (Wert)
Metrik auswählen	Wählen Sie KMS und dann XKS Proxy Certificate Metrics (XKS-Proxy-Zertifikatsmetriken). Aktivieren Sie das Kontrollkästchen neben dem `XksProxyCertificateName` , den Sie überwachen möchten. Wählen Sie dann Select Metric (Metrik auswählen) aus.
Statistik	Minimum
Intervall	5 Minuten
Threshold-Typ	Statisch
Immer, wenn ...	Wann immer XksProxyCertificateDaysToExpireist `Lower` dann`10`.

Erstellen Sie einen Alarm für das Antwort-Timeout

Dieser Alarm verwendet die XksProxyLatency Metrik, die AWS KMS veröffentlicht wird, CloudWatch um aufzuzeichnen, wie viele Millisekunden ein externer Schlüsselspeicher-Proxy benötigt, um auf eine Anfrage zu antworten. AWS KMS Sie können nicht einen einzigen Alarm für alle externen Schlüsselspeicher in Ihrem Konto oder einen Alarm für externe Schlüsselspeicher erstellen, die Sie möglichweise in Zukunft erstellen.

AWS KMS erwartet, dass der externe Schlüsselspeicher-Proxy auf jede Anfrage innerhalb von 250 Millisekunden reagiert. Wir empfehlen, einen Alarm einzustellen, der Sie benachrichtigt, wenn Ihr externer Schlüsselspeicher-Proxy länger als 200 Millisekunden braucht, um zu antworten, aber Sie sollten den Schwellenwert festlegen, der Ihren Bedürfnissen am besten entspricht.

Den Alarm erstellen

Folgen Sie den Anweisungen unter Erstellen eines CloudWatch Alarms auf der Grundlage eines statischen Schwellenwerts und verwenden Sie dabei die folgenden erforderlichen Werte. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Value (Wert)
Metrik auswählen	Wählen Sie KMS und dann XKS Proxy Latency Metrics (XKS-Proxy-Latenzmetriken). Aktivieren Sie das Kontrollkästchen neben dem `KmsOperation` , den Sie überwachen möchten. Wählen Sie dann Select Metric (Metrik auswählen) aus.
Statistik	Durchschnitt
Intervall	5 Minuten
Threshold-Typ	Statisch
Immer, wenn ...	Wann immer XksProxyLatencyist `Greater` dann`200`.

Erstellen Sie einen Alarm für Fehler, die wiederholt werden können

Dieser Alarm verwendet die XksProxyErrors Metrik, die AWS KMS veröffentlicht wird, CloudWatch um die Anzahl der Ausnahmen im Zusammenhang mit AWS KMS Anfragen an Ihren externen Schlüsselspeicher-Proxy aufzuzeichnen. Sie können nicht einen einzigen Alarm für alle externen Schlüsselspeicher in Ihrem Konto oder einen Alarm für externe Schlüsselspeicher erstellen, die Sie möglichweise in Zukunft erstellen.

Wiederholbare Fehler verringern Ihren Zuverlässigkeitsgrad und können auf Netzwerkfehler hinweisen. Wir empfehlen, einen Alarm einzustellen, der Sie warnt, wenn mehr als fünf wiederholbare Fehler innerhalb einer Minute aufgezeichnet werden, aber Sie sollten den Schwellenwert festlegen, der Ihren Bedürfnissen am besten entspricht.

Folgen Sie den Anweisungen unter Erstellen eines CloudWatch Alarms auf der Grundlage eines statischen Schwellenwerts mithilfe der folgenden erforderlichen Werte. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Value (Wert)
Metrik auswählen	Wählen Sie die Registerkarte Queries (Abfragen) aus. Wählen Sie `AWS/KMS` als Namespace aus. Geben Sie `SUM(XksProxyErrors)` als Metriknamen ein. Geben Sie `ErrorType = Retryable` bei Filter by (Filtern nach) ein. Wählen Sie Ausführen aus. Wählen Sie dann Select Metric (Metrik auswählen) aus.
Label (Bezeichnung)	*`Retryable errors`*
Intervall	1 Minute
Threshold-Typ	Statisch
Immer, wenn ...	Immer wenn q1 `Greater` als 5 ist.

Erstellen Sie einen Alarm für Fehler, die nicht wiederholt werden können

Dieser Alarm verwendet die XksProxyErrors Metrik, die AWS KMS veröffentlicht wird, CloudWatch um die Anzahl der Ausnahmen im Zusammenhang mit AWS KMS Anfragen an Ihren externen Schlüsselspeicher-Proxy aufzuzeichnen. Sie können nicht einen einzigen Alarm für alle externen Schlüsselspeicher in Ihrem Konto oder einen Alarm für externe Schlüsselspeicher erstellen, die Sie möglichweise in Zukunft erstellen.

Nicht wiederholbare Fehler können auf ein Problem mit der Konfiguration Ihres externen Schlüsselspeichers hinweisen. Wir empfehlen, einen Alarm einzustellen, der Sie warnt, wenn mehr als fünf nicht wiederholbare Fehler innerhalb einer Minute aufgezeichnet werden, aber Sie sollten den Schwellenwert festlegen, der Ihren Bedürfnissen am besten entspricht.

Folgen Sie den Anweisungen unter Erstellen eines CloudWatch Alarms auf der Grundlage eines statischen Schwellenwerts mithilfe der folgenden erforderlichen Werte. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Value (Wert)
Metrik auswählen	Wählen Sie die Registerkarte Queries (Abfragen) aus. Wählen Sie `AWS/KMS` als Namespace aus. Geben Sie `SUM(XksProxyErrors)` als Metriknamen ein. Geben Sie `ErrorType = Non-retryable` bei Filter by (Filtern nach) ein. Wählen Sie Ausführen aus. Wählen Sie dann Select Metric (Metrik auswählen) aus.
Label (Bezeichnung)	*`Non-retryable errors`*
Intervall	1 Minute
Threshold-Typ	Statisch
Immer, wenn ...	Immer wenn q1 `Greater` als 5 ist.

Überwachen Sie KMS-Schlüssel mit Amazon EventBridge

Sie können Amazon EventBridge (ehemals Amazon CloudWatch Events) verwenden, um Sie auf die folgenden wichtigen Ereignisse im Lebenszyklus Ihrer KMS-Schlüssel aufmerksam zu machen.

- Das Schlüsselmaterial in einem KMS-Schlüssel wurde automatisch oder bei Bedarf rotiert.
- Das importierte Schlüsselmaterial in einem KMS-Schlüssel ist abgelaufen.
- Ein KMS-Schlüssel, dessen Löschung geplant war, wurde gelöscht.

AWS KMS integriert sich in Amazon EventBridge , um Sie über wichtige Ereignisse zu informieren, die sich auf Ihre KMS-Schlüssel auswirken. Jedes Ereignis wird in JSON (JavaScriptObject Notation) dargestellt und enthält den Namen des Ereignisses, das Datum und die Uhrzeit, zu der das Ereignis eingetreten ist, sowie die betroffenen Ereignisse. Sie können diese Ereignisse sammeln und Regeln einrichten, die sie an ein oder mehrere Ziele weiterleiten, z. B. AWS Lambda Funktionen, Amazon SNS SNS-Themen, Amazon SQS SQS-Warteschlangen, Streams in Amazon Kinesis Data Streams oder integrierte Ziele.

Weitere Informationen zur Verwendung EventBridge mit anderen Arten von Ereignissen, einschließlich solcher, die AWS CloudTrail bei der Aufzeichnung einer read/write API-Anfrage ausgelöst werden, finden Sie im EventBridge Amazon-Benutzerhandbuch.

In den folgenden Themen werden die EventBridge Ereignisse beschrieben, die AWS KMS generiert werden.

KMS-CMK-Drehung

AWS KMS unterstützt die automatische und bedarfsgesteuerte Rotation des Schlüsselmaterials in KMS-Schlüsseln mit symmetrischer Verschlüsselung.

Immer wenn Schlüsselmaterial AWS KMS rotiert wird, wird ein `KMS CMK Rotation` Ereignis an gesendet. EventBridge AWS KMS generiert dieses Ereignis nach bestem Wissen und Gewissen.

Es folgt ein Beispiel für dieses Ereignis.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "KMS CMK Rotation",
```

```
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "key-origin": "AWS_KMS",
    "rotation-type": "ON_DEMAND",
    "previous-key-material-id":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
    "current-key-material-id":
 "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068"
  }
}
```

Ablauf der über KMS importierten Schlüsselinformationen

Wenn Sie Schlüsselmaterial in einen KMS-Schlüssel importieren, können Sie optional festlegen, wann das Schlüsselmaterial abläuft. Wenn das Schlüsselmaterial abläuft, AWS KMS löscht es und sendet ein entsprechendes `KMS Imported Key Material Expiration` Ereignis an. EventBridge AWS KMS generiert dieses Ereignis nach bestem Wissen und Gewissen.

Es folgt ein Beispiel für dieses Ereignis.

```
{
  "version": "0",
  "id": "9da9af57-9253-4406-87cb-7cc400e43465",
  "detail-type": "KMS Imported Key Material Expiration",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "key-material-id":
 "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  }
```

```
}
```

KMS-CMK-Löschung

Wenn Sie für einen KMS-Schlüssel das Löschen planen, erzwingt AWS KMS vor dem Löschen des KMS-Schlüssels eine Wartezeit. AWS KMS Löscht nach Ablauf der Wartezeit den KMS-Schlüssel und sendet ein `KMS CMK Deletion` Ereignis an. EventBridge AWS KMS garantiert dieses EventBridge Ereignis. Aufgrund von Wiederholungen kann es innerhalb weniger Sekunden mehrere Ereignisse generieren, die denselben KMS-Schlüssel löschen.

Es folgt ein Beispiel für dieses Ereignis.

```
{
  "version": "0",
  "id": "e9ce3425-7d22-412a-a699-e7a5fc3fbc9a",
  "detail-type": "KMS CMK Deletion",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab"
  }
}
```

Aliase in AWS KMS

Ein Alias ist ein Anzeigename für einen AWS KMS key. Mit einem Alias können Sie beispielsweise auf einen KMS-Schlüssel als `test-key`anstelle von `1234abcd-12ab-34cd-56ef-1234567890ab` verweisen.

Sie können einen Alias verwenden, um einen KMS-Schlüssel in der AWS KMS Konsole, im DescribeKeyVorgang und bei kryptografischen Vorgängen wie Verschlüsseln und zu identifizieren. GenerateDataKey Darüber hinaus erleichtern Aliasse das Erkennen eines Von AWS verwalteter Schlüssel. Aliase für diese KMS-Schlüssel haben immer die Form `aws/<service-name>`. Der Alias Von AWS verwalteter Schlüssel für Amazon DynamoDB lautet beispielsweise. `aws/dynamodb` Sie können ähnliche Alias-Standards für Ihre Projekte festlegen, z. B. den Namen eines Projekts oder einer Kategorie vorab für Ihre Aliasse.

Sie können den Zugriff auf KMS-Schlüssel auch anhand ihrer Aliasse zulassen und verweigern, ohne Richtlinien zu bearbeiten oder Erteilungen zu verwalten. Diese Funktion ist Teil der AWS KMS Unterstützung für die attributebasierte Zugriffskontrolle (ABAC). Details hierzu finden Sie unter Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern.

Ein Großteil der Möglichkeiten von Aliassen ergibt sich aus Ihrer Fähigkeit, den KMS-Schlüssel, der einem Alias zugeordnet ist, jederzeit zu ändern. Aliasse machen Ihren Code einfacher zu schreiben und zu verwalten. Angenommen, Sie verwenden einen Alias, um auf einen bestimmten KMS-Schlüssel zu verweisen, und Sie möchten den KMS-Schlüssel ändern. In diesem Fall ordnen Sie den Alias einfach einem anderen KMS-Schlüssel zu. Sie müssen keine Codeänderungen ausführen.

Darüber hinaus erleichtern Aliasse die Wiederverwendung des gleichen Codes in verschiedenen AWS-Regionen. Erstellen Sie Aliasse mit demselben Namen in mehreren Regionen und ordnen Sie jeden Alias einem KMS-Schlüssel in seiner Region zu. Wenn der Code in jeder Region ausgeführt wird, bezieht sich der Alias auf den zugeordneten KMS-Schlüssel in dieser Region. Ein Beispiel finden Sie unter Erfahren Sie, wie Sie Aliase in Ihren Anwendungen verwenden.

Sie können einen Alias für einen KMS-Schlüssel in der AWS KMS Konsole, mithilfe der CreateAliasAPI oder mithilfe der Vorlage erstellen. AWS::KMS::Alias AWS CloudFormation

Die AWS KMS API bietet die vollständige Kontrolle über Aliase in jedem Konto und jeder Region. Die API umfasst Operationen zum Erstellen eines Alias (CreateAlias), zum Anzeigen von Aliasnamen und Alias ARNs (ListAliases), zum Ändern des mit einem Alias (UpdateAlias) verknüpften KMS-Schlüssels und zum Löschen eines Alias (DeleteAlias).

Wie funktionieren Aliase

Erfahren Sie, wie Aliasse in AWS KMS funktionieren.

Ein Alias ist eine unabhängige Ressource AWS

Ein Alias ist keine Eigenschaft eines KMS-Schlüssels. Die Aktionen, die Sie für den Alias ausführen, wirken sich nicht auf den zugeordneten KMS-Schlüssel aus. Sie können einen Alias für einen KMS-Schlüssel erstellen und dann den Alias aktualisieren, damit er einem anderen KMS-Schlüssel zugeordnet ist. Sie können den Alias sogar löschen, ohne dass sich auf den zugeordneten KMS-Schlüssel auswirkt. Wenn Sie jedoch einen KMS-Schlüssel löschen, werden alle Aliasse gelöscht, die diesem KMS-Schlüssel zugeordnet sind.

Wenn Sie einen Alias als Ressource in einer IAM-Richtlinie angeben, bezieht sich die Richtlinie auf den Alias und nicht auf den zugeordneten KMS-Schlüssel.

Jeder Alias hat zwei Formate

Wenn Sie einen Alias erstellen, geben Sie den Aliasnamen an. AWS KMS erstellt den Alias ARN für Sie.

- Ein Alias-ARN ist ein Amazon-Ressourcenname (ARN), der den Alias eindeutig identifiziert.

```
# Alias ARN
arn:aws:kms:us-west-2:111122223333:alias/<alias-name>
```

- Der Aliasname muss im Konto und in der Region eindeutig sein. In der AWS KMS API wird dem Aliasnamen immer ein Präfix `alias/` vorangestellt. Dieses Präfix wird in der AWS KMS Konsole weggelassen.

```
# Alias name
alias/<alias-name>
```

Aliase sind nicht geheim

Aliase können in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden. Geben Sie keine vertraulichen oder sensiblen Informationen in den Alias-Namen ein.

Jeder Alias ist jeweils einem KMS-Schlüssel zugeordnet

Der Alias und sein KMS-Schlüssel müssen sich im selben Konto und in derselben Region befinden.

Sie können jedem vom Kunden verwalteten Schlüssel in derselben AWS-Konto Region einen Alias zuordnen. Sie haben jedoch keine Berechtigung, einen Alias einem Von AWS verwalteter Schlüssel zuzuordnen.

Diese ListAliasesAusgabe zeigt beispielsweise, dass der `test-key` Alias genau einem KMS-Zielschlüssel zugeordnet ist, der durch die `TargetKeyId` Eigenschaft dargestellt wird.

```
{
     "AliasName": "alias/test-key",
     "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
     "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
     "CreationDate": 1593622000.191,
     "LastUpdatedDate": 1593622000.191
}
```

Mehrere Aliasse können demselben KMS-Schlüssel zugeordnet werden

Beispielsweise können Sie die `test-key`- und `project-key`-Aliasse demselben KMS-Schlüssel zuordnen.

```
{
     "AliasName": "alias/test-key",
     "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
     "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
     "CreationDate": 1593622000.191,
     "LastUpdatedDate": 1593622000.191
},
{
     "AliasName": "alias/project-key",
     "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project-key",
     "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
     "CreationDate": 1516435200.399,
     "LastUpdatedDate": 1516435200.399
}
```

Ein Alias muss in einem Konto und einer Region eindeutig sein.

Zum Beispiel können Sie nur einen `test-key`-Alias in jedem Konto und in jeder Region haben. Aliasse beachten die Groß-/Kleinschreibung, aber Aliasse, die sich nur in ihrer Großschreibung unterscheiden, sind sehr fehleranfällig. Sie können keinen Aliasnamen ändern. Sie können den Alias jedoch löschen und einen neuen Alias mit dem gewünschten Namen erstellen.

Sie können jedoch einen Alias mit demselben Namen in verschiedenen Regionen erstellen.

Sie können beispielsweise einen `finance-key`-Alias In USA Ost (Nord-Virginia) und einen `finance-key`-Alias in Europa (Frankfurt) haben. Jeder Alias wäre einem KMS-Schlüssel in seiner Region zugeordnet. Wenn Ihr Code auf einen Aliasnamen wie `alias/finance-key` verweist, können Sie ihn in mehreren Regionen ausführen. In jeder Region wird ein anderer KMS-Schlüssel verwendet. Details hierzu finden Sie unter Erfahren Sie, wie Sie Aliase in Ihren Anwendungen verwenden.

Sie können den KMS-Schlüssel ändern, der einem Alias zugeordnet ist.

Sie können den UpdateAliasVorgang verwenden, um einen Alias einem anderen KMS-Schlüssel zuzuordnen. Wenn beispielsweise der `finance-key`-Alias dem `1234abcd-12ab-34cd-56ef-1234567890ab`-KMS-Schlüssel zugeordnet ist, verwenden, können Sie ihn so aktualisieren, dass er dem `0987dcba-09fe-87dc-65ba-ab0987654321`-KMS-Schlüssel zugeordnet wird.

Der aktuelle KMS-Schlüssel und der neue KMS-Schlüssel müssen jedoch vom selben Typ sein (beide symmetrisch oder beide asymmetrisch oder beide HMAC) und die gleiche Schlüsselnutzung (ENCRYPT_DECRYPT oder SIGN_VERIFY oder GENERATE_VERIFY_MAC) aufweisen. Diese Beschränkung verhindert Fehler in Code, der Aliasse verwendet. Wenn Sie einen Alias einem anderen Schlüsseltyp zuordnen müssen und die Risiken verringert haben, können Sie den Alias löschen und neu erstellen.

Einige KMS-Schlüssel haben keine Aliasse

Wenn Sie einen KMS-Schlüssel in der AWS KMS Konsole erstellen, müssen Sie ihm einen neuen Alias geben. Ein Alias ist jedoch nicht erforderlich, wenn Sie den CreateKeyVorgang zum Erstellen eines KMS-Schlüssels verwenden. Sie können den UpdateAliasVorgang auch verwenden, um den KMS-Schlüssel zu ändern, der einem Alias zugeordnet ist, und den DeleteAliasVorgang zum Löschen eines Alias verwenden. Daher könnten manche KMS-Schlüssel mehrere Aliasse haben, während andere keinen haben.

AWS erstellt Aliase in Ihrem Konto

AWS erstellt Aliase in Ihrem Konto für. Von AWS verwaltete Schlüssel Diese Aliasse haben Namen des Formulars `alias/aws/<service-name>`, wie beispielsweise `alias/aws/s3`.

Einige AWS Aliase haben keinen KMS-Schlüssel. Diese vordefinierten Aliase sind normalerweise mit einem verknüpft Von AWS verwalteter Schlüssel , wenn Sie den Dienst nutzen.

Verwenden von Aliassen zum Identifizieren von KMS-Schlüsseln

Sie können einen Aliasnamen oder Alias-ARN verwenden, um einen KMS-Schlüssel in kryptografischen Vorgängen zu identifizieren DescribeKey, und GetPublicKey. (Wenn sich der KMS-Schlüssel in einem anderen AWS-Konto befindet, müssen Sie seinen Schlüssel-ARN oder Alias-ARN verwenden.) Aliasse sind keine gültigen Bezeichner für KMS-Schlüssel in anderen AWS KMS -Operationen. Informationen zu den gültigen Schlüsselkennungen für jeden AWS KMS API-Vorgang finden Sie in den Beschreibungen der `KeyId` Parameter in der AWS Key Management Service API-Referenz.

Sie können keinen Aliasnamen oder Alias-ARN verwenden, um einen KMS-Schlüssel in einer IAM-Richtlinie zu identifizieren. Um den Zugriff auf einen KMS-Schlüssel anhand seiner Aliase zu steuern, verwenden Sie die ResourceAliases Bedingungsschlüssel kms: RequestAlias oder kms:. Details hierzu finden Sie unter ABAC für AWS KMS.

Steuern des Zugriffs auf Aliasse

Wenn Sie einen Alias erstellen oder ändern, wirkt sich das auf den Alias und den zugehörigen KMS-Schlüssel aus. Daher müssen Prinzipale, die Aliasse verwalten, über die Berechtigung verfügen, die Alias-Operation für den Alias und alle betroffenen KMS-Schlüssel aufzurufen. Sie können diese Berechtigungen mithilfe von Schlüsselrichtlinien, IAM-Richtlinien und Erteilungen erteilen.

> **Note**
>
> Seien Sie vorsichtig, wenn Sie Prinzipalen die Berechtigung zum Verwalten von Tags und Aliassen erteilen. Wenn Sie eine Markierung oder einen Alias ändern, wird dadurch die Berechtigung für den kundenverwalteten Schlüssel erteilt oder verweigert. Details dazu finden Sie unter ABAC für AWS KMS und Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern.

Hinweise zur Steuerung des Zugriffs auf alle AWS KMS Operationen finden Sie unterBerechtigungsreferenz.

Berechtigungen zum Erstellen und Verwalten von Aliasen funktionieren wie folgt.

km: CreateAlias

Um einen Alias zu erstellen, benötigt der Prinzipal die folgenden Berechtigungen sowohl für den Alias als auch für den zugeordneten KMS-Schlüssel.

- `kms:CreateAlias` für den Alias. Geben Sie diese Berechtigung in einer IAM-Richtlinie an, die dem Prinzipal zugeordnet ist, der den Alias erstellen darf.

  Die folgende Beispiel-Richtlinienanweisung gibt einen bestimmten Alias in einem `Resource`-Element an. Sie können jedoch mehrere Alias auflisten ARNs oder ein Aliasmuster angeben, z. B. „test*“. Sie können auch einen `Resource`-Wert von `"*"` angeben, um dem Prinzipal zu erlauben, einen Alias im Konto und der Region zu erstellen. Die Berechtigung zum Erstellen eines Aliasses kann auch in einer `kms:Create*`-Berechtigung für alle Ressourcen in einem Konto und einer Region enthalten sein.

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- `kms:CreateAlias` für den KMS-Schlüssel. Diese Berechtigung muss in einer Schlüsselrichtlinie oder in einer IAM-Richtlinie bereitgestellt werden, die von der Schlüsselrichtlinie delegiert wird.

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:CreateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

Sie können Bedingungsschlüssel verwenden, um die KMS-Schlüssel einzuschränken, die Sie einem Alias zuordnen können. Sie können beispielsweise den KeySpec Bedingungsschlüssel kms: verwenden, um es dem Prinzipal zu ermöglichen, Aliase nur für asymmetrische KMS-Schlüssel zu erstellen. Eine vollständige Liste der Bedingungsschlüssel, mit denen Sie die `kms:CreateAlias`-Berechtigung für KMS-Schlüsselressourcen einschränken können, finden Sie unter AWS KMS Berechtigungen.

km: ListAliases

Um Aliasse im Konto und in der Region aufzulisten, muss der Prinzipal über `kms:ListAliases`-Berechtigung in einer IAM-Richtlinie verfügen. Da diese Richtlinie nicht mit einem bestimmten KMS-Schlüssel oder Alias-Ressource verwandt ist, muss der Wert des Ressourcenelements in der Richtlinie `"*"` sein.

Beispielsweise gibt die folgende IAM-Richtlinienanweisung dem Prinzipal die Berechtigung, alle KMS-Schlüssel und Aliasse im Konto und in der Region aufzulisten.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
}
```

km: UpdateAlias

Um den KMS-Schlüssel zu ändern, der einem Alias zugeordnet ist, benötigt der Prinzipal drei Berechtigungselemente: eines für den Alias, eines für den aktuellen KMS-Schlüssel und eines für den neuen KMS-Schlüssel.

Angenommen, Sie möchten z. B. den `test-key`-Alias aus dem KMS-Schlüssel mit Schlüssel-ID 1234abcd-12ab-34cd-56ef-1234567890ab zum KMS-Schlüssel mit Schlüssel-

ID 0987dcba-09fe-87dc-65ba-ab0987654321 ändern. Fügen Sie in diesem Fall ähnliche Richtlinienanweisungen wie in den Beispielen in diesem Abschnitt ein.

- `kms:UpdateAlias` für den Alias. Sie stellen diese Berechtigung in einer IAM-Richtlinie zur Verfügung, die dem Prinzipal angefügt ist. Die folgende IAM-Richtlinie gibt einen bestimmten Alias an. Sie können jedoch mehrere Alias auflisten ARNs oder ein Aliasmuster angeben, z. `"test*"` B. Sie können auch einen `Resource`-Wert von `"*"` angeben, um dem Prinzipal zu erlauben, einen Alias im Konto und der Region zu aktualisieren.

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:UpdateAlias",
    "kms:ListAliases",
    "kms:ListKeys"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- `kms:UpdateAlias` für den KMS-Schlüssel, der derzeit dem Alias zugeordnet ist. Diese Berechtigung muss in einer Schlüsselrichtlinie oder in einer IAM-Richtlinie bereitgestellt werden, die von der Schlüsselrichtlinie delegiert wird.

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:UpdateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

- `kms:UpdateAlias` für den KMS-Schlüssel, den die Operation dem Alias zuordnet. Diese Berechtigung muss in einer Schlüsselrichtlinie oder in einer IAM-Richtlinie bereitgestellt werden, die von der Schlüsselrichtlinie delegiert wird.

```
{
  "Sid": "Key policy for 0987dcba-09fe-87dc-65ba-ab0987654321",
```

```
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:UpdateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

Mit Hilfe von Bedingungsschlüsseln können Sie einen oder beide KMS-Schlüssel in einer `UpdateAlias`-Operation begrenzen. Sie können beispielsweise den ResourceAliases Bedingungsschlüssel kms: verwenden, damit der Prinzipal Aliase nur dann aktualisieren kann, wenn der KMS-Zielschlüssel bereits über einen bestimmten Alias verfügt. Eine vollständige Liste der Bedingungsschlüssel, mit denen Sie die `kms:UpdateAlias`-Berechtigung für KMS-Schlüsselressourcen einschränken können, finden Sie unter AWS KMS Berechtigungen.

km: DeleteAlias

Um einen Alias zu löschen, benötigt der Prinzipal die Berechtigung für den Alias und für den zugeordneten KMS-Schlüssel.

Wie immer sollten Sie Vorsicht walten lassen, wenn Sie Prinzipalen die Berechtigung zum Löschen einer Ressource erteilen. Das Löschen eines Aliasses hat keine Auswirkungen auf den zugehörigen KMS-Schlüssel. Es kann zwar zu einem Fehler in einer Anwendung führen, die auf dem Alias beruht, aber wenn Sie versehentlich einen Alias löschen, können Sie ihn neu erstellen.

- `kms:DeleteAlias` für den Alias. Sie stellen diese Berechtigung in einer IAM-Richtlinie zur Verfügung, die dem Prinzipal angefügt ist, der den Alias löschen darf.

  Die folgende Beispiel-Richtlinienanweisung gibt einen bestimmten Alias in einem `Resource`-Element an. Sie können jedoch mehrere Alias auflisten ARNs oder ein Aliasmuster angeben, z. B. Sie können auch den `Resource` Wert von angeben`"test*"`, `"*"` damit der Prinzipal alle Alias im Konto und in der Region löschen kann.

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
```

```
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- `kms:DeleteAlias` für den zugeordneten KMS-Schlüssel. Diese Berechtigung muss in einer Schlüsselrichtlinie oder in einer IAM-Richtlinie bereitgestellt werden, die von der Schlüsselrichtlinie delegiert wird.

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"
  },
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

Begrenzen von Alias-Berechtigungen

Sie können Bedingungsschlüssel verwenden, um Alias-Berechtigungen einzuschränken, wenn es sich bei der Ressource um einen KMS-Schlüssel handelt. Die folgende IAM-Richtlinie erlaubt beispielsweise Alias-Operationen für KMS-Schlüssel in einem bestimmten Konto und einer bestimmten Region. Es verwendet jedoch den KeyOrigin Bedingungsschlüssel kms:, um die Berechtigungen weiter auf KMS-Schlüssel mit Schlüsselmaterial von zu beschränken AWS KMS.

Eine vollständige Liste der Bedingungsschlüssel, mit denen Sie Alias-Berechtigungen für eine KMS-Schlüsselressource einschränken können, finden Sie unter AWS KMS Berechtigungen.

```
{
  "Sid": "IAMPolicyKeyPermissions",
  "Effect": "Allow",
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Action": [
    "kms:CreateAlias",
```

```
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "AWS_KMS"
    }
  }
}
```

Sie können keine Bedingungsschlüssel in einer Richtlinienanweisung verwenden, bei der die Ressource ein Alias ist. Um die Aliasse einzuschränken, die ein Prinzipal verwalten kann, verwenden Sie den Wert des `Resource`-Elements der IAM-Richtlinienanweisung, die den Zugriff auf den Alias steuert. Die folgenden Richtlinienanweisungen ermöglichen es dem Prinzipal beispielsweise, jeden Alias in der Region AWS-Konto und zu erstellen, zu aktualisieren oder zu löschen, sofern der Alias nicht mit beginnt`Restricted`.

```
{
  "Sid": "IAMPolicyForAnAliasAllow",
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/*"
},
{
  "Sid": "IAMPolicyForAnAliasDeny",
  "Effect": "Deny",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/Restricted*"
}
```

Aliase erstellen

Sie können Aliase in der AWS KMS Konsole oder mithilfe von AWS KMS API-Operationen erstellen.

Der Alias muss aus einer Zeichenfolge von 1–256 Zeichen bestehen. Er kann nur alphanumerische Zeichen (A-Z, a-z, 0-9), Bindestriche (-), Schrägstriche (/), Unterstriche (_) und Bindestriche (-) enthalten. Der Aliasname für einen kundenverwalteten Schlüssel kann nicht mit `alias/aws/`beginnen. Das Präfix `alias/aws/` ist für Von AWS verwalteter Schlüssel reserviert.

Sie können einen Alias für einen neuen KMS-Schlüssel oder einen vorhandenen KMS-Schlüssel erstellen. Sie können einen Alias hinzufügen, sodass ein bestimmter KMS-Schlüssel in einem Projekt oder einer Anwendung verwendet wird.

Sie können auch eine AWS CloudFormation Vorlage verwenden, um einen Alias für einen KMS-Schlüssel zu erstellen. Weitere Informationen finden Sie unter AWS::KMS::Alias im AWS CloudFormation -Benutzerhandbuch.

Verwenden der AWS KMS Konsole

Wenn Sie einen KMS-Schlüssel in der AWS KMS Konsole erstellen, müssen Sie einen Alias für den neuen KMS-Schlüssel erstellen. Um einen Alias für einen vorhandenen KMS-Schlüssel zu erstellen, verwenden Sie die Aliasse-Registerkarte auf der Detailseite für den KMS-Schlüssel.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. Sie können keine Aliase für oder verwalten. Von AWS verwaltete Schlüssel AWS-eigene Schlüssel
4. Wählen Sie in der Tabelle die Schlüssel-ID oder den Alias des KMS-Schlüssels. Wählen Sie dann auf der Detailseite für den KMS-Schlüssel die Aliasse-Registerkarte.

   Wenn ein KMS-Schlüssel über mehrere Aliasse verfügt, zeigt die Aliasse-Spalte in der Tabelle einen Alias und eine Aliasübersicht an, z. B. (+n mehr). Wenn Sie die Aliasübersicht auswählen, gelangen Sie direkt zur Aliases (Aliasse)-Registerkarte auf der Seite mit den KMS-Schlüsseldetails.
5. Wählen Sie auf der Aliasse-Registerkarte Create Alarm (Alarm erstellen). Geben Sie einen Aliasnamen ein und wählen Sie Create alias (Alias erstellen).

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

> **Note**
>
> Fügen Sie das Präfix `alias/` nicht hinzu. Die Konsole fügt es automatisch für Sie hinzu. Wenn Sie `alias/ExampleAlias` eingeben, wird der tatsächliche Aliasname `alias/alias/ExampleAlias` sein.

Verwenden der API AWS KMS

Verwenden Sie die CreateAliasOperation, um einen Alias zu erstellen. Im Gegensatz zum Erstellen von KMS-Schlüsseln in der Konsole wird bei CreateKeydiesem Vorgang kein Alias für einen neuen KMS-Schlüssel erstellt.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

Sie können die `CreateAlias`-Operation verwenden, um einen Alias für einen neuen KMS-Schlüssel ohne Alias zu erstellen. Sie können auch die `CreateAlias`-Operation verwenden, um einen Alias zu einem vorhandenen KMS-Schlüssel hinzuzufügen oder einen Alias neu zu erstellen, der versehentlich gelöscht wurde.

Bei den AWS KMS API-Vorgängen muss der Aliasname mit beginnen, `alias/` gefolgt von einem Namen, z. B. `alias/ExampleAlias` Ein Alias muss im Konto und der Region eindeutig sein. Verwenden Sie die ListAliasesOperation, um die Aliasnamen zu finden, die bereits verwendet werden. Bei dem Aliasnamen wird zwischen Groß-/Kleinschreibung unterschieden.

Die `TargetKeyId` kann jeder kundenverwaltete Schlüssel in der gleichen AWS-Region sein. Verwenden Sie zum Identifizieren des KMS-Schlüssels seine Schlüssel-ID oder den Schlüssel-ARN. Sie können keinen anderen Alias verwenden.

Das folgende Beispiel erstellt den `example-key`-Alias und ordnet ihn dem angegebenen KMS-Schlüssel zu. In diesen Beispielen wird die AWS Command Line Interface (AWS CLI) verwendet. Beispiele in verschiedenen Programmiersprachen finden Sie unter Verwendung `CreateAlias` mit einem AWS SDK oder CLI.

```
$ aws kms create-alias \
    --alias-name alias/example-key \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

`CreateAlias` gibt keine Ausgabe zurück. Um den neuen Alias anzuzeigen, verwenden Sie die `ListAliases`-Operation. Details hierzu finden Sie unter Verwenden der API AWS KMS.

Suchen Sie den Aliasnamen und den Alias-ARN für einen KMS-Schlüssel

Aliase erleichtern das Erkennen von KMS-Schlüsseln in der AWS KMS Konsole. Sie können die Aliase für einen KMS-Schlüssel in der AWS KMS Konsole oder mithilfe des ListAliasesVorgangs anzeigen. Der DescribeKeyVorgang, der die Eigenschaften eines KMS-Schlüssels zurückgibt, beinhaltet keine Aliase.

Die folgenden Verfahren zeigen, wie Sie die mit einem KMS-Schlüssel verknüpften Aliase mithilfe der AWS KMS Konsole und der API anzeigen und AWS KMS identifizieren können. In den AWS KMS API-Beispielen wird das AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jede unterstützte Programmiersprache verwenden.

Verwenden der AWS KMS Konsole

In der AWS KMS Konsole werden die Aliase angezeigt, die dem KMS-Schlüssel zugeordnet sind.

1. Öffnen Sie die AWS KMS Konsole unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Zum Anzeigen der Schlüssel in Ihrem Konto, die Sie erstellen und verwalten, wählen Sie im Navigationsbereich Customer managed keys (Vom Kunden verwaltete Schlüssel) aus. Um die

Schlüssel in Ihrem Konto anzuzeigen, das für Sie AWS erstellt und verwaltet wird, wählen AWS Sie im Navigationsbereich die Option Verwaltete Schlüssel aus.

4. Die Spalte Aliases (Aliase) zeigt den Alias für jeden KMS-Schlüssel an. Wenn ein KMS-Schlüssel keinen Alias hat, wird ein Bindestrich (-) in der Spalte Aliases (Aliase) angezeigt.

   Wenn ein KMS-Schlüssel über mehrere Aliase verfügt, enthält die Aliase-Spalte auch eine Aliasübersicht, z. B. (+n mehr). Der folgende KMS-Schlüssel hat beispielsweise zwei Aliase, von denen einer ist `key-test`.

   Um den Aliasnamen und den Alias-ARN aller Aliase für den KMS-Schlüssel zu finden, verwenden Sie die Aliase-Registerkarte.

   - Um direkt zur Registerkarte Aliases (Aliase) zu gelangen, wählen Sie in der Spalte Aliases (Aliase) die Aliasübersicht aus (+n mehr). Eine Aliasübersicht wird nur angezeigt, wenn der KMS-Schlüssel mehr als einen Alias hat.
   - Oder wählen Sie den Alias oder die Schlüssel-ID des KMS-Schlüssels aus (wodurch die Detailseite für den KMS-Schlüssel geöffnet wird) und wählen Sie dann die Aliase-Registerkarte. Die Registerkarten werden unter dem Abschnitt General Configuration (allgemeine Konfiguration) angezeigt.

5. Die Aliase-Registerkarte zeigt den Aliasnamen und Alias-ARN aller Aliase für einen KMS-Schlüssel an. Auf dieser Registerkarte können Sie auch Aliase für den KMS-Schlüssel erstellen und löschen.

Von AWS verwaltete Schlüssel

Sie können den Alias verwenden, um einen zu erkennen Von AWS verwalteter Schlüssel, wie auf dieser Von AWS verwaltete SchlüsselBeispielseite gezeigt. Die Aliasse für Von AWS verwaltete Schlüssel haben immer das Format: `aws/<service-name>`. Der Alias Von AWS verwalteter Schlüssel für Amazon DynamoDB lautet beispielsweise. `aws/dynamodb`

Verwenden der API AWS KMS

Der ListAliasesVorgang gibt den Aliasnamen und den Alias-ARN der Aliase im Konto und in der Region zurück. Die Ausgabe umfasst Aliase für Von AWS verwaltete Schlüssel und für vom Kunden verwaltete Schlüssel. Die Aliasse für Von AWS verwaltete Schlüssel haben immer das Format `aws/<service-name>`, z. B. `aws/dynamodb`.

Die Antwort kann auch Aliase ohne das Feld `TargetKeyId` enthalten. Dabei handelt es sich um vordefinierte Aliase, die AWS zwar erstellt, aber noch keinem KMS-Schlüssel zugeordnet wurden.

```
$ aws kms list-aliases
{
    "Aliases": [
        {
            "AliasName": "alias/access-key",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1516435200.399,
            "LastUpdatedDate": 1516435200.399
        },
        {
            "AliasName": "alias/ECC-P521-Sign",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ECC-P521-Sign",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1693622000.704,
            "LastUpdatedDate": 1693622000.704
        },
        {
            "AliasName": "alias/ImportedKey",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ImportedKey",
            "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
            "CreationDate": 1493622000.704,
            "LastUpdatedDate": 1521097200.235
        },
        {
            "AliasName": "alias/finance-project",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/finance-project",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1604958290.014,
            "LastUpdatedDate": 1604958290.014
        },
        {
            "AliasName": "alias/aws/dynamodb",
```

```
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
            "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef",
            "CreationDate": 1521097200.454,
            "LastUpdatedDate": 1521097200.454
        },
        {
            "AliasName": "alias/aws/ebs",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
            "TargetKeyId": "abcd1234-09fe-ef90-09fe-ab0987654321",
            "CreationDate": 1466518990.200,
            "LastUpdatedDate": 1466518990.200
        }
    ]
}
```

Wenn Sie nur die Aliasse auflisten möchten, die einem bestimmten KMS-Schlüssel zugeordnet sind, verwenden Sie den optionalen `KeyId`-Parameter der `ListAliases`-Operation. Der `KeyId`-Parameter übernimmt die Schlüssel-ID oder den Schüssel-ARN des KMS-Schlüssels.

In diesem Beispiel werden alle Aliase abgerufen, die dem `0987dcba-09fe-87dc-65ba-ab0987654321`KMS-Schlüssel zugeordnet sind

```
$ aws kms list-aliases --key-id 0987dcba-09fe-87dc-65ba-ab0987654321
{
    "Aliases": [
        {
            "AliasName": "alias/access-key",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": "2018-01-20T15:23:10.194000-07:00",
            "LastUpdatedDate": "2018-01-20T15:23:10.194000-07:00"
        },
        {
            "AliasName": "alias/finance-project",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/finance-project",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
            "CreationDate": 1604958290.014,
            "LastUpdatedDate": 1604958290.014
        }
    ]
}
```

Der `KeyId`-Parameter akzeptiert keine Platzhalterzeichen, aber Sie können die Funktionen Ihrer Programmiersprache verwenden, um die Antwort zu filtern.

Mit dem folgenden AWS CLI Befehl werden beispielsweise nur die Aliase für abgerufen. Von AWS verwaltete Schlüssel

```
$ aws kms list-aliases --query 'Aliases[?starts_with(AliasName, `alias/aws/`)]'
```

Der folgende Befehl ruft nur den `access-key`-Alias ab. Bei dem Aliasnamen wird zwischen Groß- und Kleinschreibung unterschieden.

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/access-key`]'
[
    {
        "AliasName": "alias/access-key",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
        "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "CreationDate": "2018-01-20T15:23:10.194000-07:00",
        "LastUpdatedDate": "2018-01-20T15:23:10.194000-07:00"
    }
]
```

Aliase aktualisieren

Da es sich bei einem Alias um eine unabhängige Ressource handelt, können Sie den KMS-Schlüssel ändern, der einem Alias zugeordnet ist. Wenn der `test-key` Alias beispielsweise einem KMS-Schlüssel zugeordnet ist, können Sie ihn mithilfe des UpdateAliasVorgangs einem anderen KMS-Schlüssel zuordnen. Dies ist eine von mehreren Möglichkeiten, einen KMS-Schlüssel manuell zu drehen, ohne das Schlüsselmaterial zu ändern. Sie können auch einen KMS-Schlüssel aktualisieren, sodass eine Anwendung, die einen bestimmten KMS-Schlüssel für neue Ressourcen verwendet hat, jetzt einen anderen KMS-Schlüssel verwendet.

Sie können einen Alias in der AWS KMS Konsole nicht aktualisieren. Sie können außerdem nicht `UpdateAlias` (oder eine andere Operation) verwenden, einen Aliasnamen zu ändern. Wenn Sie einen Aliasnamen ändern möchten, löschen Sie den aktuellen Alias und erstellen Sie einen neuen Alias für den KMS-Schlüssel.

Wenn Sie einen Alias aktualisieren, müssen der aktuelle KMS-Schlüssel und der neue KMS-Schlüssel vom selben Typ sein (beide symmetrisch oder asymmetrisch oder HMAC). Sie müssen

auch die gleiche Schlüsselverwendung haben (`ENCRYPT_DECRYPT` oder `SIGN_VERIFY` oder GENERATE_VERIFY_MAC). Diese Beschränkung verhindert kryptografische Fehler in Code, der Aliasse verwendet.

Im folgenden Beispiel wird zunächst anhand des ListAliasesVorgangs gezeigt, dass der `test-key` Alias derzeit dem KMS-Schlüssel zugeordnet ist`1234abcd-12ab-34cd-56ef-1234567890ab`.

```
$ aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
    "Aliases": [
        {
            "AliasName": "alias/test-key",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "CreationDate": 1593622000.191,
            "LastUpdatedDate": 1593622000.191
        }
    ]
}
```

Als Nächstes ändert die `UpdateAlias`-Operation den KMS-Schlüssel, der dem `test-key`-Alias zugeordnet ist, in den KMS-Schlüssel `0987dcba-09fe-87dc-65ba-ab0987654321`. Sie müssen nicht den aktuell zugeordneten KMS-Schlüssel angeben, sondern nur den neuen („Ziel“)-KMS-Schlüssel. Bei dem Aliasnamen wird zwischen Groß-/Kleinschreibung unterschieden.

```
$ aws kms update-alias --alias-name 'alias/test-key' --target-key-id
 0987dcba-09fe-87dc-65ba-ab0987654321
```

Um zu überprüfen, ob der Alias jetzt dem Ziel-KMS-Schlüssel zugeordnet ist, verwenden Sie die `ListAliases`-Operation erneut. Dieser AWS CLI Befehl verwendet den `--query` Parameter, um nur den `test-key` Alias abzurufen. Die `TargetKeyId`- und `LastUpdatedDate`-Felder werden aktualisiert.

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/test-key`]'
[
    {
        "AliasName": "alias/test-key",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
        "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "CreationDate": 1593622000.191,
```

```
        "LastUpdatedDate": 1604958290.154
    }
]
```

Löscht einen Alias

Sie können einen Alias in der AWS KMS Konsole oder mithilfe des DeleteAliasVorgangs löschen. Bevor Sie einen Alias löschen, stellen Sie sicher, dass er nicht verwendet wird. Obwohl das Löschen eines Aliasses keinen Einfluss auf den zugeordneten KMS-Schlüssel hat, kann es zu Problemen für jede Anwendung führen, die den Alias verwendet. Wenn Sie einen Alias versehentlich löschen, können Sie einen neuen Alias mit demselben Namen erstellen und diesen mit demselben oder einem anderen KMS-Schlüssel verknüpfen.

Wenn Sie einen KMS-Schlüssel löschen, werden alle Aliasse gelöscht, die diesem KMS-Schlüssel zugeordnet sind.

Mithilfe der AWS KMS Konsole

Um einen Alias in der AWS KMS Konsole zu löschen, verwenden Sie die Registerkarte Aliase auf der Detailseite für den KMS-Schlüssel. Sie können mehrere Aliase für einen KMS-Schlüssel gleichzeitig löschen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. Sie können keine Aliase für oder verwalten. Von AWS verwaltete Schlüssel AWS-eigene Schlüssel
4. Wählen Sie in der Tabelle die Schlüssel-ID oder den Alias des KMS-Schlüssels. Wählen Sie dann auf der Detailseite für den KMS-Schlüssel die Aliasse-Registerkarte.

   Wenn ein KMS-Schlüssel über mehrere Aliasse verfügt, zeigt die Aliasse-Spalte in der Tabelle einen Alias und eine Aliasübersicht an, z. B. (+n mehr). Wenn Sie die Aliasübersicht auswählen, gelangen Sie direkt zur Aliases (Aliasse)-Registerkarte auf der Seite mit den KMS-Schlüsseldetails.
5. Aktivieren Sie auf der Aliasse-Registerkarte das Kontrollkästchen neben den Aliassen, die Sie löschen möchten. Wählen Sie dann Löschen aus.

Verwenden der API AWS KMS

Verwenden Sie den DeleteAliasVorgang, um einen Alias zu löschen. Diese Operation löscht jeweils einen Alias. Beim Aliasnamen wird zwischen Groß-/Kleinschreibung unterschieden und der Aliasname muss mit dem `alias/`-Präfix beginnen.

Der folgenden Befehl löscht beispielsweise den `test-key`-Alias. Dieser Befehl gibt keine Ausgabe zurück.

```
$ aws kms delete-alias --alias-name alias/test-key
```

Verwenden Sie den ListAliasesVorgang, um zu überprüfen, ob der Alias gelöscht wurde. Der folgende Befehl verwendet den `--query` Parameter in AWS CLI , um nur den `test-key` Alias abzurufen. Die leeren Klammern in der Antwort weisen darauf hin, dass die `ListAliases`-Antwort keinen `test-key`-Alias enthält. Um die Klammern zu entfernen, verwenden Sie den `--output text`-Parameter und -Wert.

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/test-key`]'
[]
```

Verwenden Sie Aliase, um den Zugriff auf KMS-Schlüssel zu steuern

Verwenden Sie diesen Bedingungsschlüssel, um den Zugriff auf einen KMS-Schlüssel anhand der Aliassen, die dem KMS-Schlüssel zugeordnet sind, zu steuern. Verwenden Sie dazu die ResourceAliases Bedingungsschlüssel kms: RequestAlias und kms:. Diese Funktion ist Teil der AWS KMS Unterstützung für die attributebasierte Zugriffskontrolle (ABAC).

Der `kms:RequestAlias`-Bedingungsschlüssel erlaubt oder verweigert den Zugriff auf einen KMS-Schlüssel basierend auf dem Alias in einer Anforderung. Der `kms:ResourceAliases`-Bedingungsschlüssel erlaubt oder verweigert den Zugriff auf einen KMS-Schlüssel basierend auf dem Alias, der mit dem KMS-Schlüssel verknüpft ist.

Diese Funktionen erlauben es Ihnen nicht, einen KMS-Schlüssel mithilfe eines Aliasses im `resource`-Element einer Richtlinienanweisung zu identifizieren. Wenn ein Alias der Wert eines`resource`-Element ist, gilt die Richtlinie für die Alias-Ressource und nicht für einen KMS-Schlüssel, der mit ihr verknüpft sein könnte.

> **Note**
>
> Es kann bis zu fünf Minuten dauern, bis Tag- und Alias-Änderungen Auswirkungen auf die KMS-Schlüsselautorisierung haben. Letzte Änderungen sind möglicherweise in API-Operationen sichtbar, bevor sie sich auf die Autorisierung auswirken.

Beachten Sie Folgendes, wenn Sie Aliasse verwenden, um den Zugriff auf KMS-Schlüssel zu steuern:

- Verwenden Sie Aliasse, um beim Zugriff die bewährte Methode der geringsten Berechtigung zu befolgen. Geben Sie IAM-Prinzipalen nur die Berechtigungen, die sie für die KMS-Schlüssel benötigen, die sie verwenden oder verwalten müssen. Verwenden Sie beispielsweise Aliase, um die KMS-Schlüssel zu identifizieren, die für ein Projekt verwendet werden. Geben Sie dann dem Projektteam die Berechtigung, nur KMS-Schlüssel mit den Projekt-Aliassen zu verwenden.
- Seien Sie vorsichtig, wenn Sie Prinzipalen die `kms:CreateAlias`-, `kms:UpdateAlias`- oder `kms:DeleteAlias`-Berechtigung erteilen, mit denen sie Aliasse hinzufügen, bearbeiten und löschen können. Wenn Sie Aliasse verwenden, um den Zugriff auf KMS-Schlüssel zu steuern, kann das Ändern eines Aliasses Prinzipalen die Berechtigung zur Verwendung von KMS-Schlüsseln erteilen, die andernfalls nicht über die Berechtigung verfügen. Es kann auch den Zugriff auf KMS-Schlüssel verweigern, die andere Prinzipale für ihre Aufträge benötigen.
- Prüfen Sie die Hauptbenutzer in Ihrer Datenbank AWS-Konto , die derzeit berechtigt sind, Aliase zu verwalten, und passen Sie die Berechtigungen gegebenenfalls an. Schlüsseladministratoren, die nicht über die Berechtigung zum Ändern von Schlüsselrichtlinien oder zum Erstellen von Erteilungen verfügen, können den Zugriff auf KMS-Schlüssel steuern, wenn sie über die Berechtigung zum Verwalten von Aliassen verfügen.

  Zum Beispiel enthält die Standard-Schlüsselrichtlinie für Schlüsseladministratoren der Konsole die `kms:CreateAlias`- `kms:DeleteAlias`- und `kms:UpdateAlias`-Berechtigung. IAM-Richtlinien geben möglicherweise dem Alias Berechtigungen für alle KMS-Schlüssel in Ihrem AWS-Konto. Die AWSKeyManagementServicePowerUserverwaltete Richtlinie ermöglicht es Prinzipalen beispielsweise, Aliase für alle KMS-Schlüssel zu erstellen, zu löschen und aufzulisten, sie jedoch nicht zu aktualisieren.
- Bevor Sie eine Richtlinie festlegen, die von einem Alias abhängt, überprüfen Sie die Aliase auf den KMS-Schlüsseln in Ihrem. AWS-Konto Stellen Sie sicher, dass Ihre Richtlinie nur für die Aliase gilt, die Sie einschließen möchten. Verwenden Sie CloudTrail Protokolle und CloudWatch Alarme, um Sie vor Aliasänderungen zu warnen, die sich auf den Zugriff auf Ihre KMS-Schlüssel auswirken

könnten. Außerdem enthält die ListAliasesAntwort das Erstellungsdatum und das Datum der letzten Aktualisierung für jeden Alias.

- Die Alias-Richtlinienbedingungen verwenden Musterabgleich; sie sind nicht an eine bestimmte Instance eines Aliasses gebunden. Eine Richtlinie, die Alias-basierte Bedingungsschlüssel verwendet, wirkt sich auf alle neuen und vorhandenen Aliasse aus, die dem Muster entsprechen. Wenn Sie einen Alias löschen und neu erstellen, der einer Richtlinienbedingung entspricht, gilt die Bedingung für den neuen Alias, genau wie für den alten Alias.

Der `kms:RequestAlias`-Bedingungsschlüssel basiert auf dem Alias, der explizit in einer Operations-Anforderung angegeben ist. Der `kms:ResourceAliases`-Bedingungsschlüssel hängt von den Aliassen ab, die einem KMS-Schlüssel zugeordnet sind, auch wenn sie nicht in der Anforderung angezeigt werden.

km: RequestAlias

Erlauben oder verweigern Sie den Zugriff auf einen KMS-Schlüssel basierend auf dem Alias, der den KMS-Schlüssel in einer Anforderung identifiziert. Sie können den RequestAlias Bedingungsschlüssel kms: in einer Schlüsselrichtlinie oder IAM-Richtlinie verwenden. Er gilt für Operationen, die einen Alias verwenden, um einen KMS-Schlüssel in einer Anfrage zu identifizieren, nämlich kryptografische Operationen DescribeKey, und. GetPublicKey Es ist nicht gültig für Aliasoperationen wie CreateAliasoder DeleteAlias.

Geben Sie im Bedingungsschlüssel einen Aliasnamen oder ein Aliasnamen-Muster an. Sie können keinen Alias-ARN angeben.

Die folgende Schlüsselrichtlinienanweisung erlaubt es beispielsweise Prinzipalen, die angegebenen Operationen auf dem KMS-Schlüssel zu verwenden. Die Berechtigung ist nur wirksam, wenn die Anforderung einen Alias verwendet, der `alpha` enthält, um den KMS-Schlüssel zu identifizieren.

```
{
  "Sid": "Key policy using a request alias condition",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/alpha-developer"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
```

```
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:RequestAlias": "alias/*alpha*"
    }
  }
}
```

Die folgende Beispielanforderung von einem autorisierten Prinzipal würde die Bedingung erfüllen. Eine Anforderung, die eine Schlüssel-ID, einen Schüssel-ARN oder einen anderen Alias verwendet, würde die Bedingung nicht erfüllen, selbst wenn diese Werte denselben KMS-Schlüssel identifizieren.

```
$ aws kms describe-key --key-id "arn:aws:kms:us-west-2:111122223333:alias/project-
alpha"
```

km: ResourceAliases

Erlauben oder verweigern Sie den Zugriff auf einen KMS-Schlüssel anhand der Aliasse, die dem KMS-Schlüssel zugeordnet sind, selbst wenn der Alias nicht in einer Anforderung verwendet wird. Mit dem ResourceAliases Bedingungsschlüssel kms: können Sie einen Alias oder ein Aliasmuster angeben, z. B.`alias/test*`, sodass Sie es in einer IAM-Richtlinie verwenden können, um den Zugriff auf mehrere KMS-Schlüssel in derselben Region zu steuern. Er ist für jeden AWS KMS Vorgang gültig, der einen KMS-Schlüssel verwendet.

Mit der folgenden IAM-Richtlinie können die Prinzipale beispielsweise die angegebenen Operationen für die KMS-Schlüssel in zwei Schritten aufrufen. AWS-Konten Die Berechtigung gilt jedoch nur für KMS-Schlüssel, die Aliassen zugeordnet sind, die mit `restricted` beginnen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*",
```

```
        "kms:DescribeKey"
      ],
      "Resource": [
        "arn:aws:kms:*:111122223333:key/*",
        "arn:aws:kms:*:444455556666:key/*"
      ],
      "Condition": {
        "ForAnyValue:StringLike": {
          "kms:ResourceAliases": "alias/restricted*"
        }
      }
    }
  ]
}
```

Die `kms:ResourceAliases`-Bedingung ist eine Bedingung der Ressource, nicht die Anforderung. Daher kann eine Anforderung, die den Alias nicht angibt, weiterhin die Bedingung erfüllen.

Die folgende Beispielanforderung, die einen übereinstimmenden Alias angibt, erfüllt die Bedingung.

```
$ aws kms enable-key-rotation --key-id "alias/restricted-project"
```

Die folgende Beispielanforderung erfüllt jedoch auch die Bedingung, vorausgesetzt, dass der angegebene KMS-Schlüssel über einen Alias verfügt, der mit `restricted` beginnt, auch wenn dieser Alias nicht in der Anforderung verwendet wird.

```
$ aws kms enable-key-rotation --key-id "1234abcd-12ab-34cd-56ef-1234567890ab"
```

Erfahren Sie, wie Sie Aliase in Ihren Anwendungen verwenden

Sie können einen Alias verwenden, um einen KMS-Schlüssel im Anwendungscode darzustellen. Der `KeyId` Parameter in AWS KMS kryptografischen Operationen DescribeKey, und GetPublicKeyakzeptiert einen Aliasnamen oder Alias-ARN.

Der folgende `GenerateDataKey`-Befehl verwendet beispielsweise einen Aliasnamen (`alias/finance`), um einen KMS-Schlüssel zu identifizieren. Der Aliasname ist der Wert des `KeyId`-Parameters.

```
$ aws kms generate-data-key --key-id alias/finance --key-spec AES_256
```

Wenn sich der KMS-Schlüssel in einem anderen befindet AWS-Konto, müssen Sie bei diesen Vorgängen einen Schlüssel-ARN oder Alias-ARN verwenden. Beachten Sie bei der Verwendung eines Alias-ARN, dass der Alias für einen KMS-Schlüssel in dem Konto definiert ist, das den KMS-Schlüssel besitzt und sich in jeder Region unterscheiden kann. Hilfestellung beim Suchen des Alias-ARN finden Sie unter Suchen Sie den Aliasnamen und den Alias-ARN für einen KMS-Schlüssel.

Der folgende `GenerateDataKey`-Befehl verwendet beispielsweise einen KMS-Schlüssel, der sich nicht im Konto des Anrufers befindet. Der `ExampleAlias`-Alias ist dem KMS-Schlüssel im angegebenen Konto und in der Region zugeordnet.

```
$ aws kms generate-data-key --key-id arn:aws:kms:us-west-2:444455556666:alias/
ExampleAlias --key-spec AES_256
```

Eine der mächtigsten Verwendungen von Aliasen ist in Anwendungen, die in mehreren AWS-Regionen ausgeführt werden. Beispielsweise könnten Sie über eine globale Anwendung verfügen, die eine RSA asymmetrischen KMS-Schlüssel zur Signatur und Verifizierung verwendet.

- In USA West (Oregon) (us-west-2) sollten Sie `arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab` verwenden.
- In Europa (Frankfurt) (eu-central-1) sollten Sie `arn:aws:kms:eu-central-1:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321` verwenden.
- In Asien-Pazifik (Singapur) (ap-southeast-1) sollten Sie `arn:aws:kms:ap-southeast-1:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d` verwenden.

Sie können in jeder Region eine andere Version Ihrer Anwendung erstellen oder ein Wörterbuch oder eine Switch-Anweisung verwenden, um den richtigen KMS-Schlüssel für jede Region auszuwählen. Es ist jedoch viel einfacher, in jeder Region einen Alias mit demselben Aliasnamen zu erstellen. Bei dem Aliasnamen wird zwischen Groß-/Kleinschreibung unterschieden.

```
aws --region us-west-2 kms create-alias \
    --alias-name alias/new-app \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab

aws --region eu-central-1 kms create-alias \
    --alias-name alias/new-app \
    --key-id arn:aws:kms:eu-central-1:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321
```

```
aws --region ap-southeast-1 kms create-alias \
    --alias-name alias/new-app \
    --key-id arn:aws:kms:ap-
southeast-1:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d
```

Verwenden Sie dann den Alias in Ihrem Code. Wenn Ihr Code in jeder Region ausgeführt wird, verweist der Alias auf den zugehörigen KMS-Schlüssel in dieser Region. Beispielsweise ruft dieser Code die Sign-Operation mit einem Aliasnamen auf.

```
aws kms sign --key-id alias/new-app \
    --message $message \
    --message-type RAW \
    --signing-algorithm RSASSA_PSS_SHA_384
```

Es besteht jedoch das Risiko, dass der Alias gelöscht oder aktualisiert wird, um einem anderen KMS-Schlüssel zugeordnet zu werden. In diesem Fall schlagen die Versuche der Anwendung, Signaturen mit dem Aliasnamen zu überprüfen, fehl, und Sie müssen den Alias möglicherweise neu erstellen oder aktualisieren.

Um dieses Risiko zu verringern, sollten Sie den Prinzipalen die Berechtigung geben, die Aliase zu verwalten, die Sie in Ihrer Anwendung verwenden. Details hierzu finden Sie unter Steuern des Zugriffs auf Aliasse.

Es gibt mehrere andere Lösungen für Anwendungen, die Daten in mehreren AWS-Regionen verschlüsseln, einschließlich des AWS Encryption SDK.

Suchen Sie nach Aliasen in Protokollen AWS CloudTrail

Sie können einen Alias verwenden, um einen AWS KMS key in einer AWS KMS API-Operation darzustellen. Wenn Sie dies tun, werden der Alias und der Schlüssel-ARN des KMS-Schlüssels im AWS CloudTrail Protokolleintrag für das Ereignis aufgezeichnet. Der Alias erscheint im `requestParameters`-Feld. Der Schlüssel-ARN erscheint im `resources`-Feld. Dies gilt auch dann, wenn ein AWS Dienst eine Von AWS verwalteter Schlüssel in Ihrem Konto verwendet.

In der folgenden GenerateDataKeyAnfrage wird beispielsweise der `project-key` Alias verwendet, um einen KMS-Schlüssel darzustellen.

```
$ aws kms generate-data-key --key-id alias/project-key --key-spec AES_256
```

Wenn diese Anforderung im CloudTrail Protokoll aufgezeichnet wird, enthält der Protokolleintrag sowohl den Alias als auch den Schlüssel-ARN des tatsächlich verwendeten KMS-Schlüssels.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDE",
        "arn": "arn:aws:iam::111122223333:role/ProjectDev",
        "accountId": "111122223333",
        "accessKeyId": "FFHIJ",
        "userName": "example-dev"
    },
    "eventTime": "2020-06-29T23:36:41Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.205.123.000",
    "userAgent": "aws-cli/1.18.89 Python/3.6.10
 Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 botocore/1.17.12",
    "requestParameters": {
        "keyId": "alias/project-key",
        "keySpec": "AES_256"
    },
    "responseElements": null,
    "requestID": "d93f57f5-d4c5-4bab-8139-5a1f7824a363",
    "eventID": "d63001e2-dbc6-4aae-90cb-e5370aca7125",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


Einzelheiten zur Protokollierung von AWS KMS Vorgängen in CloudTrail Protokollen finden Sie unterAWS KMS API-Aufrufe protokollieren mit AWS CloudTrail.

Schlagworte in AWS KMS

Ein Tag ist ein optionales Metadaten-Label, das Sie einer AWS Ressource zuweisen (oder zuweisen AWS können). Jedes Tag besteht aus einem Tag-Schlüssel und einem Tag-Wert, wobei beide Zeichenfolgen zwischen Groß-/Kleinschreibung unterscheiden. Der Wert kann auch eine leere (Null) Zeichenfolge sein. Jedes Tag auf einer Ressource muss einen anderen Tag-Schlüssel haben, aber Sie können dasselbe Tag mehreren AWS Ressourcen hinzufügen. Eine Ressource kann bis zu 50 Tags besitzen, die von Benutzern erstellt wurden.

Geben Sie keine vertraulichen oder sensiblen Informationen in den Tag-Schlüssel oder Tag-Wert ein. Tags sind für viele zugänglich AWS-Services, auch für die Abrechnung.

In AWS KMS können Sie einem vom Kunden verwalteten Schlüssel bei der Erstellung des KMS-Schlüssels Tags hinzufügen und vorhandene KMS-Schlüssel kennzeichnen oder deren Markierung aufheben, sofern sie nicht gelöscht werden müssen. Sie können Aliase, benutzerdefinierte Schlüsselspeicher oder KMS-Schlüssel in AWS-eigene Schlüssel anderen nicht kennzeichnen. Von AWS verwaltete Schlüssel AWS-Konten Tags sind optional, aber sie können sehr nützlich sein.

Beispielsweise können Sie ein `"Project"="Alpha"`-Tag allen KMS-Schlüsseln und Amazon-S3-Buckets hinzufügen, die Sie für das Alpha-Projekt verwenden.

```
TagKey   = "Project"
TagValue = "Alpha"
```

Allgemeine Informationen zu Tags, einschließlich Format und Syntax, finden Sie unter AWS Ressourcen taggen in der. Allgemeine Amazon Web Services-Referenz

Tags sind für folgende Aktivitäten nützlich:

- Identifizieren und organisieren Sie Ihre AWS Ressourcen. Viele AWS Dienste unterstützen Tagging, sodass Sie Ressourcen aus verschiedenen Diensten dasselbe Tag zuweisen können, um anzuzeigen, dass die Ressourcen miteinander verknüpft sind. Sie können beispielsweise einem KMS-Schlüssel und einem Amazon Elastic Block Store (Amazon EBS) -Volume oder AWS Secrets Manager Secret dasselbe Tag zuweisen. Sie können Tags auch zur Identifizierung von KMS-Schlüsseln für die Automatisierung verwenden.
- Verfolgen Sie Ihre AWS Kosten. Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenverteilungsbericht generiert, in dem die Nutzung und die Kosten nach Stichwörtern

zusammengefasst sind. Sie können diese Funktion verwenden, um die AWS KMS Kosten für ein Projekt, eine Anwendung oder eine Kostenstelle nachzuverfolgen.

Weitere Informationen zur Verwendung von Tags für die Kostenzuordnung finden Sie unter Verwenden von Kostenzuordnungs-Tags im AWS Billing -Benutzerhandbuch. Weitere Informationen über die Regeln, die für die Tag-Schlüssel und Tag-Werte gelten, finden Sie unter Beschränkungen benutzerdefinierter Tags im AWS Billing -Benutzerhandbuch.

- Kontrollieren Sie den Zugriff auf Ihre AWS Ressourcen. Das Zulassen und Verweigern des Zugriffs auf KMS-Schlüssel auf der Grundlage ihrer Tags ist Teil der AWS KMS Unterstützung für die attributebasierte Zugriffskontrolle (ABAC). Hinweise zur Steuerung des Zugriffs auf AWS KMS keys anhand ihrer Tags finden Sie unter. Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern Weitere allgemeine Informationen zur Verwendung von Tags zur Steuerung des Zugriffs auf AWS Ressourcen finden Sie unter Steuern des Zugriffs auf AWS Ressourcen mithilfe von Ressourcen-Tags im IAM-Benutzerhandbuch.

AWS KMS schreibt einen Eintrag in Ihr AWS CloudTrail Protokoll, wenn Sie die ListResourceTagsOperationen TagResourceUntagResource, oder verwenden.

Themen

- Zugriffssteuerung mit Tags
- Fügen Sie einem KMS-Schlüssel Tags hinzu
- Bearbeiten Sie die mit einem KMS-Schlüssel verknüpften Tags
- Mit einem KMS-Schlüssel verknüpfte Tags entfernen
- Mit einem KMS-Schlüssel verknüpfte Tags anzeigen
- Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern

Zugriffssteuerung mit Tags

Principals benötigen Tagging-Berechtigungen, um Tags entweder in der AWS KMS Konsole oder mithilfe der API hinzuzufügen, anzuzeigen und zu löschen. Sie können diese Berechtigungen in Schlüsselrichtlinien bereitstellen. Sie können sie auch in IAM-Richtlinien (einschließlich VPC-Endpunktrichtlinien) bereitstellen, aber nur wenn die Schlüsselrichtlinie es erlaubt. Die AWSKeyManagementServicePowerUserverwaltete Richtlinie ermöglicht es Prinzipalen, Tags für alle KMS-Schlüssel, auf die das Konto zugreifen kann, zu taggen, aufzuheben und aufzulisten.

Sie können diese Berechtigungen auch einschränken, indem Sie AWS globale Bedingungsschlüssel für Tags verwenden. AWS KMS In können diese Bedingungen den Zugriff auf Tagging-Operationen wie TagResourceund UntagResourcesteuern.

> **Note**
>
> Seien Sie vorsichtig, wenn Sie Prinzipalen die Berechtigung zum Verwalten von Tags und Aliasen erteilen. Wenn Sie eine Markierung oder einen Alias ändern, wird dadurch die Berechtigung für den kundenverwalteten Schlüssel erteilt oder verweigert. Details dazu finden Sie unter ABAC für AWS KMS und Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern.

Beispielrichtlinien und weitere Informationen finden Sie unter Zugriffssteuerung anhand von Tag-Schlüsseln im IAM-Benutzerhandbuch.

Berechtigungen zum Erstellen und Verwalten von Tags funktionieren wie folgt.

km: TagResource

Erlaubt es Prinzipalen, Tags hinzuzufügen oder zu bearbeiten. Um Tags beim Erstellen eines KMS-Schlüssels hinzuzufügen, muss der Prinzipal über die Berechtigung in einer IAM-Richtlinie verfügen, die nicht auf bestimmte KMS-Schlüssel beschränkt ist.

km: ListResourceTags

Erlaubt es Prinzipalen, Tags auf KMS-Schlüsseln anzuzeigen.

km: UntagResource

Erlaubt es Prinzipalen, Tags aus KMS-Schlüsseln zu löschen.

Tag-Berechtigungen in Richtlinien

Sie können Markierungs-Berechtigungen in einer Schlüsselrichtlinie oder einer IAM-Richtlinie bereitstellen. Beispielsweise erteilt die folgende Beispiel-Schlüsselrichtlinie ausgewählten Benutzern Markierungs-Berechtigung für den KMS-Schlüssel. Sie erteilt allen Benutzern, die die Beispiel-Administrator- oder Entwicklerrollen übernehmen können, die Berechtigung zum Anzeigen von Tags.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "example-key-policy",
  "Statement": [
    {
      "Sid": "EnableIAMUserPermissions",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "AllowAllTaggingPermissions",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/LeadAdmin",
        "arn:aws:iam::111122223333:user/SupportLead"
      ]},
      "Action": [
          "kms:TagResource",
          "kms:ListResourceTags",
          "kms:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowRolesViewTags",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:role/Administrator",
        "arn:aws:iam::111122223333:role/Developer"
      ]},
      "Action": "kms:ListResourceTags",
      "Resource": "*"
    }
  ]
}
```

Um Prinzipalen Markierungs-Berechtigung für mehrere KMS-Schlüssel zu erteilen, können Sie eine IAM-Richtlinie verwenden. Damit diese Richtlinie wirksam wird, muss die Schlüsselrichtlinie für jeden KMS-Schlüssel dem Konto erlauben, mithilfe von IAM-Richtlinien den Zugriff auf den KMS-Schlüssel zu steuern.

Beispielsweise kann die folgende IAM-Richtlinie den Prinzipalen erlauben, KMS-Schlüssel zu erstellen. Außerdem erlaubt es ihnen, Tags für alle KMS-Schlüssel im angegebenen Konto zu erstellen und zu verwalten. Diese Kombination ermöglicht es den Prinzipalen, den Tags-Parameter des CreateKeyVorgangs zu verwenden, um einem KMS-Schlüssel während der Erstellung Tags hinzuzufügen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKeys",
      "Effect": "Allow",
      "Action": "kms:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyTags",
      "Effect": "Allow",
      "Action": [
        "kms:TagResource",
        "kms:UntagResource",
        "kms:ListResourceTags"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    }
  ]
}
```

Beschränken von Tag-Berechtigungen

Sie können Markierungs-Berechtigungen einschränken, indem Sie Richtlinienbedingungen verwenden. Die folgenden Richtlinienbedingungen können auf die `kms:TagResource`- und `kms:UntagResource`-Berechtigungen angewendet werden. Beispielsweise können Sie mit

der `aws:RequestTag/tag-key`-Bedingung einem Prinzipal erlauben, nur bestimmte Tags hinzuzufügen, oder verhindern, dass ein Prinzipal Tags mit bestimmten Tag-Schlüsseln hinzufügen kann. Alternativ können Sie auch die `kms:KeyOrigin`-Bedingung verwenden, um zu verhindern, dass Prinzipale KMS-Schlüssel mit importiertem Schlüsselmaterial markieren oder entmarkieren.

- aws: RequestTag
- aws:ResourceTag/tag-key (nur IAM-Richtlinien)
- aws: TagKeys
- km: CallerAccount
- km: KeySpec
- km: KeyUsage
- km: KeyOrigin
- km: ViaService

Eine bewährte Methode bei der Verwendung von Tags zum Steuern des Zugriffs auf KMS-Schlüssel ist die Verwendung der Bedingungsschlüssel `aws:RequestTag/tag-key`oder `aws:TagKeys`, um zu bestimmen, welche Tags (oder Tag-Schlüssel) erlaubt sind.

Die folgende IAM-Richtlinie ähnelt beispielsweise der vorherigen. Diese Richtlinie erlaubt den Prinzipalen jedoch das Erstellen von Tags (`TagResource`) und das Löschen von Tags (`UntagResource`) nur für Tags mit einem `Project`-Tag-Schlüssel.

Da `TagResource` `UntagResource` Anfragen mehrere Tags enthalten können, müssen Sie einen Operator `ForAllValues` or `ForAnyValue` set mit der TagKeys Bedingung aws: angeben. Der `ForAnyValue`-Operator erfordert, dass mindestens einer der Tag-Schlüssel in der Anforderung mit einem der Tag-Schlüssel in der Richtlinie übereinstimmen muss. Der `ForAllValues`-Operator erfordert, dass alle der Tag-Schlüssel in der Anforderung mit einem der Tag-Schlüssel in der Richtlinie übereinstimmen müssen. Der `ForAllValues` Operator gibt auch zurück, `true` wenn die Anfrage keine Tags enthält, schlägt jedoch TagResource UntagResource fehl, wenn keine Tags angegeben sind. Ausführliche Informationen zu den Satz-Operatoren finden Sie unter Verwenden mehrerer Schlüssel und Werte im IAM-Benutzerhandbuch.

JSON

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKey",
      "Effect": "Allow",
      "Action": "kms:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyViewAllTags",
      "Effect": "Allow",
      "Action": "kms:ListResourceTags",
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMPolicyManageTags",
      "Effect": "Allow",
      "Action": [
        "kms:TagResource",
        "kms:UntagResource"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
          "ForAllValues:StringEquals": {"aws:TagKeys": "Project"}
      }
    }
  ]
}
```

Fügen Sie einem KMS-Schlüssel Tags hinzu

Mithilfe von Tags können Sie Ihre AWS Ressourcen identifizieren und organisieren. Sie können Tags zu einem vom Kunden verwalteten Schlüssel hinzufügen, wenn Sie den KMS-Schlüssel erstellen, oder Tags zu vorhandenen KMS-Schlüsseln hinzufügen. Sie können keine Tags hinzufügen Von AWS verwaltete Schlüssel.

Die folgenden Verfahren zeigen, wie Sie mithilfe der AWS KMS Konsole und der AWS KMS API Tags zu vom Kunden verwalteten Schlüsseln hinzufügen. In den AWS KMS API-Beispielen wird die AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jede unterstützte Programmiersprache verwenden.

Themen

- Hinzufügen von Tags beim Erstellen eines KMS-Schlüssels
- Fügen Sie Tags zu vorhandenen KMS-Schlüsseln hinzu

Hinzufügen von Tags beim Erstellen eines KMS-Schlüssels

Sie können einem KMS-Schlüssel Tags hinzufügen, während Sie den Schlüssel mithilfe der AWS KMS Konsole oder des CreateKeyVorgangs erstellen. Um beim Erstellen eines KMS-Schlüssels Tags hinzuzufügen, benötigen Sie `kms:TagResource` zusätzlich zu den zum Erstellen von KMS-Schlüsseln erforderlichen Berechtigungen auch die entsprechenden Berechtigungen in einer IAM-Richtlinie. Die Berechtigung muss mindestens alle KMS-Schlüssel im Konto und in der Region abdecken. Details hierzu finden Sie unter Zugriffssteuerung mit Tags.

Verwenden der Konsole AWS KMS

Um beim Erstellen eines KMS-Schlüssels in der Konsole Tags hinzuzufügen, benötigen Sie zusätzlich zu den Berechtigungen, die zum Kennzeichnen und Erstellen von KMS-Schlüsseln erforderlich sind, über die erforderlichen Berechtigungen zum Anzeigen von KMS-Schlüsseln in der Konsole. Die Berechtigung muss mindestens alle KMS-Schlüssel im Konto und in der Region abdecken.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. (Die Tags von Von AWS verwalteter Schlüssel können Sie nicht verwalten.)
4. Wählen Sie den Schlüsseltyp aus, und wählen Sie dann Next (Weiter).
5. Geben Sie einen Alias und eine optionale Beschreibung ein.
6. Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wenn Sie zusätzliche Tags hinzufügen möchten, wählen Sie Add new tag (neues Tag hinzufügen) aus. Zum Entfernen eines Tags wählen Sie Remove (Entfernen). Wählen Sie nach dem Markieren des neuen KMS-Schlüssels Next (Weiter).
7. Schließen Sie das Erstellen des KMS-Schlüssels ab.

Verwenden der API AWS KMS

Um beim Erstellen von Schlüsseln mithilfe der CreateKeyOperation Tags anzugeben, verwenden Sie den `Tags` Parameter der Operation.

Der Wert des `Tags`-Parameters von `CreateKey` ist eine Sammlung von Tag-Schlüssel- und Tag-Wert-Paaren, unter Beachtung der Groß-/Kleinschreibung. Jedes Tag für einen KMS-Schlüssel muss über einen anderen Tag-Namen verfügen. Der Tag-Wert kann auch eine leere (Null) Zeichenfolge sein.

Mit dem folgenden AWS CLI Befehl wird beispielsweise ein KMS-Schlüssel mit symmetrischer Verschlüsselung und einem `Project:Alpha` Tag erstellt. Wenn Sie mehr als ein Schlüssel-Wert-Paar angeben, verwenden Sie ein Leerzeichen, um die einzelnen Paare zu trennen.

```
$ aws kms create-key --tags TagKey=Project,TagValue=Alpha
```

Wenn dieser Befehl erfolgreich ist, gibt er ein `KeyMetadata`-Objekt mit Informationen über den neuen KMS-Schlüssel zurück. Die `KeyMetadata` enthalten jedoch keine Tags. Verwenden Sie den ListResourceTagsVorgang, um die Tags abzurufen.

Fügen Sie Tags zu vorhandenen KMS-Schlüsseln hinzu

Sie können Ihren vorhandenen, vom Kunden verwalteten KMS-Schlüsseln in der AWS KMS Konsole oder mithilfe des TagResourceVorgangs Tags hinzufügen. Um Tags hinzuzufügen, benötigen Sie eine Tagging-Berechtigung für den KMS-Schlüssel. Sie können diese Berechtigung von der Schlüsselrichtlinie für den KMS-Schlüssel oder, wenn die Schlüsselrichtlinie dies zulässt, von einer IAM-Richtlinie abrufen, die den KMS-Schlüssel enthält.

Verwenden der Konsole AWS KMS

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. (Die Tags von einem Von AWS verwalteter Schlüssel können Sie nicht verwalten.)
4. Sie können den Tabellenfilter verwenden, um nur KMS-Schlüssel mit bestimmten Tags anzuzeigen. Einzelheiten finden Sie unter Tags mithilfe der Konsole anzeigen. AWS KMS

5. Wählen Sie das Kontrollkästchen neben dem Alias für den KMS-Schlüssel.
6. Wählen Sie Schlüsselaktionen, Hinzufügen und Bearbeiten von Tags.
7. Wählen Sie auf der Detailseite für den KMS-Schlüssel die Tags-Registerkarte.
   - Um Ihr erstes Tag zu erstellen, wählen Sie Create tag (Tag erstellen) aus, geben Sie einen Tag-Schlüssel (erforderlich) und einen Tag-Wert (optional) ein und wählen Sie dann Save (Speichern).

     Wenn Sie den Tag-Wert leer lassen, ist der tatsächliche Tag-Wert eine Null oder eine leere Zeichenfolge.
   - Um ein weiteres Tag hinzuzufügen, wählen Sie Edit (Bearbeiten), wählen Sie dann Add tag (Tag hinzufügen), geben Sie einen Tag-Namen und einen Tag-Wert ein und wählen Sie dann Save (Speichern).
8. Klicken Sie auf Save changes (Änderungen speichern), um die Änderungen zu speichern.

Die AWS KMS API verwenden

Der TagResourceVorgang fügt einem KMS-Schlüssel ein oder mehrere Tags hinzu. Sie können diesen Vorgang nicht verwenden, um Tags in einem anderen hinzuzufügen AWS-Konto. Sie können den TagResource Vorgang auch verwenden, um vorhandene Tags zu bearbeiten. Weitere Informationen finden Sie unter the section called “Bearbeiten von Tags”.

Geben Sie zum Hinzufügen eines Tags einen neuen Tag-Schlüssel und einen Tag-Wert an. Jedes Tag für einen KMS-Schlüssel muss über einen anderen Tag-Schlüssel verfügen. Der Tag-Wert kann auch eine leere (Null) Zeichenfolge sein.

Mit dem folgenden Befehl werden z. B. die **`Purpose`**- und **`Department`**-Tags zu einem Beispiel-KMS-Schlüssel hinzugefügt.

```
$ aws kms tag-resource \
        --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
        --tags TagKey=Purpose,TagValue=Pretest TagKey=Department,TagValue=Finance
```

Wenn der Befehl erfolgreich war, erfolgt keine Ausgabe. Verwenden Sie den ListResourceTagsVorgang, um die Tags auf einem KMS-Schlüssel anzuzeigen.

Bearbeiten Sie die mit einem KMS-Schlüssel verknüpften Tags

Mithilfe von Tags können Sie Ihre AWS Ressourcen identifizieren und organisieren. Sie können die mit Ihren vom Kunden verwalteten KMS-Schlüsseln verknüpften Tags in der AWS KMS Konsole oder mithilfe des TagResourceVorgangs bearbeiten. Sie können die Tags eines nicht bearbeiten Von AWS verwalteter Schlüssel.

Die folgenden Verfahren zeigen, wie Sie die mit einem KMS-Schlüssel verknüpften Tags bearbeiten. In den AWS KMS API-Beispielen wird das AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jede unterstützte Programmiersprache verwenden.

Verwenden der AWS KMS Konsole

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. (Sie können die Tags eines nicht bearbeiten) Von AWS verwalteter Schlüssel
4. Sie können den Tabellenfilter verwenden, um nur KMS-Schlüssel mit bestimmten Tags anzuzeigen. Einzelheiten finden Sie unter Tags mithilfe der AWS KMS Konsole anzeigen.
5. Wählen Sie das Kontrollkästchen neben dem Alias für den KMS-Schlüssel.
6. Wählen Sie Schlüsselaktionen, Hinzufügen und Bearbeiten von Tags.
7. Wählen Sie auf der Detailseite für den KMS-Schlüssel die Tags-Registerkarte.
   - Um den Namen oder den Wert eines Tags zu ändern, wählen Sie Edit (Bearbeiten) geben einen Tagnamen und einen Tagwert ein und wählen dann Save (Speichern).
8. Klicken Sie auf Save changes (Änderungen speichern), um die Änderungen zu speichern.

Die AWS KMS API verwenden

Der TagResourceVorgang fügt einem vom Kunden verwalteten Schlüssel ein oder mehrere Tags hinzu;. Sie können es jedoch auch verwenden, TagResourceum den Tag-Wert eines vorhandenen Tags zu ändern. Sie können diese Operation nicht verwenden, um Tags in einem anderen AWS-Konto hinzuzufügen oder zu bearbeiten.

Um ein Tag zu bearbeiten, geben Sie einen vorhandenen Tag-Schlüssel und einen neuen Tag-Wert an. Jedes Tag für einen KMS-Schlüssel muss über einen anderen Tag-Schlüssel verfügen. Der Tag-Wert kann auch eine leere (Null) Zeichenfolge sein.

Beispielsweise ändert dieser Befehl den Wert des `Purpose`-Tag von `Pretest` auf `Test`.

```
$ aws kms tag-resource \
         --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
         --tags TagKey=Purpose,TagValue=Test
```

Mit einem KMS-Schlüssel verknüpfte Tags entfernen

Mithilfe von Tags können Sie Ihre AWS Ressourcen identifizieren und organisieren. Sie können die Tags, die Ihren vom Kunden verwalteten KMS-Schlüsseln zugeordnet sind, in der AWS KMS Konsole oder mithilfe des UntagResourceVorgangs entfernen. Sie können die Tags eines nicht bearbeiten oder entfernen Von AWS verwalteter Schlüssel.

Die folgenden Verfahren zeigen, wie Tags aus einem KMS-Schlüssel entfernt werden. In den AWS KMS API-Beispielen wird das AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jede unterstützte Programmiersprache verwenden.

Verwenden der AWS KMS Konsole

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. (Die Tags von einem Von AWS verwalteter Schlüssel können Sie nicht verwalten.)
4. Sie können den Tabellenfilter verwenden, um nur KMS-Schlüssel mit bestimmten Tags anzuzeigen. Einzelheiten finden Sie unter Tags mithilfe der Konsole anzeigen. AWS KMS
5. Wählen Sie das Kontrollkästchen neben dem Alias für den KMS-Schlüssel.
6. Wählen Sie Schlüsselaktionen, Hinzufügen und Bearbeiten von Tags.
7. Wählen Sie auf der Detailseite für den KMS-Schlüssel die Tags-Registerkarte.

   - Um ein Tag zu löschen, wählen Sie zunächst Edit (Bearbeiten). Wählen Sie dann in der Zeile des Tags die Option Remove (Entfernen) und dann Save (Speichern).

8. Klicken Sie auf Save changes (Änderungen speichern), um die Änderungen zu speichern.

Die AWS KMS API verwenden

Der UntagResourceVorgang löscht Tags aus einem KMS-Schlüssel. Um die zu löschenden Tags zu identifizieren, geben Sie die Tag-Schlüssel an. Sie können diese Operation nicht verwenden, um Tags aus KMS-Schlüsseln in einem anderen AWS-Konto zu löschen.

Wenn sie erfolgreich ist, gibt die `UntagResource`-Operation keine Ausgabe zurück. Wenn der angegebene Tag-Schlüssel nicht auf dem KMS-Schlüssel gefunden wird, wird keine Ausnahme ausgelöst und keine Antwort zurückgegeben. Verwenden Sie den Vorgang, um zu überprüfen, ob der ListResourceTagsVorgang funktioniert hat.

Dieser Befehl löscht beispielsweise das **`Purpose`**-Tag und alle zugehörigen Werte von dem angegebenen KMS-Schlüssel.

```
$ aws kms untag-resource --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --tag-keys
 Purpose
```

Mit einem KMS-Schlüssel verknüpfte Tags anzeigen

Mithilfe von Tags können Sie Ihre AWS Ressourcen identifizieren und organisieren. Sie können die Tags, die Ihren vom Kunden verwalteten KMS-Schlüsseln zugeordnet sind, in der AWS KMS Konsole oder mithilfe des ListResourceTagsVorgangs anzeigen.

Die folgenden Verfahren zeigen, wie Sie die Tags finden, die einem bestimmten KMS-Schlüssel zugeordnet sind. In den AWS KMS API-Beispielen wird das AWS Command Line Interface (AWS CLI) verwendet, Sie können jedoch jede unterstützte Programmiersprache verwenden.

Verwenden der AWS KMS Konsole

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie im Navigationsbereich auf Kundenverwaltete Schlüssel. (Die Tags von einem Von AWS verwalteter Schlüssel können Sie nicht verwalten.)

4. Sie können den Tabellenfilter verwenden, um nur KMS-Schlüssel mit bestimmten Tags anzuzeigen.

   Um nur KMS-Schlüssel mit einem bestimmten Tag anzuzeigen, wählen Sie das Filterfeld aus, wählen Sie den Tag-Schlüssel aus, und wählen Sie dann einen der tatsächlichen Tag-Werte aus. Sie können den Tag-Wert auch ganz oder teilweise eingeben.

   Es werden in der Tabelle alle KMS-Schlüssel mit dem ausgewählten Tag angezeigt. Das Tag wird jedoch nicht angezeigt. Um das Tag anzuzeigen, wählen Sie die Schlüssel-ID oder den Alias des KMS-Schlüssels aus und auf der Detailseite die Tags-Registerkarte. Die Registerkarte wird unter dem Abschnitt General Configuration (allgemeine Konfiguration) angezeigt.

   Dieser Filter erfordert sowohl den Tag-Schlüssel als auch den Tag-Wert. Es wird keine KMS-Schlüssel finden, indem nur der Tag-Schlüssel oder nur dessen Wert eingegeben wird. Um Tags ganz oder teilweise nach dem Tag-Schlüssel oder -Wert zu filtern, verwenden Sie den ListResourceTagsVorgang, um markierte KMS-Schlüssel abzurufen, und verwenden Sie dann die Filterfunktionen Ihrer Programmiersprache.
5. Wählen Sie das Kontrollkästchen neben dem Alias für den KMS-Schlüssel.
6. Wählen Sie Schlüsselaktionen, Hinzufügen und Bearbeiten von Tags.
7. Wählen Sie auf der Detailseite für den KMS-Schlüssel die Tags-Registerkarte.

Verwenden der AWS KMS API

Der ListResourceTagsVorgang ruft die Tags für einen KMS-Schlüssel ab. Der Parameter `KeyId` muss angegeben werden. Sie können diese Operation nicht verwenden, um Tags für KMS-Schlüssel in einem anderen AWS-Konto anzuzeigen.

Der folgende Befehl ruft die Tags für einen Beispiel-KMS-Schlüssel ab.

```
$ aws kms list-resource-tags --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

  "Truncated": false,
  "Tags": [
      {
        "TagKey": "Project",
        "TagValue": "Alpha"
     },
     {
       "TagKey": "Purpose",
```

```
        "TagValue": "Test"
      },
      {
        "TagKey": "Department",
        "TagValue": "Finance"
      }
  ]
}
```

Verwenden Sie Tags, um den Zugriff auf KMS-Schlüssel zu steuern

Sie können den Zugriff auf AWS KMS keys anhand der Tags auf dem KMS-Schlüssel steuern. Sie können beispielsweise eine IAM-Richtlinie schreiben, die es Prinzipalen nur erlaubt, die KMS-Schlüssel mit einem bestimmten Tag zu aktivieren und zu deaktivieren. Oder Sie können eine IAM-Richtlinie verwenden, um zu verhindern, dass Prinzipale KMS-Schlüssel in kryptografischen Operationen verwenden, es sei denn, der KMS-Schlüssel hat ein bestimmtes Tag.

Diese Funktion ist Teil der AWS KMS Unterstützung für die attributebasierte Zugriffskontrolle (ABAC). Informationen zur Verwendung von Tags zur Steuerung des Zugriffs auf AWS Ressourcen finden Sie unter Wozu dient ABAC? AWS und Steuern des Zugriffs auf AWS Ressourcen mithilfe von Ressourcen-Tags im IAM-Benutzerhandbuch. Hilfe zur Behebung von Zugriffsproblemen im Zusammenhang mit ABAC finden Sie unter Problembehandlung bei ABAC für AWS KMS.

> **Note**
>
> Es kann bis zu fünf Minuten dauern, bis Tag- und Alias-Änderungen Auswirkungen auf die KMS-Schlüsselautorisierung haben. Letzte Änderungen sind möglicherweise in API-Operationen sichtbar, bevor sie sich auf die Autorisierung auswirken.

AWS KMS unterstützt den globalen Bedingungskontextschlüssel aws:ResourceTag/tag-key, mit dem Sie den Zugriff auf KMS-Schlüssel anhand der Tags auf dem KMS-Schlüssel steuern können. Da mehrere KMS-Schlüssel das gleiche Tag haben können, können Sie mit diesem Feature die Berechtigung auf einen ausgewählten Satz von KMS-Schlüsseln anwenden. Sie können die KMS-Schlüssel im Satz auch einfach ändern, indem Sie deren Tags ändern.

In AWS KMS, der `aws:ResourceTag/`*`tag-key`* Bedingungsschlüssel wird nur in IAM-Richtlinien unterstützt. Er wird nicht in wichtigen Richtlinien unterstützt, die nur für einen KMS-Schlüssel

gelten, oder bei Vorgängen, die keinen bestimmten KMS-Schlüssel verwenden, wie z. B. die ListAliasesOperationen ListKeysoder.

Die Steuerung des Zugriffs mit Tags bietet eine einfache, skalierbare und flexible Möglichkeit, Berechtigungen zu verwalten. Wenn es jedoch nicht richtig gestaltet und verwaltet wird, kann es versehentlich den Zugriff auf Ihre KMS-Schlüssel zulassen oder verweigern. Wenn Sie Tags verwenden, um den Zugriff zu steuern, sollten Sie die folgenden Methoden berücksichtigen.

- Verwenden Sie Tags, um beim Zugriff die bewährte Methode der geringsten Berechtigung zu befolgen. Geben Sie IAM-Prinzipalen nur die Berechtigungen ein, die sie nur für die KMS-Schlüssel benötigen, die sie verwenden oder verwalten müssen. Verwenden Sie beispielsweise Tags, um die KMS-Schlüssel zu markieren, die für ein Projekt verwendet werden. Geben Sie dann dem Projektteam die Berechtigung, nur KMS-Schlüssel mit dem Projekt-Tag zu verwenden.
- Seien Sie vorsichtig, wenn Sie Prinzipalen die `kms:TagResource`- und `kms:UntagResource`-Berechtigung erteilen, mit denen sie Tags hinzufügen, bearbeiten und löschen können. Wenn Sie Tags verwenden, um den Zugriff auf KMS-Schlüssel zu steuern, kann das Ändern eines Tags Prinzipalen die Berechtigung zur Verwendung von KMS-Schlüsseln erteilen, die andernfalls nicht über die Berechtigung verfügen. Es kann auch den Zugriff auf KMS-Schlüssel verweigern, die andere Prinzipale für ihre Aufträge benötigen. Schlüsseladministratoren, die nicht über die Berechtigung zum Ändern von Schlüsselrichtlinien oder zum Erstellen von Erteilungen verfügen, können den Zugriff auf KMS-Schlüssel steuern, wenn sie über die Berechtigung zum Verwalten von Tags verfügen.

  Verwenden Sie nach Möglichkeit eine Richtlinienbedingung, z. B. `aws:RequestTag/`*`tag-key`* oder `aws:TagKeys` zum Beschränken der Markierungs-Berechtigungen eines Prinzipals auf bestimmte Tags oder Tag-Muster für bestimmte KMS-Schlüssel.
- Überprüfen Sie die Prinzipale in Ihrem System AWS-Konto , die derzeit über Berechtigungen zum Markieren und Entmarkieren verfügen, und passen Sie diese gegebenenfalls an. Zum Beispiel enthält die Standard-Schlüsselrichtlinie für Schlüsseladministratoren der Konsole die `kms:TagResource`- und `kms:UntagResource`-Berechtigung für diesen KMS-Schlüssel. IAM-Richtlinien erlauben vielleicht Markierungs- und Entmarkierungs-Berechtigungen für alle KMS-Schlüssel. Die AWSKeyManagementServicePowerUserverwaltete Richtlinie ermöglicht es Prinzipalen beispielsweise, Tags für alle KMS-Schlüssel zu kennzeichnen, aufzuheben und Tags aufzulisten.
- Bevor Sie eine Richtlinie festlegen, die von einem Tag abhängt, überprüfen Sie die Tags auf den KMS-Schlüsseln in Ihrem. AWS-Konto Stellen Sie sicher, dass Ihre Richtlinie nur für die Tags gilt, die Sie einschließen möchten. Verwenden Sie CloudTrail Protokolle und CloudWatch Alarme, um

Sie auf Tagänderungen aufmerksam zu machen, die sich auf den Zugriff auf Ihre KMS-Schlüssel auswirken könnten.

- Die tag-basierten Richtlinienbedingungen verwenden Musterabgleich; sie sind nicht an eine bestimmte Instance eines Tags gebunden. Eine Richtlinie, die Tag-basierte Bedingungsschlüssel verwendet, wirkt sich auf alle neuen und vorhandenen Tags aus, die dem Muster entsprechen. Wenn Sie ein Tag löschen und neu erstellen, das einer Richtlinienbedingung entspricht, gilt die Bedingung für das neue Tag, genau wie für das alte Tag.

Betrachten Sie beispielsweise die folgende IAM-Richtlinie: Es ermöglicht den Principals, die Operationen GenerateDataKeyWithoutPlaintextund Decrypt nur für KMS-Schlüssel in Ihrem Konto aufzurufen, die sich in der Region Asien-Pazifik (Singapur) befinden und über ein `"Project"="Alpha"` Tag verfügen. Sie können diese Richtlinie an Rollen im Beispiel Alpha-Projekt anfügen.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:ap-southeast-1:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "Alpha"
        }
      }
    }
  ]
}
```

Die folgende IAM-Beispielrichtlinie erlaubt es den Prinzipalen, den KMS-Schlüssel im Konto für kryptografische Operationen zu verwenden. Sie verbietet es den Prinzipalen jedoch, diese

kryptografischen Operationen für KMS-Schlüssel mit einem `"Type"="Reserved"`-Tag oder keinem `"Type"`-Tag zu verwenden.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAllowCryptographicOperations",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMDenyOnTag",
      "Effect": "Deny",
      "Action": [
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Type": "Reserved"
        }
      }
    },
    {
      "Sid": "IAMDenyNoTag",
      "Effect": "Deny",
      "Action": [
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:Decrypt",
        "kms:ReEncrypt*"
```

```
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "Null": {
          "aws:ResourceTag/Type": "true"
        }
      }
    }
  ]
}
```

Wichtige Geschäfte

Ein Schlüsselspeicher ist ein sicherer Ort für die Speicherung und Verwendung kryptografischer Schlüssel. Der Standardschlüsselspeicher in unterstützt AWS KMS auch Methoden zum Generieren und Verwalten der Schlüssel, die er speichert. Standardmäßig wird das kryptografische Schlüsselmaterial für das, in dem Sie es erstellen AWS KMS keys , in Hardware-Sicherheitsmodulen (HSMs) generiert und durch diese geschützt. Dabei AWS KMS handelt es sich um das FIPS 140-3 Cryptographic Module Validation Program. Das Schlüsselmaterial für Ihre KMS-Schlüssel wird niemals unverschlüsselt verlassen. HSMs

AWS KMS unterstützt verschiedene Arten von Schlüsselspeichern zum Schutz Ihres Schlüsselmaterials bei der Erstellung und Verwaltung Ihrer Verschlüsselungsschlüssel. AWS KMS Alle von bereitgestellten Schlüsselspeicheroptionen AWS KMS werden kontinuierlich gemäß FIPS 140-3 auf Sicherheitsstufe 3 validiert und sollen verhindern, dass Dritte, auch AWS Bediener, auf Ihre Klartext-Schlüssel zugreifen oder sie ohne Ihre Zustimmung verwenden.

AWS KMS Standard-Schlüsselspeicher

Standardmäßig wird ein KMS-Schlüssel mithilfe des AWS KMS Standard-HSM erstellt. Dieser HSM-Typ kann als eine Flotte mit mehreren Mandanten betrachtet werden HSMs , die aus Ihrer Sicht den skalierbarsten, kostengünstigsten und am einfachsten zu verwaltenden Schlüsselspeicher ermöglicht. Wenn Sie einen KMS-Schlüssel zur Verwendung innerhalb eines oder mehrerer Schlüssel erstellen, AWS-Services sodass der Service Ihre Daten in Ihrem Namen verschlüsseln kann, erstellen Sie einen symmetrischen Schlüssel. Wenn Sie einen KMS-Schlüssel für Ihr eigenes Anwendungsdesign verwenden, können Sie wählen, ob Sie einen symmetrischen Verschlüsselungsschlüssel, einen asymmetrischen Schlüssel oder einen HMAC-Schlüssel erstellen möchten.

AWS KMS Erstellt in der Standardoption für den Schlüsselspeicher Ihren Schlüssel und verschlüsselt ihn dann unter Schlüsseln, die der Dienst intern verwaltet. Mehrere Kopien verschlüsselter Versionen Ihrer Schlüssel werden dann in Systemen gespeichert, die auf Langlebigkeit ausgelegt sind. Durch die Generierung und den Schutz Ihres Schlüsselmaterials im Standard-Schlüsselspeichertyp können Sie die Skalierbarkeit, Verfügbarkeit und Haltbarkeit von AWS Schlüsselspeichern AWS KMS bei geringstem Betriebsaufwand und geringsten Kosten voll ausschöpfen.

AWS KMS Standard-Schlüsselspeicher mit importiertem Schlüsselmaterial

Anstatt AWS KMS zu verlangen, dass die einzigen Kopien eines bestimmten Schlüssels sowohl generiert als auch gespeichert werden, können Sie Schlüsselmaterial in importieren. So können Sie Ihren eigenen symmetrischen 256-Bit-Verschlüsselungsschlüssel AWS KMS, RSA- oder ECC-Schlüssel (Elliptic Curve) oder HMAC-Schlüssel (Hash-Based Message Authentication Code) generieren und ihn auf eine KMS-Schlüssel-ID (KeyID) anwenden. Dies wird manchmal als Bring Your Own Key (BYOK) bezeichnet. Importiertes Schlüsselmaterial aus Ihrem lokalen Schlüsselverwaltungssystem muss mithilfe eines von ausgegebenen öffentlichen Schlüssels AWS KMS, eines unterstützten kryptografischen Wrapping-Algorithmus und eines zeitbasierten Import-Tokens geschützt werden, das von bereitgestellt wird. AWS KMS Durch diesen Vorgang wird überprüft, ob Ihr verschlüsselter, importierter Schlüssel immer nur dann von einem AWS KMS HSM entschlüsselt werden kann, wenn er Ihre Umgebung verlassen hat.

Importiertes Schlüsselmaterial kann nützlich sein, wenn Sie spezielle Anforderungen an das System haben, das Schlüssel generiert, oder wenn Sie eine Kopie Ihres Schlüssels außerhalb AWS als Backup benötigen. Beachten Sie, dass Sie für die allgemeine Verfügbarkeit und Haltbarkeit eines importierten Schlüsselmaterials verantwortlich sind. AWS KMS Hat zwar eine Kopie Ihres importierten Schlüssels und bleibt hochverfügbar, solange Sie ihn benötigen, aber importierte Schlüssel bieten eine spezielle API zum Löschen — DeleteImportedKeyMaterial. Diese API löscht sofort alle Kopien des importierten Schlüsselmaterials, ohne dass AWS KMS der AWS Schlüssel wiederhergestellt werden kann. Darüber hinaus können Sie für einen importierten Schlüssel eine Ablaufzeit festlegen, nach deren Ablauf der Schlüssel unbrauchbar ist. Damit der Schlüssel wieder nützlich ist AWS KMS, müssen Sie das Schlüsselmaterial erneut importieren und es derselben KeyID zuweisen. Diese Löschaktion für importierte Schlüssel unterscheidet sich von Standardschlüsseln, die in Ihrem Namen für Sie AWS KMS generiert und gespeichert werden. Im Standardfall gibt es beim Löschen von Schlüsseln eine obligatorische Wartezeit, in der ein Schlüssel, der gelöscht werden soll, zunächst für die Nutzung gesperrt wird. Mit dieser Aktion können Sie in den Protokollen aller Anwendungen oder AWS Dienste, die diesen Schlüssel möglicherweise für den Zugriff auf Daten benötigen, Fehler im Zusammenhang mit der Zugriffsverweigerung anzeigen. Wenn Sie solche Zugriffsanfragen sehen, können Sie das geplante Löschen abbrechen und den Schlüssel erneut aktivieren. Erst nach einer konfigurierbaren Wartezeit (zwischen 7 und 30 Tagen) löscht KMS das Schlüsselmaterial, die KeyID und alle mit dem Schlüssel verknüpften Metadaten. Weitere Informationen zu Verfügbarkeit und Haltbarkeit finden Sie im AWS KMS Entwicklerhandbuch unter Schutz von importiertem Schlüsselmaterial.

Beachten Sie bei importiertem Schlüsselmaterial einige zusätzliche Einschränkungen. Da AWS KMS kein neues Schlüsselmaterial generiert werden kann, gibt es keine Möglichkeit, die automatische Rotation importierter Schlüssel zu konfigurieren. Sie müssen einen neuen KMS-Schlüssel mit einer neuen KeyID erstellen und anschließend neues Schlüsselmaterial importieren, um eine effektive Rotation zu erreichen. Außerdem können Chiffretexte, die AWS KMS unter einem importierten symmetrischen Schlüssel erstellt wurden, nicht einfach mit Ihrer lokalen Kopie des Schlüssels außerhalb von entschlüsselt werden. AWS Dies liegt daran, dass das von verwendete authentifizierte Verschlüsselungsformat zusätzliche Metadaten an den Chiffretext AWS KMS anhängt, um während des Entschlüsselungsvorgangs sicherzustellen, dass der Chiffretext mit dem erwarteten KMS-Schlüssel im Rahmen eines früheren Verschlüsselungsvorgangs erstellt wurde. Die meisten externen kryptografischen Systeme verstehen nicht, wie diese Metadaten analysiert werden müssen, um Zugriff auf den rohen Chiffretext zu erhalten und ihre Kopie eines symmetrischen Schlüssels verwenden zu können. Chiffretexte, die unter importierten asymmetrischen Schlüsseln (z. B. RSA oder ECC) erstellt wurden, können außerhalb des entsprechenden (öffentlichen oder privaten) Teils des AWS KMS Schlüssels verwendet werden, da dem Chiffretext keine zusätzlichen Metadaten hinzugefügt werden. AWS KMS

AWS KMS benutzerdefinierte Schlüsselspeicher

Wenn Sie jedoch noch mehr Kontrolle über benötigen HSMs, können Sie einen benutzerdefinierten Schlüsselspeicher erstellen.

Ein benutzerdefinierter Schlüsselspeicher ist ein Schlüsselspeicher innerhalb AWS KMS , der von einem externen Schlüsselmanager unterstützt wird AWS KMS, den Sie besitzen und verwalten. Benutzerdefinierte Schlüsselspeicher kombinieren die praktische und umfassende Schlüsselverwaltungsoberfläche von AWS KMS mit der Möglichkeit, das Schlüsselmaterial und die kryptografischen Operationen zu besitzen und zu kontrollieren. Wenn Sie einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher verwenden, werden die kryptografischen Vorgänge tatsächlich von Ihrem Schlüsselmanager unter Verwendung Ihrer kryptografischen Schlüssel durchgeführt. Dadurch übernehmen Sie mehr Verantwortung für die Verfügbarkeit und Haltbarkeit kryptografischer Schlüssel sowie für den Betrieb der. HSMs

Ihr Eigentum HSMs kann nützlich sein, um bestimmte gesetzliche Anforderungen zu erfüllen, die es noch nicht zulassen, dass mehrinstanzenfähige Webdienste wie der standardmäßige KMS-Schlüsselspeicher Ihre kryptografischen Schlüssel speichern. Benutzerdefinierte Schlüsselspeicher sind nicht sicherer als KMS-Schlüsselspeicher, die AWS-managed verwenden HSMs, aber sie haben andere (und höhere) Verwaltungs- und Kostenauswirkungen. Daher übernehmen Sie mehr

Verantwortung für die Verfügbarkeit und Haltbarkeit kryptografischer Schlüssel und für den Betrieb der. HSMs Unabhängig davon, ob Sie den Standard-Schlüsselspeicher mit AWS KMS HSMs oder einen benutzerdefinierten Schlüsselspeicher verwenden, ist der Dienst so konzipiert, dass niemand, auch keine AWS Mitarbeiter, Ihre Klartext-Schlüssel abrufen oder ohne Ihre Zustimmung verwenden kann. AWS KMS unterstützt zwei Arten von benutzerdefinierten Schlüsselspeichern: AWS CloudHSM Schlüsselspeicher und externe Schlüsselspeicher.

Nicht unterstützte Funktionen

AWS KMS unterstützt die folgenden Funktionen in benutzerdefinierten Schlüsselspeichern nicht.

- Asymmetrische KMS-Schlüssel
- HMAC-KMS-Schlüssel
- KMS-Schlüssel mit importiertem Schlüsselmaterial
- Automatische Schlüsselrotation
- Multiregionale Schlüssel

AWS CloudHSM Schlüsselspeicher

Sie können einen KMS-Schlüssel in einem AWS CloudHSMSchlüsselspeicher erstellen, in dem Root-Benutzerschlüssel in einem AWS CloudHSM Cluster generiert, gespeichert und verwendet werden, den Sie besitzen und verwalten. Anfragen AWS KMS zur Verwendung eines Schlüssels für einen kryptografischen Vorgang werden zur Ausführung des Vorgangs an Ihren AWS CloudHSM Cluster weitergeleitet. Ein AWS CloudHSM Cluster wird zwar von gehostet AWS, es handelt sich jedoch um eine Single-Tenant-Lösung, die direkt von Ihnen verwaltet und betrieben wird. Sie besitzen einen Großteil der Verfügbarkeit und Leistung der KMS-Schlüssel in einem AWS CloudHSM Cluster. Um herauszufinden, ob ein AWS CloudHSM benutzerdefinierter Schlüsselspeicher für Ihre Anforderungen geeignet ist, lesen Sie den Artikel Sind AWS KMS benutzerdefinierte Schlüsselspeicher das Richtige für Sie? im AWS Sicherheits-Blog.

Externer Schlüsselspeicher

Sie können AWS KMS die Verwendung eines externen Schlüsselspeichers (XKS) konfigurieren, in dem Root-Benutzerschlüssel generiert, gespeichert und in einem Schlüsselverwaltungssystem außerhalb von verwendet werden. AWS Cloud Anfragen AWS KMS zur Verwendung eines Schlüssels für einen kryptografischen Vorgang werden zur Ausführung des Vorgangs an Ihr extern gehostetes System weitergeleitet. Insbesondere werden Anfragen an einen XKS-Proxy in Ihrem

Netzwerk weitergeleitet, der die Anfrage dann an das von Ihnen verwendete kryptografische System weiterleitet. Der XKS Proxy ist eine Open-Source-Spezifikation, in die sich jeder integrieren kann. Viele kommerzielle Anbieter von Key-Management-Systemen unterstützen die XKS Proxy-Spezifikation. Da ein externer Schlüsselspeicher von Ihnen oder einem Drittanbieter gehostet wird, besitzen Sie die gesamte Verfügbarkeit, Haltbarkeit und Leistung der Schlüssel im System. Um herauszufinden, ob ein externer Schlüsselspeicher Ihren Anforderungen entspricht, lesen Sie im AWS News-Blog den Artikel Ankündigung eines AWS KMS externen Schlüsselspeichers (XKS).

AWS CloudHSM wichtige Geschäfte

Ein AWS CloudHSM Schlüsselspeicher ist ein benutzerdefinierter Schlüsselspeicher, der von einem AWS CloudHSM Cluster unterstützt wird. Wenn Sie einen AWS KMS key in einem benutzerdefinierten Schlüsselspeicher erstellen, AWS KMS generiert und speichert er nicht extrahierbares Schlüsselmaterial für den KMS-Schlüssel in einem AWS CloudHSM Cluster, den Sie besitzen und verwalten. Wenn Sie einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher verwenden, werden die kryptografischen Operationen HSMs im Cluster ausgeführt. Diese Funktion kombiniert den Komfort und die umfassende Integration von AWS KMS mit der zusätzlichen Steuerung eines AWS CloudHSM Clusters in Ihrem AWS-Konto.

AWS KMS bietet vollständige Konsolen- und API-Unterstützung für die Erstellung, Verwendung und Verwaltung Ihrer benutzerdefinierten Schlüsselspeicher. Verwenden Sie die KMS-Schlüssel in Ihrem benutzerdefinierten Schlüsselspeicher auf die gleiche Weise, wie Sie jeden KMS-Schlüssel verwenden. Sie können KMS-Schlüssel beispielsweise zum Generieren von Datenschlüsseln und zum Verschlüsseln von Daten verwenden. Sie können die KMS-Schlüssel auch in Ihrem benutzerdefinierten Schlüsselspeicher mit AWS Diensten verwenden, die vom Kunden verwaltete Schlüssel unterstützen.

Brauche ich einen benutzerdefinierten Schlüsselspeicher?

Für die meisten Benutzer erfüllt der AWS KMS Standard-Schlüsselspeicher, der durch FIPS 140-3-validierte kryptografische Module geschützt ist, ihre Sicherheitsanforderungen. Es gibt keinen besonderen Grund, eine zusätzliche Ebene von Wartungsverantwortlichkeit und eine Abhängigkeit zu einem weiteren Service zu schaffen.

Es könnte jedoch für Sie sinnvoll sein, einen benutzerdefinierten Schlüsselspeicher zu erstellen, wenn für Sie eine der folgenden Anforderungen gilt:

- Sie haben Schlüssel, die ausdrücklich geschützt werden müssen, in einem Einzelmandanten-HSM oder in einem HSM, über das Sie direkte Kontrolle haben.

- Sie müssen in der Lage sein, Schlüsselmaterial sofort von zu entfernen. AWS KMS
- Sie müssen in der Lage sein, die gesamte Verwendung Ihrer Schlüssel unabhängig von AWS KMS oder zu überprüfen AWS CloudTrail.

Wie funktionieren benutzerdefinierten Schlüsselspeicher?

Jeder benutzerdefinierte Schlüsselspeicher ist einem AWS CloudHSM Cluster in Ihrem zugeordnet AWS-Konto. Wenn Sie den benutzerdefinierten Schlüsselspeicher mit seinem Cluster verbinden, AWS KMS wird die Netzwerkinfrastruktur zur Unterstützung der Verbindung erstellt. Anschließend meldet es sich mit den Anmeldeinformationen eines dedizierten Krypto-Benutzers im Cluster beim AWS CloudHSM Key-Client im Cluster an.

Sie erstellen und verwalten Ihre benutzerdefinierten Schlüsselspeicher AWS KMS und erstellen und verwalten Ihre HSM-Cluster in AWS CloudHSM. Wenn Sie einen AWS KMS benutzerdefinierten Schlüsselspeicher erstellen AWS KMS keys , können Sie die KMS-Schlüssel in AWS KMS anzeigen und verwalten. Sie können jedoch das Schlüsselmaterial der KMS-Schlüssel wie bei anderen Schlüsseln in dem Cluster auch in AWS CloudHSM anzeigen und verwalten.

Sie können KMS-Schlüssel mit symmetrischer Verschlüsselung mit Schlüsselmaterial erstellen, das AWS KMS in Ihrem benutzerdefinierten Schlüsselspeicher generiert wurde. Verwenden Sie dann dieselben Techniken, um die KMS-Schlüssel in Ihrem benutzerdefinierten Schlüsselspeicher anzuzeigen und zu verwalten, die Sie für KMS-Schlüssel im AWS KMS Schlüsselspeicher verwenden. Sie können den Zugriff über IAM und Schlüsselrichtlinien steuern, Tags und Aliasse erstellen, die KMS-Schlüssel aktivieren und deaktivieren und die Löschung von Schlüsseln planen. Sie können die KMS-Schlüssel für kryptografische Operationen verwenden und sie mit AWS Diensten verwenden, die sich in integrieren lassen. AWS KMS

Darüber hinaus haben Sie die volle Kontrolle über den AWS CloudHSM Cluster, einschließlich der Erstellung HSMs und Löschung sowie der Verwaltung von Backups. Sie können den AWS CloudHSM Client und die unterstützten Softwarebibliotheken verwenden, um das Schlüsselmaterial für Ihre KMS-Schlüssel einzusehen, zu prüfen und zu verwalten. Solange der benutzerdefinierte Schlüsselspeicher getrennt ist, AWS KMS kann er nicht darauf zugreifen, und Benutzer können die KMS-Schlüssel im benutzerdefinierten Schlüsselspeicher nicht für kryptografische Operationen verwenden. Diese zusätzliche Kontrollebene macht benutzerdefinierte Schlüsselspeicher zu einer leistungsstarken Lösung für Unternehmen mit entsprechenden Anforderungen.

Wo fange ich an?

Um einen AWS CloudHSM Schlüsselspeicher zu erstellen und zu verwalten, verwenden Sie die Funktionen von AWS KMS und. AWS CloudHSM

1. Fangen Sie an AWS CloudHSM. Erstellen Sie einen aktiven AWS CloudHSM -Cluster oder wählen Sie einen bestehenden aus. Auf dem Cluster müssen mindestens zwei aktive Mitglieder HSMs in unterschiedlichen Availability Zones vorhanden sein. Erstellen Sie dann ein dediziertes Krypto-Benutzerkonto (CU-Konto) für AWS KMS in diesem Cluster.
2. Erstellen Sie unter einen benutzerdefinierten Schlüsselspeicher, der Ihrem ausgewählten AWS CloudHSM Cluster zugeordnet ist. AWS KMS AWS KMS bietet eine vollständige Verwaltungsoberfläche, mit der Sie Ihre benutzerdefinierten Schlüsselspeicher erstellen, anzeigen, bearbeiten und löschen können.
3. Wenn Sie bereit sind, Ihren benutzerdefinierten Schlüsselspeicher zu verwenden, verbinden Sie ihn mit dem zugehörigen AWS CloudHSM Cluster. AWS KMS erstellt die Netzwerkinfrastruktur, die zur Unterstützung der Verbindung benötigt wird. Anschließend meldet sich KMS mit den Anmeldeinformationen des dedizierten Krypto-Benutzers an dem Cluster an, um das Schlüsselmaterial in dem Cluster zu generieren und zu verwalten.
4. Jetzt können Sie KMS-Schlüssel mit symmetrischer Verschlüsselung in Ihrem benutzerdefinierten Schlüsselspeicher erstellen. Sie brauchen dazu lediglich bei der Erstellung eines KMS-Schlüssels Ihren benutzerdefinierten Schlüsselspeicher anzugeben.

Falls Sie bei dem Prozess Hilfe benötigen, schlagen Sie im Thema Fehlerbehebung für einen Custom Key Store nach. Wenn Sie dort keine Antwort oder Lösung finden, Verwenden Sie den Feedback-Link unten auf der Seite in diesem Handbuch, oder posten Sie eine Frage im AWS Key Management Service -Diskussionsforum.

Kontingente

AWS KMS ermöglicht bis zu 10 benutzerdefinierte Schlüsselspeicher in jeder AWS-Konto Region, einschließlich sowohl AWS CloudHSM Schlüsselspeichern als auch externer Schlüsselspeicher, unabhängig von deren Verbindungsstatus. Darüber hinaus gibt es AWS KMS Anforderungskontingente für die Verwendung von KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher.

Preise

Informationen zu den Kosten für AWS KMS benutzerdefinierte Schlüsselspeicher und vom Kunden verwaltete Schlüssel in einem benutzerdefinierten Schlüsselspeicher finden Sie unter AWS Key

Management Service Preise. Informationen zu den Kosten von AWS CloudHSM Clustern und HSMs finden Sie unter AWS CloudHSM Preise.

Regionen

AWS KMS unterstützt AWS CloudHSM wichtige Geschäfte in allen Ländern, in AWS-Regionen denen dies unterstützt AWS KMS wird, mit Ausnahme von Asien-Pazifik (Melbourne), China (Peking), China (Ningxia) und Europa (Spanien).

Nicht unterstützte Funktionen

AWS KMS unterstützt die folgenden Funktionen in benutzerdefinierten Schlüsselspeichern nicht.

- Asymmetrische KMS-Schlüssel
- HMAC-KMS-Schlüssel
- KMS-Schlüssel mit importiertem Schlüsselmaterial
- Automatische Schlüsselrotation
- Multiregionale Schlüssel

AWS CloudHSM Konzepte für Schlüsselspeicher

In diesem Thema werden einige der Begriffe und Konzepte erklärt, die in AWS CloudHSM Schlüsselspeichern verwendet werden.

AWS CloudHSM Schlüsselspeicher

Ein AWS CloudHSM Schlüsselspeicher ist ein benutzerdefinierter Schlüsselspeicher, der einem AWS CloudHSM Cluster zugeordnet ist, den Sie besitzen und verwalten. AWS CloudHSM Cluster werden durch Hardware-Sicherheitsmodule (HSMs) unterstützt, die nach FIPS 140-2 oder FIPS 140-3 Level 3 zertifiziert sind.

Wenn Sie einen KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher erstellen, AWS KMS generiert dieser im zugehörigen Cluster einen persistenten, nicht exportierbaren symmetrischen AES-Schlüssel (Advanced Encryption Standard) mit 256 Bit. AWS CloudHSM Dieses Schlüsselmaterial verlässt Ihren Computer niemals unverschlüsselt. HSMs Wenn Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher verwenden, werden die kryptografischen Operationen HSMs im Cluster ausgeführt.

AWS CloudHSM Schlüsselspeicher kombinieren die praktische und umfassende Schlüsselverwaltungsoberfläche von AWS KMS mit den zusätzlichen Steuerelementen, die ein AWS CloudHSM Cluster in Ihrem AWS-Konto bietet. Mit dieser integrierten Funktion können Sie KMS-Schlüssel erstellen, verwalten und verwenden und AWS KMS gleichzeitig die volle Kontrolle über die Schlüssel behalten HSMs , die ihr Schlüsselmaterial speichern, einschließlich der Verwaltung von Clustern und Backups. HSMs Sie können die AWS KMS Konsole verwenden und APIs den AWS CloudHSM Schlüsselspeicher und die zugehörigen KMS-Schlüssel verwalten. Sie können auch die AWS CloudHSM Konsole APIs, die Clientsoftware und die zugehörigen Softwarebibliotheken verwenden, um den zugehörigen Cluster zu verwalten.

Sie können Ihren AWS CloudHSM Schlüsselspeicher anzeigen und verwalten, seine Eigenschaften bearbeiten und ihn mit dem zugehörigen AWS CloudHSM Cluster verbinden und die Verbindung zu diesem trennen. Wenn Sie einen AWS CloudHSM Schlüsselspeicher löschen müssen, müssen Sie zuerst die KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher löschen, indem Sie deren Löschung planen und warten, bis die Kulanzzeit abgelaufen ist. Durch das Löschen des AWS CloudHSM Schlüsselspeichers wird die Ressource aus dem Cluster entfernt AWS KMS, was sich jedoch nicht auf Ihren AWS CloudHSM Cluster auswirkt.

AWS CloudHSM Cluster

Jeder AWS CloudHSM Schlüsselspeicher ist einem AWS CloudHSM Cluster zugeordnet. Wenn Sie einen AWS KMS key in Ihrem AWS CloudHSM Schlüsselspeicher erstellen, AWS KMS erstellt das zugehörige Schlüsselmaterial im zugehörigen Cluster. Wenn Sie einen KMS-Schlüssel in Ihrem AWS CloudHSM -Schlüsselspeicher verwenden, wird die kryptographische Produktion in dem zugehörigen Cluster ausgeführt.

Jeder AWS CloudHSM Cluster kann nur einem AWS CloudHSM Schlüsselspeicher zugeordnet werden. Der Cluster, den Sie auswählen, kann nicht mit einem anderen AWS CloudHSM Schlüsselspeicher verknüpft werden oder einen gemeinsamen Backup-Verlauf mit einem Cluster teilen, der einem anderen AWS CloudHSM Schlüsselspeicher zugeordnet ist. Der Cluster muss initialisiert und aktiv sein und sich in derselben AWS-Konto Region wie der AWS CloudHSM Schlüsselspeicher befinden. Sie können einen neuen Cluster erstellen oder einen vorhandenen verwenden. AWS KMS benötigt keine ausschließliche Nutzung des Clusters. Um KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher und dem zugehörigen Cluster zu erstellen, muss er mindestens zwei aktive Schlüssel enthalten HSMs. Alle anderen Operationen erfordern nur ein HSM.

Sie geben den AWS CloudHSM Cluster an, wenn Sie den AWS CloudHSM Schlüsselspeicher erstellen, und Sie können ihn nicht ändern. Sie können jedoch eine Cluster als Ersatz verwenden,

der den gleichen Sicherungsverlauf wie der ursprüngliche Cluster hat. So können Sie bei Bedarf den Cluster löschen und ihn durch einen aus einer seiner Sicherungen erstellten Cluster ersetzen. Sie behalten die volle Kontrolle über den zugehörigen AWS CloudHSM Cluster, sodass Sie Benutzer und Schlüssel verwalten, Backups erstellen und löschen HSMs sowie verwenden und verwalten können.

Wenn Sie bereit sind, Ihren AWS CloudHSM Schlüsselspeicher zu verwenden, verbinden Sie ihn mit dem zugehörigen AWS CloudHSM Cluster. Sie können Ihren Custom Key Store jederzeit verbinden oder trennen. Wenn ein benutzerdefinierten Schlüsselspeicher verbunden ist, können Sie seine KMS-Schlüssel erstellen und verwenden. Wenn die Verbindung unterbrochen wird, können Sie den AWS CloudHSM Schlüsselspeicher und die zugehörigen KMS-Schlüssel anzeigen und verwalten. Sie können jedoch keine neuen KMS-Schlüssel erstellen oder die KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher für kryptografische Operationen verwenden.

`kmsuser`-Kryptobenutzer

Um in Ihrem Namen Schlüsselmaterial im zugehörigen AWS CloudHSM Cluster zu erstellen und zu verwalten, AWS KMS verwendet einen dedizierten AWS CloudHSM Crypto-Benutzer (CU) im genannten `kmsuser` Cluster. Das `kmsuser` CU ist ein Standard-CU-Konto, das automatisch mit allen Mitgliedern des Clusters synchronisiert und HSMs in Cluster-Backups gespeichert wird.

Bevor Sie Ihren AWS CloudHSM Schlüsselspeicher erstellen, erstellen Sie mit dem Befehl user create in der CloudHSM-CLI ein `kmsuser` CU-Konto in Ihrem AWS CloudHSM Cluster. Wenn Sie dann den AWS CloudHSM Schlüsselspeicher erstellen, geben Sie das `kmsuser` Kontopasswort für an. AWS KMS Wenn Sie den benutzerdefinierten Schlüsselspeicher verbinden, AWS KMS meldet er sich als `kmsuser` CU beim Cluster an und wechselt das zugehörige Passwort. AWS KMS verschlüsselt Ihr `kmsuser` Passwort, bevor es sicher gespeichert wird. Wenn das Passwort rotiert wird, wird das neue Passwort verschlüsselt und auf die gleiche Weise gespeichert.

AWS KMS bleibt angemeldet, `kmsuser` solange der AWS CloudHSM Schlüsselspeicher verbunden ist. Sie sollten dieses CU-Konto nicht für andere Zwecke verwenden. Sie behalten jedoch die letztendliche Kontrolle des `kmsuser`-CU-Kontos. Sie können jederzeit die Schlüssel finden, die Ihnen `kmsuser` gehören. Bei Bedarf können Sie den benutzerdefinierten Schlüsselspeicher trennen, das `kmsuser`-Passwort ändern, sich bei dem Cluster als `kmsuser` anmelden sowie die Schlüssel, über die `kmsuser` verfügt, anzeigen und verwalten.

Für Anweisungen zum Erstellen Ihres `kmsuser`-CU-Kontos vgl. Erstellen des `kmsuser`-Crypto-Benutzers.

KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher

Sie können die AWS KMS API AWS KMS oder verwenden, um einen AWS KMS keys in einem AWS CloudHSM Schlüsselspeicher zu erstellen. Sie verwenden die gleiche Methode wie bei jedem KMS-Schlüssel. Der einzige Unterschied besteht darin, dass Sie den AWS CloudHSM Schlüsselspeicher identifizieren und angeben müssen, dass der Ursprung des Schlüsselmaterials der AWS CloudHSM Cluster ist.

Wenn Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher erstellen, AWS KMS erstellt er den KMS-Schlüssel in AWS KMS und generiert ein persistentes, nicht exportierbares symmetrisches 256-Bit-Schlüsselmaterial (Advanced Encryption Standard, AES) im zugehörigen Cluster. Wenn Sie den AWS KMS Schlüssel in einem kryptografischen Vorgang verwenden, wird der Vorgang im AWS CloudHSM Cluster unter Verwendung des clusterbasierten AES-Schlüssels ausgeführt. Obwohl symmetrische und asymmetrische Schlüssel verschiedener Typen AWS CloudHSM unterstützt werden, unterstützen AWS CloudHSM Schlüsselspeicher nur symmetrische AES-Verschlüsselungsschlüssel.

Sie können die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher in der AWS KMS Konsole anzeigen und mithilfe der Konsolenoptionen die benutzerdefinierte Schlüsselspeicher-ID anzeigen. Sie können den DescribeKeyVorgang auch verwenden, um die AWS CloudHSM Schlüsselspeicher-ID und die AWS CloudHSM Cluster-ID zu finden.

Die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher funktionieren genauso wie alle KMS-Schlüssel in AWS KMS. Autorisierte Benutzer benötigen die gleichen Berechtigungen zum Verwenden und Verwalten der KMS-Schlüssel. Sie verwenden dieselben Konsolenprozeduren und API-Operationen, um die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher anzuzeigen und zu verwalten. Dazu gehören das Aktivieren und Deaktivieren von KMS-Schlüsseln, das Erstellen und Verwenden von Tags und Aliasen sowie das Festlegen und Ändern von IAM- und Schlüsselrichtlinien. Sie können die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher für kryptografische Operationen verwenden und sie mit integrierten AWS Diensten verwenden, die die Verwendung von kundenverwalteten Schlüsseln unterstützen. Sie können jedoch weder die automatische Schlüsselrotation aktivieren noch Schlüsselmaterial in einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher importieren.

Sie verwenden dasselbe Verfahren auch, um das Löschen eines KMS-Schlüssels in einem AWS CloudHSM Schlüsselspeicher zu planen. AWS KMS Löscht nach Ablauf der Wartezeit den KMS-Schlüssel aus KMS. Anschließend wird nach besten Kräften versucht, das Schlüsselmaterial für den KMS-Schlüssel aus dem zugehörigen AWS CloudHSM Cluster zu löschen. Möglicherweise müssen Sie jedoch das verwaiste Schlüsselmaterial manuell aus dem Cluster und seinen Backups löschen.

Kontrollieren Sie den Zugang zu Ihrem AWS CloudHSM Schlüsselspeicher

Sie verwenden IAM-Richtlinien, um den Zugriff auf Ihren AWS CloudHSM Schlüsselspeicher und Ihren AWS CloudHSM Cluster zu kontrollieren. Sie können Schlüsselrichtlinien, IAM-Richtlinien und Zuweisungen verwenden, um den Zugriff auf den Schlüsselspeicher AWS KMS keys in Ihrem AWS CloudHSM Schlüsselspeicher zu kontrollieren. Sie sollten Benutzern, Gruppen und Rollen ausschließlich die Berechtigungen gewähren, die sie für die Aufgaben benötigen, die sie voraussichtlich ausführen werden.

Um Ihre AWS CloudHSM Schlüsselspeicher zu unterstützen, AWS KMS ist eine Genehmigung zum Abrufen von Informationen über Ihre AWS CloudHSM Cluster erforderlich. Außerdem benötigt es die Erlaubnis, die Netzwerkinfrastruktur zu erstellen, die Ihren AWS CloudHSM Schlüsselspeicher mit seinem AWS CloudHSM Cluster verbindet. Um diese Berechtigungen zu erhalten, AWS KMS erstellt die AWSServiceRoleForKeyManagementServiceCustomKeyStoresdienstbezogene Rolle in Ihrem AWS-Konto. Weitere Informationen finden Sie unter Autorisierung AWS KMS zur Verwaltung AWS CloudHSM und Amazon-Ressourcen EC2 .

Achten Sie beim Entwerfen Ihres AWS CloudHSM Schlüsselspeichers darauf, dass die Prinzipale, die ihn verwenden und verwalten, nur über die erforderlichen Berechtigungen verfügen. In der folgenden Liste werden die Mindestberechtigungen beschrieben, die für Manager und Benutzer von AWS CloudHSM Schlüsselspeichern erforderlich sind.

- Principals, die Ihren AWS CloudHSM Schlüsselspeicher erstellen und verwalten, benötigen die folgenden Berechtigungen, um die AWS CloudHSM Key-Store-API-Operationen verwenden zu können.
  - `cloudhsm:DescribeClusters`
  - `kms:CreateCustomKeyStore`
  - `kms:ConnectCustomKeyStore`
  - `kms:DeleteCustomKeyStore`
  - `kms:DescribeCustomKeyStores`
  - `kms:DisconnectCustomKeyStore`
  - `kms:UpdateCustomKeyStore`
  - `iam:CreateServiceLinkedRole`
- Principals, die den AWS CloudHSM Cluster erstellen und verwalten, der Ihrem AWS CloudHSM Schlüsselspeicher zugeordnet ist, benötigen die Erlaubnis, einen Cluster zu erstellen und zu initialisieren. AWS CloudHSM Dies beinhaltet die Erlaubnis, eine Amazon Virtual Private Cloud

(VPC) zu erstellen oder zu verwenden, Subnetze zu erstellen und eine EC2 Amazon-Instance zu erstellen. Möglicherweise müssen sie auch Backups erstellen HSMs, löschen und verwalten. Eine Liste der erforderlichen Berechtigungen finden Sie unter Identitäts- und Zugriffsverwaltung für AWS CloudHSM im AWS CloudHSM Benutzerhandbuch.

- Prinzipale, die AWS KMS keys in Ihrem AWS CloudHSM Schlüsselspeicher etwas erstellen und verwalten, benötigen dieselben Berechtigungen wie diejenigen, die alle KMS-Schlüssel in AWS KMS erstellen und verwalten. Die Standard-Schlüsselrichtlinie für einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher ist identisch mit der Standard-Schlüsselrichtlinie für KMS-Schlüssel in AWS KMS. Die attributebasierte Zugriffskontrolle (ABAC), die Tags und Aliase verwendet, um den Zugriff auf KMS-Schlüssel zu steuern, ist auch für KMS-Schlüssel in Schlüsselspeichern wirksam. AWS CloudHSM
- Principals, die die KMS-Schlüssel in Ihrem AWS CloudHSM Schlüsselspeicher für kryptografische Operationen verwenden, benötigen eine Genehmigung, um den kryptografischen Vorgang mit dem KMS-Schlüssel auszuführen, z. B. kms:Decrypt. Sie können diese Berechtigungen in einer Schlüsselrichtlinie oder einer IAM-Richtlinie bereitstellen. Sie benötigen jedoch keine zusätzlichen Berechtigungen, um einen KMS-Schlüssel in einem Schlüsselspeicher zu verwenden. AWS CloudHSM

Erstellen Sie einen AWS CloudHSM Schlüsselspeicher

Sie können einen oder mehrere AWS CloudHSM Schlüsselspeicher in Ihrem Konto erstellen. Jeder AWS CloudHSM Schlüsselspeicher ist einem AWS CloudHSM Cluster in derselben AWS-Konto Region zugeordnet. Vor dem Erstellen des AWS CloudHSM -Schlüsselspeichers müssen Sie einige Voraussetzungen erfüllen. Bevor Sie Ihren AWS CloudHSM Schlüsselspeicher verwenden können, müssen Sie ihn dann mit seinem AWS CloudHSM Cluster verbinden.

> **Hinweise**
>
> KMS kann nicht IPv6 mit AWS CloudHSM Schlüsselspeichern kommunizieren.
> Wenn Sie versuchen, einen AWS CloudHSM Schlüsselspeicher mit denselben Eigenschaftswerten wie ein vorhandener AWS CloudHSM Schlüsselspeicher ohne Verbindung zu erstellen, AWS KMS wird kein neuer AWS CloudHSM Schlüsselspeicher erstellt und es wird keine Ausnahme ausgelöst oder ein Fehler angezeigt. AWS KMS Erkennt das Duplikat stattdessen als wahrscheinliche Folge eines erneuten Versuchs und gibt die ID des vorhandenen AWS CloudHSM Schlüsselspeichers zurück.

Sie müssen Ihren AWS CloudHSM Schlüsselspeicher nicht sofort verbinden. Sie können die Verbindung getrennt lassen, bis Sie zur Verwendung bereit sind. Wenn Sie jedoch prüfen möchten, ob die Konfiguration ordnungsgemäß erfolgt ist, können Sie eine Verbindung herstellen, den Verbindungsstatus anzeigen und anschließend die Verbindung wieder trennen.

Themen

- Erfüllen der Voraussetzungen
- Erstellen Sie einen neuen AWS CloudHSM Schlüsselspeicher

Erfüllen der Voraussetzungen

Jeder AWS CloudHSM Schlüsselspeicher wird von einem AWS CloudHSM Cluster unterstützt. Um einen AWS CloudHSM Schlüsselspeicher zu erstellen, müssen Sie einen aktiven AWS CloudHSM Cluster angeben, der noch keinem anderen Schlüsselspeicher zugeordnet ist. Sie müssen außerdem einen dedizierten Crypto-Benutzer (CU) in den Clustern einrichten HSMs , mit dem Sie Schlüssel in Ihrem Namen erstellen und verwalten AWS KMS können.

Bevor Sie einen AWS CloudHSM Schlüsselspeicher erstellen, gehen Sie wie folgt vor:

Wählen Sie einen AWS CloudHSM Cluster

Jeder AWS CloudHSM Schlüsselspeicher ist genau einem AWS CloudHSM Cluster zugeordnet. Wenn Sie AWS KMS keys in Ihrem AWS CloudHSM Schlüsselspeicher erstellen, AWS KMS erstellt die KMS-Schlüsselmetadaten, z. B. eine ID und einen Amazon-Ressourcennamen (ARN) in AWS KMS. Anschließend erstellt es das Schlüsselmaterial im HSMs des zugehörigen Clusters. Sie können einen neuen AWS CloudHSM Cluster erstellen oder einen vorhandenen verwenden. AWS KMS erfordert keinen exklusiven Zugriff auf den Cluster.

Der AWS CloudHSM Cluster, den Sie auswählen, ist dauerhaft mit dem AWS CloudHSM Schlüsselspeicher verknüpft. Nachdem Sie den AWS CloudHSM Schlüsselspeicher erstellt haben, können Sie die Cluster-ID des zugehörigen Clusters ändern. Der von Ihnen angegebene Cluster muss jedoch einen Backup-Verlauf mit dem ursprünglichen Cluster gemeinsam nutzen. Um einen Cluster zu verwenden, der nichts miteinander zu tun hat, müssen Sie einen neuen AWS CloudHSM Schlüsselspeicher erstellen.

Der AWS CloudHSM Cluster, den Sie auswählen, muss die folgenden Eigenschaften aufweisen:

- Der Cluster muss aktiv sein.

  Sie müssen den Cluster erstellen, initialisieren, die AWS CloudHSM Client-Software für Ihre Plattform installieren und dann den Cluster aktivieren. Ausführliche Anweisungen finden Sie im Abschnitt Erste Schritte AWS CloudHSM im AWS CloudHSM -Benutzerhandbuch.

- Mutual TLS (mTLS) ist nicht aktiviert.

  KMS unterstützt mTLS für Cluster nicht. Diese Einstellung darf nicht aktiviert sein.

- Der Cluster muss sich im selben Konto und in derselben Region wie der AWS CloudHSM Schlüsselspeicher befinden. Sie können einen AWS CloudHSM Schlüsselspeicher in einer Region nicht einem Cluster in einer anderen Region zuordnen. Um eine Schlüsselinfrastruktur in mehreren Regionen zu erstellen, müssen Sie in jeder Region AWS CloudHSM Schlüsselspeicher und Cluster erstellen.

- Der Cluster kann nicht mit einem anderen benutzerdefinierten Schlüsselspeicher desselben Kontos und derselben Region verknüpft werden. Jeder AWS CloudHSM Schlüsselspeicher im Konto und in der Region muss einem anderen AWS CloudHSM Cluster zugeordnet sein. Sie können keinen Cluster angeben, der bereits einem benutzerdefinierten Schlüsselspeicher zugeordnet ist, und auch keinen Cluster, der einen gemeinsamen Sicherungsverlauf mit einem zugeordneten Cluster hat. Cluster mit einem gemeinsamen Sicherungsverlauf haben dasselbe Cluster-Zertifikat. Um das Clusterzertifikat eines Clusters anzuzeigen, verwenden Sie die AWS CloudHSM Konsole oder den DescribeClustersVorgang.

  Wenn Sie einen AWS CloudHSM -Cluster in einer anderen Region sichern, wird er als anderer Cluster betrachtet, und Sie können die Sicherung mit einem benutzerdefinierten Schlüsselspeicher in seiner Region verknüpfen. Die KMS-Schlüssel in den beiden benutzerdefinierten Schlüsselspeichern sind jedoch nicht interoperabel, selbst wenn sie denselben Backing-Schlüssel haben. AWS KMS bindet Metadaten an den Chiffretext, sodass dieser nur mit dem KMS-Schlüssel entschlüsselt werden kann, mit dem er verschlüsselt wurde.

- Der Cluster muss mit privaten Subnetzen in mindestens zwei Availability Zones in der Region konfiguriert werden. Da AWS CloudHSM dies nicht in allen Availability Zones unterstützt wird, empfehlen wir, private Subnetze in allen Availability Zones in der Region zu erstellen. Sie können die Subnetze für einen vorhandenen Cluster nicht neu konfigurieren, haben jedoch die Möglichkeit, einen Cluster von einer Sicherung zu erstellen, mit anderen Subnetzen in der Cluster-Konfiguration.

> **Important**
>
> Nachdem Sie Ihren AWS CloudHSM Schlüsselspeicher erstellt haben, löschen Sie keines der für den Cluster konfigurierten privaten Subnetze. AWS CloudHSM Wenn AWS KMS nicht alle Subnetze in der Clusterkonfiguration gefunden werden können, schlagen Versuche, eine Verbindung zum benutzerdefinierten Schlüsselspeicher herzustellen, fehl und es wird ein `SUBNET_NOT_FOUND` Verbindungsfehler angezeigt. Details hierzu finden Sie unter Beheben eines Verbindungsfehlers.

- Die Sicherheitsgruppe für den Cluster (`cloudhsm-cluster-<cluster-id>-sg`) muss Regeln für eingehenden und ausgehenden Datenverkehr enthalten, die TCP-Verkehr über die Ports 2223-2225 IPv4 zulassen. Die Angabe für Source (Quelle) in den eingehenden Regeln und für Destination (Ziel) in den ausgehenden Regeln muss mit der Sicherheitsgruppen-ID übereinstimmen. Diese Regeln werden standardmäßig festgelegt, wenn Sie den Cluster erstellen. Sie dürfen nicht gelöscht oder geändert werden.
- Der Cluster muss mindestens zwei aktive HSMs Mitglieder in unterschiedlichen Availability Zones enthalten. Verwenden Sie die AWS CloudHSM Konsole oder den DescribeClustersVorgang HSMs, um die Anzahl von zu überprüfen. Falls erforderlich, können Sie ein HSM hinzufügen.

Finden des Trust Anchor-Zertifikats

Wenn Sie einen benutzerdefinierten Schlüsselspeicher erstellen, müssen Sie das Trust Anchor-Zertifikat für den AWS CloudHSM Cluster hochladen AWS KMS. AWS KMS benötigt das Trust Anchor-Zertifikat, um den AWS CloudHSM Schlüsselspeicher mit dem zugehörigen AWS CloudHSM Cluster zu verbinden.

Jeder aktive AWS CloudHSM Cluster verfügt über ein Trust Anchor-Zertifikat. Wenn Sie den Cluster initialisieren, generieren Sie dieses Zertifikat, speichern es in der Datei `customerCA.crt` und kopieren es auf Hosts, die eine Verbindung zu dem Cluster herstellen.

Erstellen Sie den **`kmsuser`** Crypto-Benutzer für AWS KMS

Um Ihren AWS CloudHSM Schlüsselspeicher zu verwalten, melden Sie AWS KMS sich beim `kmsuser`Crypto User (CU) -Konto im ausgewählten Cluster an. Bevor Sie Ihren AWS CloudHSM Schlüsselspeicher erstellen, müssen Sie die `kmsuser` CU erstellen. Wenn Sie dann Ihren AWS CloudHSM Schlüsselspeicher erstellen, geben Sie das Passwort für `kmsuser` to ein AWS KMS. Immer wenn Sie den AWS CloudHSM Schlüsselspeicher mit dem zugehörigen AWS CloudHSM

Cluster verbinden, AWS KMS meldet sich das Passwort als an `kmsuser` und wechselt das Passwort `kmsuser`

> **Important**
>
> Geben Sie nicht die Option `2FA` beim Erstellen des `kmsuser`-CU an. Wenn Sie dies tun, AWS KMS können Sie sich nicht anmelden und Ihr AWS CloudHSM Schlüsselspeicher kann nicht mit diesem AWS CloudHSM Cluster verbunden werden. Wenn Sie 2FA angeben, können Sie dies nicht rückgängig machen. Sie müssen dann den CU löschen und neu erstellen.

> **Hinweise**
>
> Die folgenden Verfahren verwenden das AWS CloudHSM Client SDK 5-Befehlszeilentool CloudHSM CLI. Die CloudHSM-CLI ersetzt durch`key-handle`. `key-reference`
> Am 1. Januar 2025 AWS CloudHSM wird der Support für die Client SDK 3-Befehlszeilentools, das CloudHSM Management Utility (CMU) und das Key Management Utility (KMU) eingestellt. Weitere Informationen zu den Unterschieden zwischen den Client SDK 3-Befehlszeilentools und dem Client SDK 5-Befehlszeilentool finden Sie unter Migrieren von Client SDK 3 CMU und KMU zu Client SDK 5 CloudHSM CLI im Benutzerhandbuch.AWS CloudHSM

1. Folgen Sie den Anleitungen für die ersten Schritte, wie sie im Abschnitt Erste Schritte mit CloudHSM Command Line Interface (CLI) im AWS CloudHSM Benutzerhandbuch beschrieben sind.
2. Verwenden Sie den Befehl user create, um eine CU mit dem Namen zu erstellen. `kmsuser`

   Das Passwort muss 7 – 32 alphanumerische Zeichen umfassen. Bei der Angabe wird zwischen Groß- und Kleinschreibung unterschieden, Sonderzeichen sind nicht zulässig.

   Der folgende Beispielbefehl erstellt eine `kmsuser` CU.

   ```
   aws-cloudhsm > user create --username kmsuser --role crypto-user
   Enter password:
   Confirm password:
   {
   ```

```
  "error_code": 0,
  "data": {
    "username": "kmsuser",
    "role": "crypto-user"
  }
}
```

Erstellen Sie einen neuen AWS CloudHSM Schlüsselspeicher

Nachdem Sie die Voraussetzungen zusammengestellt haben, können Sie in der AWS KMS Konsole oder mithilfe des CreateCustomKeyStoreVorgangs einen neuen AWS CloudHSM Schlüsselspeicher erstellen.

Verwenden der Konsole AWS KMS

Wenn Sie in der einen AWS CloudHSM Schlüsselspeicher erstellen AWS Management Console, können Sie die erforderlichen Komponenten als Teil Ihres Workflows hinzufügen und erstellen. Der Prozess ist jedoch schneller, wenn Sie die Voraussetzungen schon vorab erfüllt haben.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (benutzerdefinierte Schlüsselspeicher) AWS CloudHSM -Schlüsselspeicher aus.
4. Wählen Sie Einen Schlüsselspeicher erstellen aus.
5. Geben Sie einen Anzeigenamen für den benutzerdefinierten Schlüsselspeicher ein. Der Name muss unter allen beutzerdefinierten Schlüsselspeichern in Ihrem Konto eindeutig sein.

   > **Important**
   >
   > Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

6. Wählen Sie einen AWS CloudHSM Cluster für den AWS CloudHSM Schlüsselspeicher aus. Oder, um einen neuen AWS CloudHSM Cluster zu erstellen, wählen Sie den Link AWS CloudHSM Cluster erstellen.

Im Menü werden die AWS CloudHSM Cluster in Ihrem Konto und Ihrer Region angezeigt, die noch keinem AWS CloudHSM Schlüsselspeicher zugeordnet sind. Der Cluster muss die Anforderungen für die Zuordnung zu einem benutzerdefinierten Schlüsselspeicher erfüllen.

7. Wählen Sie Datei auswählen und laden Sie dann das Trust Anchor-Zertifikat für den ausgewählten AWS CloudHSM Cluster hoch. Dies ist die Datei `customerCA.crt`, die Sie bei der Initialisierung des Clusters erstellt haben.
8. Geben Sie das Passwort des `kmsuser`-Kryptobenutzers (CU) ein, den Sie in dem ausgewählten Cluster erstellt haben.
9. Wählen Sie Erstellen aus.

Wenn das Verfahren erfolgreich ist, wird der neue AWS CloudHSM Schlüsselspeicher in der Liste der AWS CloudHSM Schlüsselspeicher im Konto und in der Region angezeigt. Bei nicht erfolgreicher Ausführung wird eine Fehlermeldung mit einer Beschreibung des Problems und Hilfestellung zur Fehlerbehebung angezeigt. Wenn Sie weitere Hilfe benötigen, beachten Sie den Abschnitt Fehlerbehebung für einen Custom Key Store.

Wenn Sie versuchen, einen AWS CloudHSM Schlüsselspeicher mit denselben Eigenschaftswerten wie ein vorhandener AWS CloudHSM Schlüsselspeicher ohne Verbindung zu erstellen, AWS KMS wird kein neuer AWS CloudHSM Schlüsselspeicher erstellt und es wird keine Ausnahme ausgelöst oder ein Fehler angezeigt. AWS KMS Erkennt das Duplikat stattdessen als wahrscheinliche Folge eines erneuten Versuchs und gibt die ID des vorhandenen AWS CloudHSM Schlüsselspeichers zurück.

Weiter: Neue AWS CloudHSM Schlüsselspeicher werden nicht automatisch verbunden. Bevor Sie AWS KMS keys im AWS CloudHSM Schlüsselspeicher etwas erstellen können, müssen Sie den benutzerdefinierten Schlüsselspeicher mit dem zugehörigen AWS CloudHSM Cluster verbinden.

Verwenden der AWS KMS API

Sie können den CreateCustomKeyStoreVorgang verwenden, um einen neuen AWS CloudHSM Schlüsselspeicher zu erstellen, der einem AWS CloudHSM Cluster im Konto und in der Region zugeordnet ist. Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Für die Produktion `CreateCustomKeyStore` sind folgende Parameterwerte erforderlich.

- CustomKeyStoreName — Ein benutzerfreundlicher Name für den benutzerdefinierten Schlüsselspeicher, der für das Konto eindeutig ist.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

- CloudHsmClusterId — Die Cluster-ID eines AWS CloudHSM Clusters, der die Anforderungen für einen AWS CloudHSM Schlüsselspeicher erfüllt.
- KeyStorePassword — Das Passwort des `kmsuser` CU-Kontos im angegebenen Cluster.
- TrustAnchorCertificate — Der Inhalt der `customerCA.crt` Datei, die Sie bei der Initialisierung des Clusters erstellt haben.

Im folgenden Beispiel wird eine fiktive Cluster-ID verwendet. Ersetzen Sie diese vor Ausführung des Befehls durch eine gültige Cluster-ID.

```
$ aws kms create-custom-key-store
        --custom-key-store-name ExampleCloudHSMKeyStore \
        --cloud-hsm-cluster-id cluster-1a23b4cdefg \
        --key-store-password kmsPswd \
        --trust-anchor-certificate <certificate-goes-here>
```

Wenn Sie die verwenden AWS CLI, können Sie die Trust-Anker-Zertifikatsdatei anstelle ihres Inhalts angeben. Im folgenden Beispiel befindet sich die Datei `customerCA.crt` im Stammverzeichnis.

```
$ aws kms create-custom-key-store
        --custom-key-store-name ExampleCloudHSMKeyStore \
        --cloud-hsm-cluster-id cluster-1a23b4cdefg \
        --key-store-password kmsPswd \
        --trust-anchor-certificate file://customerCA.crt
```

Bei einer erfolgreichen Produktion gibt `CreateCustomKeyStore` die ID des benutzerdefinierten Schlüsselspeichers zurück, wie in der folgenden Beispielantwort dargestellt.

```
{
    "CustomKeyStoreId": cks-1234567890abcdef0
}
```

Wenn die Produktion fehlschlägt, korrigieren Sie den in der Ausnahme angegebenen Fehler und versuchen Sie es erneut. Weitere Informationen finden Sie unter Fehlerbehebung für einen Custom Key Store.

Wenn Sie versuchen, einen AWS CloudHSM Schlüsselspeicher mit denselben Eigenschaftswerten wie ein vorhandener AWS CloudHSM Schlüsselspeicher ohne Verbindung zu erstellen, AWS KMS wird kein neuer AWS CloudHSM Schlüsselspeicher erstellt und es wird keine Ausnahme ausgelöst oder ein Fehler angezeigt. AWS KMS Erkennt das Duplikat stattdessen als wahrscheinliche Folge eines erneuten Versuchs und gibt die ID des vorhandenen AWS CloudHSM Schlüsselspeichers zurück.

Weiter: Um den AWS CloudHSM Schlüsselspeicher zu verwenden, verbinden Sie ihn mit seinem AWS CloudHSM Cluster.

Sehen Sie sich einen AWS CloudHSM Schlüsselspeicher an

Sie können die AWS CloudHSM Schlüsselspeicher für jedes Konto und jede Region mithilfe der AWS KMS Konsole oder der DescribeCustomKeyStoresOperation anzeigen.

Mithilfe der AWS KMS Konsole

Wenn Sie sich die AWS CloudHSM Schlüsselspeicher in der ansehen AWS Management Console, können Sie Folgendes sehen:

- Name und ID des benutzerdefinierten Schlüsselspeichers
- Die ID des zugehörigen AWS CloudHSM Clusters
- Die Nummer von HSMs im Cluster
- Der aktuelle Verbindungsstatus

Der Wert Disconnected für den Verbindungsstatus (Status) gibt an, dass der benutzerdefinierte Schlüsselspeicher neu ist und noch nie verbunden wurde oder dass er absichtlich vom Cluster getrennt wurde. AWS CloudHSM Wenn Ihre Versuche, einen KMS-Schlüssel in einem verbundenen benutzerdefinierten Schlüsselspeicher zu verwenden, fehlschlagen, kann dies auf ein Problem mit dem benutzerdefinierten Schlüsselspeicher oder seinem AWS CloudHSM Cluster hinweisen. Weitere Informationen dazu finden Sie unter Beheben eines fehlerhaften KMS-Schlüssels.

Gehen Sie wie folgt vor, um die AWS CloudHSM Schlüsselspeicher in einem bestimmten Konto und einer bestimmten Region anzuzeigen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (benutzerdefinierte Schlüsselspeicher) AWS CloudHSM -Schlüsselspeicher aus.

Klicken Sie auf das Zahnradsymbol unter der Schaltfläche Create key store (Schlüsselspeicher erstellen), um die Anzeige anzupassen.

Verwenden der API AWS KMS

Verwenden Sie den DescribeCustomKeyStoresVorgang, um Ihre AWS CloudHSM wichtigsten Speicher anzuzeigen. Standardmäßig gibt diese Operation alle benutzerdefinierten Schlüsselspeicher im Konto und in der Region zurück. Sie können jedoch entweder den Parameter `CustomKeyStoreId` oder `CustomKeyStoreName` (aber nicht beide) verwenden, um die Ausgabe auf einen bestimmten benutzerdefinierten Schlüsselspeicher zu begrenzen. Bei AWS CloudHSM Schlüsselspeichern besteht die Ausgabe aus der ID und dem Namen des benutzerdefinierten Schlüsselspeichers, dem Typ des benutzerdefinierten Schlüsselspeichers, der ID des zugehörigen AWS CloudHSM Clusters und dem Verbindungsstatus. Wenn der Verbindungsstatus auf einen Fehler hinweist, enthält die Ausgabe außerdem einen Fehlercode, der die Ursache angibt.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Beispielsweise gibt der folgende Befehl alle benutzerdefinierten Schlüsselspeicher im Konto und in der Region zurück. Sie können die Parameter `Marker` und `Limit` verwenden, um durch die benutzerdefinierten Schlüsselspeicher in der Ausgabe zu blättern.

```
$ aws kms describe-custom-key-stores
```

Der folgende Beispielbefehl verwendet den Parameter `CustomKeyStoreName`, um nur den benutzerdefinierten Schlüsselspeicher mit dem Anzeigenamen `ExampleCloudHSMKeyStore` abzurufen. In jedem Befehl können Sie entweder den Parameter `CustomKeyStoreName` oder `CustomKeyStoreId` (aber nicht beide) verwenden.

Die folgende Beispielausgabe stellt einen AWS CloudHSM Schlüsselspeicher dar, der mit seinem AWS CloudHSM Cluster verbunden ist.

> **Note**
>
> Das `CustomKeyStoreType` Feld wurde der `DescribeCustomKeyStores` Antwort hinzugefügt, um AWS CloudHSM Schlüsselspeicher von externen Schlüsselspeichern zu unterscheiden.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleCloudHSMKeyStore
{
   "CustomKeyStores": [
      {
         "CloudHsmClusterId": "cluster-1a23b4cdefg",
         "ConnectionState": "CONNECTED",
         "CreationDate": "1.499288695918E9",
         "CustomKeyStoreId": "cks-1234567890abcdef0",
         "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
         "CustomKeyStoreType": "AWS_CLOUDHSM",
         "TrustAnchorCertificate": "<certificate appears here>"
      }
   ]
}
```

Ein `Disconnected` Wert `ConnectionState` von gibt an, dass mit einem benutzerdefinierten Schlüsselspeicher noch nie eine Verbindung hergestellt wurde oder dass die Verbindung zu seinem AWS CloudHSM Cluster absichtlich getrennt wurde. Wenn jedoch Versuche, einen KMS-Schlüssel in einem verbundenen AWS CloudHSM Schlüsselspeicher zu verwenden, fehlschlagen, kann dies auf ein Problem mit dem AWS CloudHSM Schlüsselspeicher oder seinem AWS CloudHSM Cluster hinweisen. Weitere Informationen dazu finden Sie unter Beheben eines fehlerhaften KMS-Schlüssels.

Wenn der `ConnectionState` des benutzerdefinierten Schlüsselspeichers `FAILED` lautet, enthält die `DescribeCustomKeyStores`-Antwort ein `ConnectionErrorCode`-Element, das den Grund für den Fehler angibt.

In der folgenden Ausgabe deutet der Wert `INVALID_CREDENTIALS` beispielsweise darauf hin, dass die Verbindung des benutzerdefinierten Schlüsselspeichers fehlgeschlagen ist, da das `kmsuser`-Passwort ungültig ist. Weitere Informationen zu diesem und anderen Verbindungsfehlern finden Sie unter Fehlerbehebung für einen Custom Key Store.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
```

```
    "CustomKeyStores": [
       {
          "CloudHsmClusterId": "cluster-1a23b4cdefg",
          "ConnectionErrorCode": "INVALID_CREDENTIALS",
          "ConnectionState": "FAILED",
          "CustomKeyStoreId": "cks-1234567890abcdef0",
          "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
          "CustomKeyStoreType": "AWS_CLOUDHSM",
          "CreationDate": "1.499288695918E9",
          "TrustAnchorCertificate": "<certificate appears here>"
       }
    ]
}
```

Weitere Informationen:

- Externe Schlüsselspeicher anzeigen
- Identifizieren Sie KMS-Schlüssel in AWS CloudHSM Schlüsselspeichern
- AWS KMS API-Aufrufe protokollieren mit AWS CloudTrail

Einstellungen AWS CloudHSM für den Schlüsselspeicher bearbeiten

Sie können die Einstellungen eines vorhandenen AWS CloudHSM Schlüsselspeichers ändern. Der benutzerdefinierte Schlüsselspeicher muss vom AWS CloudHSM Cluster getrennt werden.

Um die Einstellungen für AWS CloudHSM den Schlüsselspeicher zu bearbeiten:

1. Trennen Sie den benutzerdefinierten Schlüsselspeicher von dessen AWS CloudHSM -Cluster.

   Solange der benutzerdefinierte Schlüsselspeicher getrennt ist, können Sie im benutzerdefinierten Schlüsselspeicher keine AWS KMS keys (KMS-Schlüssel) erstellen und die darin enthaltenen KMS-Schlüssel nicht für kryptografische Operationen verwenden.
2. Bearbeiten Sie eine oder mehrere Einstellungen für den AWS CloudHSM Schlüsselspeicher.

   Sie können in einem benutzerdefinierten Schlüsselspeicher die folgenden Einstellungen ändern:

   Anzeigename des benutzerdefinierten Schlüsselspeichers.

   Geben Sie einen neuen Anzeigenamen ein. Der neue Name muss unter allen benutzerdefinierten Schlüsselspeichern in Ihrem eindeutig sein AWS-Konto.

> **Important**
>
> Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

Die Cluster-ID des zugehörigen AWS CloudHSM Clusters.

Bearbeiten Sie diesen Wert, um den ursprünglichen AWS CloudHSM Cluster durch einen verwandten Cluster zu ersetzen. Sie können diese Funktion verwenden, um einen benutzerdefinierten Schlüsselspeicher zu reparieren, falls sein AWS CloudHSM Cluster beschädigt oder gelöscht wird.

Geben Sie einen AWS CloudHSM Cluster an, der den Backup-Verlauf mit dem ursprünglichen Cluster teilt und der die Anforderungen für die Zuordnung zu einem benutzerdefinierten Schlüsselspeicher erfüllt, einschließlich zweier, die HSMs in unterschiedlichen Availability Zones aktiv sind. Cluster mit einem gemeinsamen Sicherungsverlauf haben dasselbe Cluster-Zertifikat. Verwenden Sie den DescribeClustersVorgang, um das Clusterzertifikat eines Clusters anzuzeigen. Sie können das Bearbeitungs-Feature nicht verwenden, um den benutzerdefinierten Schlüsselspeicher mit einem nicht verwandten AWS CloudHSM -Cluster zu verknüpfen.

Das aktuelle Passwort des `kmsuser` Kryptobenutzers (Crypto User, CU).

Gibt AWS KMS das aktuelle Passwort der `kmsuser` CU im AWS CloudHSM Cluster an. Durch diese Aktion wird das Passwort der `kmsuser` CU im AWS CloudHSM Cluster nicht geändert.

Wenn Sie das Passwort der `kmsuser` CU im AWS CloudHSM Cluster ändern, verwenden Sie diese Funktion, um AWS KMS das neue `kmsuser` Passwort mitzuteilen. Andernfalls kann sich AWS KMS nicht am Cluster anmelden und jeder Versuch, den benutzerdefinierten Schlüsselspeicher mit dem Cluster zu verbinden, schlägt fehl.

3. Verbinden Sie den benutzerdefinierten Schlüsselspeicher wieder mit dessen AWS CloudHSM -Cluster.

Bearbeiten Sie Ihre Key-Store-Einstellungen

Sie können Ihre AWS CloudHSM Key-Store-Einstellungen in der AWS KMS Konsole oder mithilfe des UpdateCustomKeyStoreVorgangs bearbeiten.

Mithilfe der AWS KMS Konsole

Wenn Sie einen AWS CloudHSM Schlüsselspeicher bearbeiten, können Sie einen oder mehrere der konfigurierbaren Werte ändern.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (benutzerdefinierte Schlüsselspeicher) AWS CloudHSM -Schlüsselspeicher aus.
4. Wählen Sie die Zeile des AWS CloudHSM Schlüsselspeichers aus, den Sie bearbeiten möchten.

   Wenn der Wert in der Spalte Verbindungsstatus nicht Getrennt lautet, müssen Sie den benutzerdefinierten Schlüsselspeicher trennen, um ihn bearbeiten zu können. (Wählen Sie im Menü Key store actions(Key Store-Aktionen) die Option Disconnect (Trennen) aus.)

   Solange ein AWS CloudHSM Schlüsselspeicher getrennt ist, können Sie den AWS CloudHSM Schlüsselspeicher und seine KMS-Schlüssel verwalten, aber Sie können im AWS CloudHSM Schlüsselspeicher keine KMS-Schlüssel erstellen oder verwenden.
5. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Edit (Bearbeiten) aus.
6. Führen Sie eine oder mehrere der folgenden Aktionen aus.
   - Geben Sie einen neuen Anzeigenamen für den benutzerdefinierten Schlüsselspeicher ein.
   - Geben Sie die Cluster-ID eines zugehörigen AWS CloudHSM Clusters ein.
   - Geben Sie das aktuelle Passwort des `kmsuser` Krypto-Benutzers im zugehörigen AWS CloudHSM Cluster ein.
7. Wählen Sie Speichern.

   Wenn der Vorgang erfolgreich ist, wird Ihnen eine Meldung mit einer Beschreibung der von Ihnen bearbeiteten Einstellungen angezeigt. Wenn der Vorgang nicht erfolgreich ist, wird Ihnen eine

Fehlermeldung mit einer Beschreibung des Problems und Hilfestellung zur Fehlerbehebung angezeigt. Wenn Sie weitere Hilfe benötigen, beachten Sie den Abschnitt Fehlerbehebung für einen Custom Key Store.

8. Stellen Sie die Verbindung des benutzerdefinierten Schlüsselspeichers wieder her.

   Um den AWS CloudHSM Schlüsselspeicher verwenden zu können, müssen Sie ihn nach der Bearbeitung erneut verbinden. Sie können den AWS CloudHSM -Schlüsselspeicher getrennt lassen. Solange die Verbindung unterbrochen ist, können Sie jedoch keine KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher erstellen oder die KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher für kryptografische Operationen verwenden.

Verwenden der API AWS KMS

Verwenden Sie den UpdateCustomKeyStoreVorgang, um die Eigenschaften eines AWS CloudHSM Schlüsselspeichers zu ändern. Sie können im selben Befehl mehrere Eigenschaften eines benutzerdefinierten Schlüsselspeichers ändern. Wenn der Vorgang erfolgreich ist, werden eine HTTP 200-Antwort und ein JSON-Objekt ohne Eigenschaften AWS KMS zurückgegeben. Verwenden Sie den DescribeCustomKeyStoresVorgang, um zu überprüfen, ob die Änderungen wirksam sind.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Verwenden Sie zunächst DisconnectCustomKeyStore, um den benutzerdefinierten Schlüsselspeicher von seinem AWS CloudHSM Cluster zu trennen. Ersetzen Sie die Beispiel-ID des benutzerdefinierten Schlüsselspeichers, cks-1234567890abcdef0, durch eine tatsächliche ID.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

Im ersten Beispiel wird UpdateCustomKeyStoreder Anzeigename des AWS CloudHSM Schlüsselspeichers in geändert`DevelopmentKeys`. Der Befehl verwendet den `CustomKeyStoreId` Parameter, um den AWS CloudHSM Schlüsselspeicher `CustomKeyStoreName` zu identifizieren und dann den neuen Namen für den benutzerdefinierten Schlüsselspeicher anzugeben.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --new-
custom-key-store-name DevelopmentKeys
```

Im folgenden Beispiel wird der Cluster, der einem AWS CloudHSM Schlüsselspeicher zugeordnet ist, in ein anderes Backup desselben Clusters geändert. Der Befehl verwendet den

`CustomKeyStoreId` Parameter, um den AWS CloudHSM Schlüsselspeicher zu identifizieren, und den `CloudHsmClusterId` Parameter, um die neue Cluster-ID anzugeben.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --cloud-
hsm-cluster-id cluster-1a23b4cdefg
```

Das folgende Beispiel zeigt AWS KMS , dass das aktuelle `kmsuser` Passwort lautet`ExamplePassword`. Der Befehl verwendet den `CustomKeyStoreId` Parameter, um den AWS CloudHSM Schlüsselspeicher zu identifizieren, und den `KeyStorePassword` Parameter, um das aktuelle Passwort anzugeben.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --key-
store-password ExamplePassword
```

Der letzte Befehl verbindet den AWS CloudHSM Schlüsselspeicher erneut mit seinem AWS CloudHSM Cluster. Sie können den benutzerdefinierten Schlüsselspeicher im getrennten Status belassen. Sie müssen die Verbindung jedoch wiederherstellen, bevor Sie neue KMS-Schlüssel erstellen oder vorhandene KMS-Schlüssel für kryptografische Operationen verwenden können. Ersetzen Sie die Beispiel-ID des benutzerdefinierten Schlüsselspeichers durch eine tatsächliche ID.

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

Einen AWS CloudHSM Schlüsselspeicher Connect

Neue AWS CloudHSM Schlüsselspeicher sind nicht verbunden. Bevor Sie Ihren AWS CloudHSM Schlüsselspeicher erstellen und verwenden AWS KMS keys können, müssen Sie ihn mit dem zugehörigen AWS CloudHSM Cluster verbinden. Sie können Ihren AWS CloudHSM Schlüsselspeicher jederzeit verbinden und die Verbindung trennen und seinen Verbindungsstatus einsehen.

Sie müssen Ihren AWS CloudHSM Schlüsselspeicher nicht verbinden. Sie können einen AWS CloudHSM Schlüsselspeicher auf unbestimmte Zeit in einem getrennten Zustand belassen und ihn nur dann verbinden, wenn Sie ihn benötigen. Möglicherweise möchten Sie die Verbindung jedoch von Zeit zu Zeit testen, um zu prüfen, ob die Einstellungen korrekt sind und eine Verbindungsherstellung möglich ist.

> **Note**
>
> AWS CloudHSM Schlüsselspeicher haben nur dann einen `DISCONNECTED` Verbindungsstatus, wenn der Schlüsselspeicher noch nie verbunden war oder wenn Sie die Verbindung explizit trennen. Wenn Ihr AWS CloudHSM Schlüsselspeicher-Verbindungsstatus lautet, Sie `CONNECTED` aber Probleme bei der Verwendung haben, stellen Sie sicher, dass der zugehörige AWS CloudHSM Cluster aktiv ist und mindestens einen aktiven enthält HSMs. Hilfestellung bei fehlgeschlagenen Verbindungen finden Sie unter the section called "Fehlerbehebung für einen Custom Key Store".

Wenn Sie eine Verbindung zu einem AWS CloudHSM Schlüsselspeicher herstellen AWS KMS , wird der zugehörige AWS CloudHSM Cluster gefunden, eine Verbindung zu diesem hergestellt, sich als `kmsuser`Crypto User (CU) beim AWS CloudHSM Client angemeldet und dann das `kmsuser` Passwort rotiert. AWS KMS bleibt beim AWS CloudHSM Client angemeldet, solange der AWS CloudHSM Schlüsselspeicher verbunden ist.

Um die Verbindung herzustellen, AWS KMS erstellt eine Sicherheitsgruppe, die `kms-<custom key store ID>` in der Virtual Private Cloud (VPC) des Clusters benannt ist. Die Sicherheitsgruppe hat eine einzige Regel, die eingehenden Datenverkehr von der Cluster-Sicherheitsgruppe zulässt. AWS KMS erstellt außerdem ein elastic network interface (ENI) in jeder Availability Zone des privaten Subnetzes für den Cluster. AWS KMS fügt der ENIs `kms-<cluster ID>` Sicherheitsgruppe und der Sicherheitsgruppe für den Cluster hinzu. Die Beschreibung der einzelnen ENIs lautet `KMS managed ENI for cluster <cluster-ID>`.

Der Verbindungsvorgang kann einige Zeit – bis zu 20 Minuten – in Anspruch nehmen.

Bevor Sie eine Verbindung zum AWS CloudHSM Schlüsselspeicher herstellen, stellen Sie sicher, dass er die Anforderungen erfüllt.

- Der zugehörige AWS CloudHSM Cluster muss mindestens ein aktives HSM enthalten. Um die Anzahl von HSMs im Cluster zu ermitteln, zeigen Sie den Cluster in der AWS CloudHSM Konsole an oder verwenden Sie den DescribeClustersVorgang. Falls erforderlich, können Sie ein HSM hinzufügen.
- Der Cluster muss über ein `kmsuser`Crypto-Benutzerkonto (CU) verfügen, aber dieses CU kann nicht beim Cluster angemeldet werden, wenn Sie eine Verbindung zum AWS CloudHSM Schlüsselspeicher herstellen. Hilfe zum Abmelden finden Sie unter Abmelden und erneutes Verbinden.

- Der Verbindungsstatus des AWS CloudHSM Schlüsselspeichers kann nicht `DISCONNECTING` oder sein`FAILED`. Verwenden Sie die AWS KMS Konsole oder die DescribeCustomKeyStoresAntwort, um den Verbindungsstatus anzuzeigen. Wenn der Verbindungsstatus `FAILED` lautet, trennen Sie den benutzerdefinierten Schlüsselspeicher, lösen Sie das Problem und stellen Sie anschließend die Verbindung her.

Hilfestellung bei fehlgeschlagenen Verbindungen finden Sie unter Beheben eines Verbindungsfehlers.

Wenn Ihr AWS CloudHSM Schlüsselspeicher verbunden ist, können Sie darin KMS-Schlüssel erstellen und vorhandene KMS-Schlüssel für kryptografische Operationen verwenden.

Connect zu Ihrem AWS CloudHSM Schlüsselspeicher her und stellen Sie erneut eine Verbindung her

Sie können in der AWS KMS Konsole oder mithilfe des Vorgangs eine Verbindung zu Ihrem AWS CloudHSM Schlüsselspeicher herstellen oder die Verbindung erneut herstellen. ConnectCustomKeyStore

Mithilfe der Konsole AWS KMS

Um einen AWS CloudHSM Schlüsselspeicher in der zu verbinden AWS Management Console, wählen Sie zunächst den AWS CloudHSM Schlüsselspeicher auf der Seite Benutzerdefinierte Schlüsselspeicher aus. Der Verbindungsvorgang kann bis zu 20 Minuten dauern.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (benutzerdefinierte Schlüsselspeicher) AWS CloudHSM -Schlüsselspeicher aus.
4. Wählen Sie die Zeile des AWS CloudHSM Schlüsselspeichers aus, den Sie verbinden möchten.

   Wenn der Verbindungsstatus des AWS CloudHSM Schlüsselspeichers „Fehlgeschlagen“ lautet, müssen Sie die Verbindung zum benutzerdefinierten Schlüsselspeicher trennen, bevor Sie eine Verbindung herstellen.
5. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Connect (Verbinden) aus.

AWS KMS beginnt mit dem Verbindungsaufbau Ihres benutzerdefinierten Schlüsselspeichers. Der Service findet den zugehörigen AWS CloudHSM -Cluster, schafft die erforderlichen Netzwerkinfrastruktur, stellt eine Verbindung her, meldet sich beim AWS CloudHSM -Cluster als `kmsuser`-CU an und rotiert das `kmsuser`-Passwort. Wenn die Produktion abgeschlossen ist, ändert sich der Verbindungsstatus in Verbunden.

Wenn die Produktion fehlschlägt, wird eine Fehlermeldung mit einer Beschreibung der Fehlerursache angezeigt. Bevor Sie erneut versuchen, eine Verbindung herzustellen, überprüfen Sie den Verbindungsstatus Ihres AWS CloudHSM Schlüsselspeichers. Wenn dieser Fehlgeschlagen lautet, müssen Sie die Verbindung des benutzerdefinierten Schlüsselspeichers trennen, bevor Sie erneut eine Verbindung herstellen. Wenn Sie Hilfe benötigen, beachten Sie den Abschnitt Fehlerbehebung für einen Custom Key Store.

Danach: the section called “Erstellen Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher”.

Verwenden der AWS KMS API

Verwenden Sie den ConnectCustomKeyStoreVorgang, um einen getrennten AWS CloudHSM Schlüsselspeicher zu verbinden. Der zugehörige AWS CloudHSM Cluster muss mindestens ein aktives HSM enthalten, und der Verbindungsstatus darf nicht sein. `FAILED`

Der Verbindungsvorgang nimmt einige Zeit – bis zu 20 Minuten – in Anspruch. Sofern sie nicht schnell fehlschlägt, gibt die Produktion eine HTTP-Antwort 200 und ein JSON-Objekt ohne Eigenschaften zurück. Diese erste Antwort gibt jedoch nicht an, dass die Verbindung erfolgreich war. Informationen zum Ermitteln des Verbindungsstatus des benutzerdefinierten Schlüsselspeichers finden Sie in der DescribeCustomKeyStoresAntwort.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Um den AWS CloudHSM Schlüsselspeicher zu identifizieren, verwenden Sie seine benutzerdefinierte Schlüsselspeicher-ID. Sie finden die ID auf der Seite Benutzerdefinierte Schlüsselspeicher in der Konsole oder indem Sie den DescribeCustomKeyStoresVorgang ohne Parameter verwenden. Ersetzen Sie vor Ausführung dieses Beispiels die Beispiel-ID durch eine gültige ID.

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

Verwenden Sie den DescribeCustomKeyStoresVorgang, um zu überprüfen, ob der AWS CloudHSM Schlüsselspeicher verbunden ist. Diese Produktion gibt standardmäßig alle benutzerdefinierten

Schlüsselspeicher innerhalb des Kontos und der Region zurück. Sie können jedoch entweder den Parameter `CustomKeyStoreId` oder `CustomKeyStoreName` (aber nicht beide) verwenden, um die Antwort auf bestimmte benutzerdefinierte Schlüsselspeicher zu begrenzen. Der `ConnectionState`-Wert `CONNECTED` gibt an, dass der benutzerdefinierte Schlüsselspeicher mit seinem AWS CloudHSM -Cluster verbunden ist.

> **Note**
>
> Das `CustomKeyStoreType` Feld wurde der `DescribeCustomKeyStores` Antwort hinzugefügt, um AWS CloudHSM Schlüsselspeicher von externen Schlüsselspeichern zu unterscheiden.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
   "CustomKeyStores": [
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "CONNECTED"
   ],
}
```

Wenn der `ConnectionState`-Wert „Failed (Fehlgeschlagen)" lautet, ist im Element `ConnectionErrorCode` die Fehlerursache angegeben. In diesem Fall AWS KMS konnte in Ihrem Konto kein AWS CloudHSM Cluster mit der Cluster-ID gefunden `cluster-1a23b4cdefg` werden. Wenn Sie den Cluster gelöscht haben, können Sie ihn von einer Sicherung des ursprünglichen Clusters wiederherstellen und dann die Cluster-ID für den benutzerdefinierten Schlüsselspeicher bearbeiten. Hilfe beim Beantworten eines Verbindungsfehlercodes finden Sie unter Beheben eines Verbindungsfehlers.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
   "CustomKeyStores": [
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
```

```
        "CustomKeyStoreType": "AWS_CLOUDHSM",
        "TrustAnchorCertificate": "<certificate string appears here>",
        "CreationDate": "1.499288695918E9",
        "ConnectionState": "FAILED"
        "ConnectionErrorCode": "CLUSTER_NOT_FOUND"
    ],
}
```

Verbindung zu einem AWS CloudHSM Schlüsselspeicher trennen

Wenn Sie die Verbindung zu einem AWS CloudHSM Schlüsselspeicher trennen, AWS KMS meldet sich der AWS CloudHSM Client ab, trennt die Verbindung zum zugehörigen AWS CloudHSM Cluster und entfernt die Netzwerkinfrastruktur, die zur Unterstützung der Verbindung erstellt wurde.

Solange ein AWS CloudHSM Schlüsselspeicher getrennt ist, können Sie den AWS CloudHSM Schlüsselspeicher und seine KMS-Schlüssel verwalten, aber Sie können keine KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher erstellen oder verwenden. Der Verbindungsstatus des Schlüsselspeichers lautet `DISCONNECTED` und der Schlüsselstatus der KMS-Schlüssel in dem benutzerdefinierten Schlüsselspeicher `Unavailable`, sofern er nicht `PendingDeletion` lautet. Sie können den AWS CloudHSM Schlüsselspeicher jederzeit wieder verbinden.

> **Note**
>
> AWS CloudHSM Schlüsselspeicher haben nur dann einen `DISCONNECTED` Verbindungsstatus, wenn der Schlüsselspeicher noch nie verbunden war oder wenn Sie die Verbindung explizit trennen. Wenn Ihr AWS CloudHSM Schlüsselspeicher-Verbindungsstatus lautet, Sie `CONNECTED` aber Probleme bei der Verwendung haben, stellen Sie sicher, dass der zugehörige AWS CloudHSM Cluster aktiv ist und mindestens einen aktiven enthält HSMs. Hilfestellung bei fehlgeschlagenen Verbindungen finden Sie unter the section called “Fehlerbehebung für einen Custom Key Store”.

Wenn Sie die Verbindung zu einem benutzerdefinierten Schlüsselspeicher trennen, werden die KMS-Schlüssel im Schlüsselspeicher sofort unbrauchbar (je nach letztendlicher Konsistenz). Ressourcen, die mit durch den KMS-Schlüssel geschützten Datenschlüsseln verschlüsselt wurden, sind jedoch nicht betroffen, bis der KMS-Schlüssel erneut verwendet wird, z. B. zur Entschlüsselung des Datenschlüssels. Dieses Problem betrifft AWS-Services, von denen viele Datenschlüssel verwenden, um Ihre Ressourcen zu schützen. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

> **Note**
>
> Sämtliche Versuche, KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zu erstellen oder vorhandene KMS-Schlüssel in kryptografischen Produktionen zu nutzen, schlagen fehl, während der benutzerdefinierte Schlüsselspeicher getrennt ist. Diese Aktion kann verhindern, dass Benutzer vertrauliche Daten speichern und darauf zugreifen.

Um die Auswirkung einer Trennung der Verbindung Ihres benutzerdefinierten Schlüsselspeichers besser beurteilen zu können, ermitteln Sie die KMS-Schlüssel im benutzerdefinierten Schlüsselspeicher und ihre bisherige Verwendung.

Sie können die Verbindung zu einem AWS CloudHSM Schlüsselspeicher aus folgenden Gründen trennen:

- Zum Rotieren des **`kmsuser`**-Passworts. AWS KMS ändert das `kmsuser`-Passwort jedes Mal, wenn eine Verbindung zu dem AWS CloudHSM -Cluster hergestellt wird. Um eine Passwort-Rotation zu erzwingen, trennen Sie einfach die Verbindung und stellen Sie sie wieder her.
- Um das Schlüsselmaterial für die KMS-Schlüssel im AWS CloudHSM Cluster zu prüfen. Wenn Sie die Verbindung zum benutzerdefinierten Schlüsselspeicher trennen, AWS KMS meldet es sich vom `kmsuser`Krypto-Benutzerkonto im AWS CloudHSM Client ab. So können Sie sich als `kmsuser`-CU beim Cluster anmelden und das Schlüsselmaterial für den KMS-Schlüssel prüfen und verwalten.
- Zum sofortigen Deaktivieren aller KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher. Sie können KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher deaktivieren und erneut aktivieren, indem Sie die DisableKeyOperation AWS Management Console oder verwenden. Diese Produktionen werden schnell ausgeführt, beziehen sich jedoch immer nur auf jeweils einen KMS-Schlüssel. Wenn Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher trennen, ändert sich der Schlüsselstatus aller KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher sofort auf`Unavailable`, sodass sie nicht für kryptografische Operationen verwendet werden können.
- Zur Behebung eines fehlgeschlagenen Verbindungsversuchs. Wenn der Versuch, eine Verbindung zu einem AWS CloudHSM Schlüsselspeicher herzustellen, fehlschlägt (der Verbindungsstatus des benutzerdefinierten Schlüsselspeichers ist`FAILED`), müssen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher trennen, bevor Sie erneut versuchen, eine Verbindung herzustellen.

Trennen Sie die Verbindung zu Ihrem AWS CloudHSM Schlüsselspeicher

Sie können die Verbindung zu Ihrem AWS CloudHSM Schlüsselspeicher in der AWS KMS Konsole oder mithilfe des DisconnectCustomKeyStoreVorgangs trennen.

Trennen Sie die Verbindung über die AWS KMS Konsole

Um die Verbindung zu einem verbundenen AWS CloudHSM Schlüsselspeicher in der AWS KMS Konsole zu trennen, wählen Sie zunächst den AWS CloudHSM Schlüsselspeicher auf der Seite Benutzerdefinierte Schlüsselspeicher aus.

1. Melden Sie sich bei der Konsole AWS Key Management Service (AWS KMS) an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (benutzerdefinierte Schlüsselspeicher) AWS CloudHSM -Schlüsselspeicher aus.
4. Wählen Sie die Zeile des externen Schlüsselspeichers aus, mit dem Sie die Verbindung trennen möchten.
5. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Disconnect (Verbindung trennen) aus.

Nach Abschluss der Produktion ändert sich der Verbindungsstatus von Verbindung wird getrennt in Verbindung wird getrennt. Wenn die Produktion fehlschlägt, wird eine Fehlermeldung mit einer Beschreibung des Problems und Hilfestellung zur Fehlerbehebung angezeigt. Wenn Sie weitere Hilfe benötigen, beachten Sie den Abschnitt Fehlerbehebung für einen Custom Key Store.

Trennen Sie die Verbindung mithilfe der API AWS KMS

Verwenden Sie den DisconnectCustomKeyStoreVorgang, um die Verbindung zu einem verbundenen AWS CloudHSM Schlüsselspeicher zu trennen. Wenn der Vorgang erfolgreich ist, werden eine HTTP 200-Antwort und ein JSON-Objekt ohne Eigenschaften AWS KMS zurückgegeben.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

In diesem Beispiel wird die Verbindung zu einem AWS CloudHSM Schlüsselspeicher getrennt. Ersetzen Sie vor Ausführung dieses Beispiels die Beispiel-ID durch eine gültige ID.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

Verwenden Sie den Vorgang, um zu überprüfen, ob der AWS CloudHSM DescribeCustomKeyStoresSchlüsselspeicher getrennt ist. Diese Produktion gibt standardmäßig alle benutzerdefinierten Schlüsselspeicher innerhalb des Kontos und der Region zurück. Sie können jedoch entweder den Parameter `CustomKeyStoreId` oder `CustomKeyStoreName` (aber nicht beide) verwenden, um die Antwort auf bestimmte benutzerdefinierte Schlüsselspeicher zu begrenzen. Der `ConnectionState` Wert von `DISCONNECTED` gibt an, dass dieser AWS CloudHSM Beispielschlüsselspeicher nicht mit seinem AWS CloudHSM Cluster verbunden ist.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
   "CustomKeyStores": [
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionState": "DISCONNECTED",
      "CreationDate": "1.499288695918E9",
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate string appears here>"
   ],
}
```

Einen AWS CloudHSM Schlüsselspeicher löschen

Wenn Sie einen AWS CloudHSM Schlüsselspeicher AWS KMS löschen, werden alle Metadaten über den AWS CloudHSM Schlüsselspeicher aus KMS gelöscht, einschließlich Informationen zu seiner Zuordnung zu einem AWS CloudHSM Cluster. Dieser Vorgang hat keine Auswirkungen auf den AWS CloudHSM Cluster HSMs, seine oder seine Benutzer. Sie können einen neuen AWS CloudHSM Schlüsselspeicher erstellen, der demselben AWS CloudHSM Cluster zugeordnet ist, aber Sie können den Löschvorgang nicht rückgängig machen.

Sie können nur einen AWS CloudHSM Schlüsselspeicher löschen, der nicht mit seinem AWS CloudHSM Cluster verbunden ist und keinen Schlüsselspeicher enthält. AWS KMS keys Vor dem Löschen eines benutzerdefinierten Schlüsselspeichers müssen Sie folgende Schritte ausführen.

- Stellen Sie sicher, dass Sie keinen der KMS-Schlüssel in dem Schlüsselspeicher für kryptografische Produktionen benötigen. Anschließend planen Sie die Löschung aller KMS-Schlüssel aus dem Schlüsselspeicher. Hilfe bei der Suche nach den KMS-Schlüsseln in einem

AWS CloudHSM Schlüsselspeicher finden Sie unterSuchen Sie die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher.

- Überprüfen Sie, ob alle KMS-Schlüssel gelöscht wurden. Informationen zum Anzeigen der KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher finden Sie unterthe section called “Identifizieren Sie KMS-Schlüssel in AWS CloudHSM Schlüsselspeichern”.
- Trennen Sie den AWS CloudHSM Schlüsselspeicher von seinem AWS CloudHSM Cluster.

Anstatt den AWS CloudHSM Schlüsselspeicher zu löschen, sollten Sie erwägen, ihn vom zugehörigen AWS CloudHSM Cluster zu trennen. Solange ein AWS CloudHSM Schlüsselspeicher getrennt ist, können Sie den AWS CloudHSM Schlüsselspeicher und seine Daten verwalten. AWS KMS keys Sie können jedoch keine KMS-Schlüssel im AWS CloudHSM Schlüsselspeicher erstellen oder verwenden. Sie können den AWS CloudHSM Schlüsselspeicher jederzeit wieder verbinden.

Löschen Sie Ihren AWS CloudHSM Schlüsselspeicher

Sie können Ihren AWS CloudHSM Schlüsselspeicher in der AWS KMS Konsole oder mithilfe des DeleteCustomKeyStoreVorgangs löschen.

Mithilfe der AWS KMS Konsole

Um einen AWS CloudHSM Schlüsselspeicher in der zu löschen AWS Management Console, wählen Sie zunächst den AWS CloudHSM Schlüsselspeicher auf der Seite Benutzerdefinierte Schlüsselspeicher aus.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (benutzerdefinierte Schlüsselspeicher) AWS CloudHSM -Schlüsselspeicher aus.
4. Suchen Sie die Zeile, die den AWS CloudHSM Schlüsselspeicher darstellt, den Sie löschen möchten. Wenn der Verbindungsstatus des AWS CloudHSM Schlüsselspeichers nicht Getrennt lautet, müssen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher trennen, bevor Sie ihn löschen.
5. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Delete (Löschen) aus.

Wenn der Vorgang abgeschlossen ist, wird eine Erfolgsmeldung angezeigt und der AWS CloudHSM Schlüsselspeicher wird nicht mehr in der Liste der Schlüsselspeicher angezeigt. Wenn die Produktion nicht erfolgreich ist, wird eine Fehlermeldung mit einer Beschreibung des Problems und Hilfestellung zur Fehlerbehebung angezeigt. Wenn Sie weitere Hilfe benötigen, beachten Sie den Abschnitt Fehlerbehebung für einen Custom Key Store.

Verwenden der AWS KMS API

Verwenden Sie den DeleteCustomKeyStoreVorgang, um einen AWS CloudHSM Schlüsselspeicher zu löschen. Wenn der Vorgang erfolgreich ist, werden eine HTTP 200-Antwort und ein JSON-Objekt ohne Eigenschaften AWS KMS zurückgegeben.

Stellen Sie zunächst sicher, dass der AWS CloudHSM Schlüsselspeicher keine enthält AWS KMS keys. Sie können benutzerdefinierte Schlüsselspeicher, die KMS-Schlüssel enthalten, nicht löschen. Der erste Beispielbefehl verwendet ListKeysund DescribeKey, um AWS KMS keys im AWS CloudHSM Schlüsselspeicher mit der *cks-1234567890abcdef0* benutzerdefinierten Schlüsselspeicher-ID zu suchen. In diesem Fall gibt der Befehl keine KMS-Schlüssel zurück. Ist dies der Fall, verwenden Sie den ScheduleKeyDeletionVorgang, um das Löschen der einzelnen KMS-Schlüssel zu planen.

Bash

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;
do aws kms describe-key --key-id $key |
grep '"CustomKeyStoreId": "cks-1234567890abcdef0"' --context 100; done
```

PowerShell

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyStoreId -eq
 'cks-1234567890abcdef0'
```

Trennen Sie als Nächstes die Verbindung AWS CloudHSM zum Schlüsselspeicher. In diesem Beispielbefehl wird der DisconnectCustomKeyStoreVorgang verwendet, um einen AWS CloudHSM Schlüsselspeicher von seinem AWS CloudHSM Cluster zu trennen. Vor der Ausführung dieses Befehls müssen Sie die Beispiel-ID des benutzerdefinierten Schlüsselspeichers durch eine gültige ID ersetzen.

Bash

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

PowerShell

```
PS C:\> Disconnect-KMSCustomKeyStore -CustomKeyStoreId cks-1234567890abcdef0
```

Nachdem der benutzerdefinierte Schlüsselspeicher getrennt wurde, können Sie ihn mithilfe des DeleteCustomKeyStoreVorgangs löschen.

Bash

```
$ aws kms delete-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

PowerShell

```
PS C:\> Remove-KMSCustomKeyStore -CustomKeyStoreId cks-1234567890abcdef0
```

Fehlerbehebung für einen Custom Key Store

AWS CloudHSM Key Stores sind so konzipiert, dass sie verfügbar und belastbar sind. Es gibt jedoch einige Fehlerbedingungen, die Sie möglicherweise beheben müssen, um Ihren AWS CloudHSM Schlüsselspeicher betriebsbereit zu halten.

Themen

- So reparieren Sie nicht-verfügbare KMS-Schlüssel
- Beheben eines fehlerhaften KMS-Schlüssels
- Beheben eines Verbindungsfehlers
- Wie man auf Fehler bei kryptografischen Produktionen reagiert
- Reparieren ungültiger kmsuser-Anmeldeinformationen
- Löschen von verwaistem Schlüsselmaterial
- Wiederherstellen von gelöschtem Schlüsselmaterial für einen KMS-Schlüssel
- Anmeldung als kmsuser

So reparieren Sie nicht-verfügbare KMS-Schlüssel

Der Schlüsselstatus von AWS KMS keys in einem AWS CloudHSM Schlüsselspeicher ist in der Regel`Enabled`. Wie bei allen KMS-Schlüsseln ändert sich der Schlüsselstatus, wenn Sie die KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher deaktivieren oder ihre Löschung planen. Im Gegensatz zu anderen KMS-Schlüsseln können die KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher aber auch den Schlüsselstatus `Unavailable` haben.

Der Schlüsselstatus `Unavailable` gibt an, dass sich der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher befindet, der absichtlich getrennt wurde und eventuelle Versuche, die Verbindung wiederherzustellen, fehlgeschlagen sind. Wenn ein KMS-Schlüssel nicht verfügbar ist, können Sie ihn anzeigen und verwalten, ihn jedoch nicht für kryptographische Produktionen verwenden.

Um den Schlüsselstatus eines KMS-Schlüssels zu ermitteln, zeigen Sie auf der Seite Customer managed keys (kundenverwaltete Schlüssel) das Feld Status des KMS-Schlüssels an. Oder verwenden Sie den DescribeKeyVorgang und sehen Sie sich das `KeyState` Element in der Antwort an. Details hierzu finden Sie unter Schlüssel identifizieren und anzeigen.

Die KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher haben den Schlüsselstatus `Unavailable` oder `PendingDeletion`. KMS-Schlüssel, die zur Löschung aus einem benutzerdefinierten Schlüsselspeicher geplant sind, haben den Schlüsselstatus `Pending Deletion`, auch wenn der benutzerdefinierten Schlüsselspeicher getrennt ist. Auf diese Weise können Sie die geplante Löschung des Schlüssels stornieren, ohne dass Sie erneut eine Verbindung mit dem Custom Key Store herstellen müssen.

Um die Nichtverfügbarkeit eines KMS-Schlüssels zu beheben, verbinden Sie den benutzerdefinierten Schlüsselspeicher wieder. Wenn der benutzerdefinierten Schlüsselspeicher wieder verbunden ist, wechselt der Schlüsselstatus der KMS-Schlüssels im benutzerdefinierten Schlüsselspeicher automatisch wieder zum vorherigen Zustand, etwa zu `Enabled` oder `Disabled`. KMS-Schlüssel, die zur Löschung ausstehen, bleiben im Status `PendingDeletion`. Während das Problem bestehen bleibt, ändert das Aktivieren und Deaktivieren eines nicht-verfügbaren KMS-Schlüssels dessen Schlüsselstatus nicht. Das Aktivieren und Deaktivieren wirkt sich erst aus, wenn der Schlüssel wieder verfügbar ist.

Für Hilfe bei fehlgeschlagenen Verbindungen vgl. Beheben eines Verbindungsfehlers.

Beheben eines fehlerhaften KMS-Schlüssels

Probleme bei der Erstellung und Verwendung von KMS-Schlüsseln in AWS CloudHSM Schlüsselspeichern können durch ein Problem mit Ihrem AWS CloudHSM Schlüsselspeicher, dem zugehörigen AWS CloudHSM Cluster, dem KMS-Schlüssel oder dem zugehörigen Schlüsselmaterial verursacht werden.

Wenn ein AWS CloudHSM Schlüsselspeicher von seinem AWS CloudHSM Cluster getrennt wird, lautet der Schlüsselstatus der KMS-Schlüssel im benutzerdefinierten Schlüsselspeicher. `Unavailable` Alle Anfragen zur Erstellung von KMS-Schlüsseln in einem getrennten AWS CloudHSM Schlüsselspeicher geben eine `CustomKeyStoreInvalidStateException` Ausnahme zurück. Alle Anforderungen zum Verschlüsseln, erneuten Verschlüsseln oder zum Generieren von Datenschlüsseln führen zu der Ausnahme `KMSInvalidStateException`. Um das Problem zu beheben, stellen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher erneut her.

Ihre Versuche, einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher für kryptografische Operationen zu verwenden, können jedoch fehlschlagen, selbst wenn sein Schlüsselstatus `Enabled` und der Verbindungsstatus des AWS CloudHSM Schlüsselspeichers ist. `Connected` Dies kann durch eine der folgenden Ursachen bedingt sein.

- Möglicherweise wurde das Schlüsselmaterial für den KMS-Schlüssel von dem zugehörigen AWS CloudHSM -Cluster gelöscht. Suchen Sie zur Untersuchung nach der Schlüssel-ID des Schlüsselmaterials für einen KMS-Schlüssel und versuchen Sie gegebenenfalls, das Schlüsselmaterial wiederherzustellen.
- Alle HSMs wurden aus dem AWS CloudHSM Cluster gelöscht, der dem AWS CloudHSM Schlüsselspeicher zugeordnet ist. Um einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher in einem kryptografischen Vorgang verwenden zu können, muss der zugehörige AWS CloudHSM Cluster mindestens ein aktives HSM enthalten. Verwenden Sie die AWS CloudHSM Konsole oder den Vorgang, um die Anzahl und den Status von HSMs in einem AWS CloudHSM Cluster zu überprüfen. DescribeClusters Verwenden Sie die AWS CloudHSM Konsole oder den CreateHsmVorgang, um dem Cluster ein HSM hinzuzufügen.
- Der dem AWS CloudHSM Schlüsselspeicher zugeordnete AWS CloudHSM Cluster wurde gelöscht. Um das Problem zu beheben, erstellen Sie einen Cluster aus einer Sicherung, die mit dem ursprünglichen Cluster verbunden ist, oder aus einer Sicherung, die zur Erstellung des ursprünglichen Clusters verwendet wurde. Bearbeiten Sie dann die Cluster-ID in den Einstellungen für den Custom Key Store. Detaillierte Anweisungen finden Sie unter Wiederherstellen von gelöschtem Schlüsselmaterial für einen KMS-Schlüssel.

- Für den AWS CloudHSM Cluster, der dem benutzerdefinierten Schlüsselspeicher zugeordnet war, waren keine PKCS #11 -Sitzungen verfügbar. Dies tritt normalerweise in Zeiten mit hohem Burst-Verkehr auf, wenn zusätzliche Sitzungen erforderlich sind, um den Datenverkehr zu bedienen. Um auf eine `KMSInternalException` mit einer Fehlermeldung über PKCS#11-Sitzungen zu antworten, gehen Sie zurück und wiederholen Sie die Anfrage.

Beheben eines Verbindungsfehlers

Wenn Sie versuchen, einen AWS CloudHSM Schlüsselspeicher mit seinem AWS CloudHSM Cluster zu verbinden, der Vorgang jedoch fehlschlägt, ändert sich der Verbindungsstatus des AWS CloudHSM Schlüsselspeichers auf`FAILED`. Um den Verbindungsstatus eines AWS CloudHSM Schlüsselspeichers zu ermitteln, verwenden Sie die AWS KMS Konsole oder den DescribeCustomKeyStoresVorgang.

Alternativ können einige Verbindungsversuche aufgrund leicht zu erkennender Cluster-Konfigurationsfehler fehlschlagen. In diesem Fall lautet der Verbindungsstatus immer noch `DISCONNECTED`. Diese Fehler geben eine Fehlermeldung oder Ausnahme mit einer Begründung für den fehlgeschlagenen Verbindungsversuch zurück. Lesen Sie die Beschreibung der Ausnahme und die Clusteranforderungen, beheben Sie das Problem, aktualisieren Sie gegebenenfalls den AWS CloudHSM Schlüsselspeicher und versuchen Sie erneut, eine Verbindung herzustellen.

Wenn der Verbindungsstatus lautet`FAILED`, führen Sie den DescribeCustomKeyStoresVorgang aus und sehen Sie sich das `ConnectionErrorCode` Element in der Antwort an.

> **Note**
>
> Wenn der Verbindungsstatus eines AWS CloudHSM Schlüsselspeichers lautet`FAILED`, müssen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher trennen, bevor Sie erneut versuchen können, eine Verbindung herzustellen. Sie können einen AWS CloudHSM Schlüsselspeicher mit einem `FAILED` Verbindungsstatus nicht verbinden.

- `CLUSTER_NOT_FOUND`gibt an, dass AWS KMS kein AWS CloudHSM Cluster mit der angegebenen Cluster-ID gefunden werden kann. Dies kann auftreten, wenn die falsche Cluster-ID für eine API-Produktion bereitgestellt wurde, oder wenn der Cluster gelöscht und nicht ersetzt wurde. Um diesen Fehler zu beheben, überprüfen Sie die Cluster-ID, z. B. mithilfe der AWS CloudHSM Konsole oder mithilfe des DescribeClustersVorgangs. Wenn der Cluster gelöscht wurde, erstellen Sie einen Cluster aus einer möglichst neuen Sicherung des Originals. Trennen Sie dann die

Verbindung AWS CloudHSM zum Schlüsselspeicher, bearbeiten Sie die AWS CloudHSM Cluster-ID-Einstellung für den Schlüsselspeicher und verbinden Sie den AWS CloudHSM Schlüsselspeicher erneut mit dem Cluster.

- `INSUFFICIENT_CLOUDHSM_HSMS`gibt an, dass der zugehörige AWS CloudHSM Cluster keine HSMs enthält. Zum herstellen einer Verbindung muss der Cluster mindestens über ein HSM verfügen. Verwenden Sie die DescribeClustersOperation, um die Nummer von HSMs im Cluster zu ermitteln. Um diesen Fehler zu beheben, fügen Sie mindestens ein HSM dem Cluster hinzu. Wenn Sie mehrere hinzufügen HSMs, ist es am besten, sie in verschiedenen Availability Zones zu erstellen.
- `INSUFFICIENT_FREE_ADDRESSES_IN_SUBNET`gibt an, dass der AWS CloudHSM Schlüsselspeicher nicht mit seinem AWS CloudHSM Cluster verbunden werden AWS KMS konnte, weil mindestens ein dem Cluster zugeordnetes privates Subnetz keine verfügbaren IP-Adressen hat. Eine AWS CloudHSM Schlüsselspeicherverbindung erfordert eine freie IP-Adresse in jedem der zugehörigen privaten Subnetze, obwohl zwei vorzuziehen sind.

  Sie können in einem vorhandenen Subnetz keine IP-Adressen (CIDR-Blöcke) hinzufügen. Verschieben oder löschen Sie nach Möglichkeit andere Ressourcen, die die IP-Adressen im Subnetz verwenden, z. B. ungenutzte EC2 Instances oder elastische Netzwerkschnittstellen. Andernfalls können Sie einen Cluster aus einer aktuellen Sicherung des AWS CloudHSM Clusters mit neuen oder vorhandenen privaten Subnetzen erstellen, die über mehr freien Adressraum verfügen. Um den neuen Cluster dann Ihrem AWS CloudHSM Schlüsselspeicher zuzuordnen, trennen Sie die Verbindung zum benutzerdefinierten Schlüsselspeicher, ändern Sie die Cluster-ID des AWS CloudHSM Schlüsselspeichers in die ID des neuen Clusters und versuchen Sie erneut, eine Verbindung herzustellen.

  > **Tip**
  >
  > Verwenden Sie das neueste Backup des Clusters, um ein Zurücksetzen des `kmsuser` Kennworts zu vermeiden. AWS CloudHSM

- `INTERNAL_ERROR`gibt an, dass die Anfrage aufgrund eines internen Fehlers nicht abgeschlossen werden AWS KMS konnte. Wiederholen Sie die Anforderung. Trennen Sie bei `ConnectCustomKeyStore` Anfragen die Verbindung zum AWS CloudHSM Schlüsselspeicher, bevor Sie erneut versuchen, eine Verbindung herzustellen.
- `INVALID_CREDENTIALS`gibt an, dass AWS KMS man sich nicht beim zugehörigen AWS CloudHSM Cluster anmelden kann, weil er nicht das richtige `kmsuser` Kontopasswort hat. Für Unterstützung bei diesem Fehler vgl. Reparieren ungültiger `kmsuser`-Anmeldeinformationen.

- `NETWORK_ERRORS` weist normalerweise auf vorübergehende Netzwerkprobleme hin. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher, warten Sie einige Minuten und versuchen Sie erneut, eine Verbindung herzustellen.
- `SUBNET_NOT_FOUND`gibt an, dass mindestens ein Subnetz in der AWS CloudHSM Clusterkonfiguration gelöscht wurde. Wenn AWS KMS nicht alle Subnetze in der Clusterkonfiguration gefunden werden können, schlagen Versuche fehl, den AWS CloudHSM Schlüsselspeicher mit dem AWS CloudHSM Cluster zu verbinden.

  Um diesen Fehler zu beheben, erstellen Sie einen Cluster aus einer aktuellen Sicherung desselben AWS CloudHSM Clusters. (Dieser Prozess erstellt eine neue Clusterkonfiguration mit einer VPC und privaten Subnetzen.) Stellen Sie sicher, dass der neue Cluster die Anforderungen für einen benutzerdefinierten Schlüsselspeicher erfüllt, und notieren Sie sich die neue Cluster-ID. Um den neuen Cluster dann Ihrem AWS CloudHSM Schlüsselspeicher zuzuordnen, trennen Sie die Verbindung zum benutzerdefinierten Schlüsselspeicher, ändern Sie die Cluster-ID des AWS CloudHSM Schlüsselspeichers in die ID des neuen Clusters und versuchen Sie erneut, eine Verbindung herzustellen.

  > **Tip**
  >
  > Verwenden Sie das neueste Backup des Clusters, um ein Zurücksetzen des `kmsuser` Kennworts zu vermeiden. AWS CloudHSM

- `USER_LOCKED_OUT` gibt an, dass das `kmsuser`-Kryptobenutzer (CU)-Konto aufgrund zu vieler fehlerhafter Passworteingaben aus dem zugehörigen AWS CloudHSM -Cluster ausgesperrt wurde. Für Unterstützung bei diesem Fehler vgl. Reparieren ungültiger `kmsuser`-Anmeldeinformationen.

  Um diesen Fehler zu beheben, trennen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher und verwenden Sie den Befehl user change-password in der CloudHSM CLI, um das Kontopasswort zu ändern. `kmsuser` Bearbeiten Sie dann die `kmsuser`-Passworteinstellung für den Custom Key Store, und versuchen Sie erneut, eine Verbindung herzustellen. Falls Sie Hilfe benötigen, verwenden Sie die Vorgehensweise aus dem Thema Reparieren ungültiger `kmsuser`-Anmeldeinformationen.

- `USER_LOGGED_IN`gibt an, dass das `kmsuser` CU-Konto beim zugehörigen Cluster angemeldet ist. AWS CloudHSM Dadurch wird AWS KMS verhindert, dass das `kmsuser` Kontopasswort geändert und die Anmeldung am Cluster erfolgt. Melden Sie den `kmsuser`-CU vom Cluster ab, um diesen Fehler zu beheben. Wenn Sie das `kmsuser` Passwort für die Anmeldung am Cluster geändert haben, müssen Sie auch den Wert des Schlüsselspeicherkennworts für den AWS CloudHSM

Schlüsselspeicher aktualisieren. Weitere Informationen dazu finden Sie unter Abmelden und erneutes Verbinden.

- `USER_NOT_FOUND`gibt an, dass im zugehörigen AWS CloudHSM Cluster AWS KMS kein `kmsuser` CU-Konto gefunden werden kann. Um diesen Fehler zu beheben, erstellen Sie ein `kmsuser` CU-Konto im Cluster und aktualisieren Sie dann den Kennwortwert des Schlüsselspeichers für den AWS CloudHSM Schlüsselspeicher. Weitere Informationen dazu finden Sie unter Reparieren ungültiger `kmsuser`-Anmeldeinformationen.

Wie man auf Fehler bei kryptografischen Produktionen reagiert

Eine kryptografische Produktion, die einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher verwendet, schlägt möglicherweise mit einem `KMSInvalidStateException` fehl. Die folgenden Fehlermeldungen könnten dem `KMSInvalidStateException` beiliegen.

> KMS kann nicht mit Ihrem CloudHSM-Cluster kommunizieren. Dies könnte ein vorübergehendes Netzwerkproblem sein. Wenn dieser Fehler wiederholt auftritt, überprüfen Sie, ob die Netzwerk ACLs - und Sicherheitsgruppenregeln für die VPC Ihres AWS CloudHSM Clusters korrekt sind.

- Obwohl dies ein HTTPS-400-Fehler ist, kann es auf vorübergehende Netzwerkprobleme zurückzuführen sein. Um zu antworten, versuchen Sie zunächst die Anforderung erneut. Wenn es jedoch weiterhin fehlschlägt, überprüfen Sie die Konfiguration der Netzwerkkomponenten. Dieser Fehler wird höchstwahrscheinlich durch die Fehlkonfiguration einer Netzwerkkomponente verursacht, z. B. eine Firewall-Regel oder eine VPC -Sicherheitsgruppen-Regel, die ausgehenden Datenverkehr blockiert. Beispielsweise kann KMS nicht mit AWS CloudHSM Clustern über IPv6 kommunizieren. Einzelheiten zu den Voraussetzungen finden Sie unterErstellen Sie einen AWS CloudHSM Schlüsselspeicher.

> KMS kann nicht mit Ihrem AWS CloudHSM Cluster kommunizieren, da der kmsuser gesperrt ist. Wenn dieser Fehler wiederholt auftritt, trennen Sie die Verbindung zum AWS CloudHSM Schlüssel speicher und setzen Sie das Kennwort für das kmsuser-Konto zurück. Aktualisieren Sie das kmsuser-Passwort für den benutzerdefinierten Schlüsselspeicher und versuchen Sie es erneut mit der Anfrage.

- Diese Fehlermeldung gibt an, dass das `kmsuser`-Crypto-Benutzer (CU)-Konto aufgrund zu vieler fehlerhafter Passworteingaben aus dem zugehörigen AWS CloudHSM -Cluster ausgesperrt wurde. Für Unterstützung bei diesem Fehler vgl. Trennen und Anmelden.

Reparieren ungültiger **`kmsuser`**-Anmeldeinformationen

Wenn Sie eine Verbindung zu einem AWS CloudHSM Schlüsselspeicher herstellen, AWS KMS meldet er sich als `kmsuser`Crypto User (CU) beim zugehörigen AWS CloudHSM Cluster an. Er bleibt angemeldet, bis die Verbindung zum AWS CloudHSM Schlüsselspeicher getrennt wird. Die Antwort DescribeCustomKeyStores meldet `ConnectionState` als `FAILED` und `ConnectionErrorCode` mit dem Wert `INVALID_CREDENTIALS` (siehe folgendes Beispiel).

Wenn Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher trennen und das `kmsuser` Passwort ändern, AWS KMS können Sie sich nicht mit den Anmeldeinformationen des `kmsuser` CU-Kontos beim AWS CloudHSM Cluster anmelden. Daher schlagen alle Versuche fehl, eine Verbindung zum AWS CloudHSM Schlüsselspeicher herzustellen. Die Antwort `DescribeCustomKeyStores` meldet `ConnectionState` als `FAILED` und `ConnectionErrorCode` mit dem Wert `INVALID_CREDENTIALS` (siehe folgendes Beispiel).

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleKeyStore
{
   "CustomKeyStores": [
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionErrorCode": "INVALID_CREDENTIALS"
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "FAILED"
   ],
}
```

Weiterhin gilt, dass, AWS CloudHSM nach fünf fehlgeschlagenen Anmeldeversuchen bei dem Cluster mit einem inkorrekten Passwort das Benutzerkonto sperrt. Zur Anmeldung bei dem Cluster müssen Sie das Kontopasswort ändern.

Wenn beim Versuch, sich als `kmsuser` CU beim Cluster anzumelden, eine Sperrantwort angezeigt AWS KMS wird, schlägt die Anforderung, eine Verbindung zum AWS CloudHSM Schlüsselspeicher herzustellen, fehl. Die DescribeCustomKeyStoresAntwort enthält einen Wert `ConnectionState`

von `FAILED` und einen `ConnectionErrorCode` Wert von`USER_LOCKED_OUT`, wie im folgenden Beispiel gezeigt.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleKeyStore
{
   "CustomKeyStores": [
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionErrorCode": "USER_LOCKED_OUT"
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "FAILED"
   ],
}
```

Gehen Sie zum Beheben eines dieser Zustände wie folgt vor.

1. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher.
2. Führen Sie den DescribeCustomKeyStoresVorgang aus und zeigen Sie den Wert des `ConnectionErrorCode` Elements in der Antwort an.
   - Wenn der `ConnectionErrorCode`-Wert `INVALID_CREDENTIALS` ist, ermitteln Sie das aktuelle Passwort für das `kmsuser`-Konto. Verwenden Sie bei Bedarf den Befehl user change-password in der CloudHSM CLI, um das Passwort auf einen bekannten Wert festzulegen.
   - Wenn der `ConnectionErrorCode` Wert lautet`USER_LOCKED_OUT`, müssen Sie den Befehl user change-password in der CloudHSM-CLI verwenden, um das Passwort zu ändern. `kmsuser`
3. Bearbeiten Sie die `kmsuser` Passworteinstellung, so dass sie dem aktuellen `kmsuser`-Passwort in dem Cluster entspricht. Diese Aktion weist AWS KMS an, welches Passwort für die Anmeldung bei dem Cluster zu verwenden ist. Das `kmsuser`-Passwort in dem Cluster wird nicht geändert.
4. Verbinden Sie den Custom Key Store.

Löschen von verwaistem Schlüsselmaterial

Nachdem Sie das Löschen eines KMS-Schlüssels aus einem AWS CloudHSM Schlüsselspeicher geplant haben, müssen Sie das entsprechende Schlüsselmaterial möglicherweise manuell aus dem zugehörigen Cluster löschen. AWS CloudHSM

Wenn Sie einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher erstellen, werden die KMS-Schlüsselmetadaten im zugehörigen AWS CloudHSM Cluster AWS KMS erstellt AWS KMS und das Schlüsselmaterial generiert. Wenn Sie das Löschen eines KMS-Schlüssels in einem AWS CloudHSM Schlüsselspeicher planen, werden nach Ablauf der Wartezeit die KMS-Schlüsselmetadaten AWS KMS gelöscht. AWS KMS bemüht sich dann nach besten Kräften, das entsprechende Schlüsselmaterial aus dem AWS CloudHSM Cluster zu löschen. Der Versuch schlägt möglicherweise fehl, wenn AWS KMS kein Zugriff auf den Cluster möglich ist, z. B. wenn die Verbindung zum AWS CloudHSM Schlüsselspeicher getrennt wird oder das `kmsuser` Passwort geändert wird. AWS KMS versucht nicht, Schlüsselmaterial aus Cluster-Backups zu löschen.

AWS KMS meldet die Ergebnisse seines Versuchs, das Schlüsselmaterial aus dem Cluster zu löschen, im `DeleteKey` Ereigniseintrag Ihrer AWS CloudTrail Protokolle. Es erscheint im `backingKeysDeletionStatus`-Element des `additionalEventData`-Elements, wie im folgenden Beispieleintrag gezeigt. Der Eintrag enthält auch den KMS-Schlüssel ARN, die AWS CloudHSM Cluster-ID und die ID (`backing-key-id`) des Schlüsselmaterials.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-12-10T14:23:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements":  {
        "keyId":"arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0",
```

```
        "clusterId": "cluster-1a23b4cdefg",
        "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]",
        "backingKeysDeletionStatus": "[{\"backingKeyId\":\"backing-key-id\",
\"deletionStatus\":\"FAILURE\"}]"
    },
    "eventID": "c21f1f47-f52b-4ffe-bff0-6d994403cf40",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "managementEvent": true,
    "eventCategory": "Management"
}
```

> **Hinweise**
>
> Die folgenden Verfahren verwenden das AWS CloudHSM Client SDK 5-Befehlszeilentool CloudHSM CLI. Die CloudHSM-CLI ersetzt durch`key-handle`. `key-reference`
> Am 1. Januar 2025 AWS CloudHSM wird der Support für die Client SDK 3-Befehlszeilentools, das CloudHSM Management Utility (CMU) und das Key Management Utility (KMU) eingestellt. Weitere Informationen zu den Unterschieden zwischen den Client SDK 3-Befehlszeilentools und dem Client SDK 5-Befehlszeilentool finden Sie unter Migrieren von Client SDK 3 CMU und KMU zu Client SDK 5 CloudHSM CLI im Benutzerhandbuch.AWS CloudHSM

Die folgenden Verfahren zeigen, wie das verwaiste Schlüsselmaterial aus dem zugehörigen Cluster gelöscht wird. AWS CloudHSM

1. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher, falls er nicht bereits getrennt ist, und melden Sie sich an, wie unter beschrieben. Trennen und Anmelden

> **Note**
>
> Sämtliche Versuche, KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zu erstellen oder vorhandene KMS-Schlüssel in kryptografischen Produktionen zu nutzen, schlagen fehl, während der benutzerdefinierte Schlüsselspeicher getrennt ist. Diese Aktion kann verhindern, dass Benutzer vertrauliche Daten speichern und darauf zugreifen.

2. Verwenden Sie den Befehl key delete in der CloudHSM-CLI, um den Schlüssel aus dem Cluster HSMs zu löschen.

   Alle CloudTrail Protokolleinträge für kryptografische Operationen mit einem KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher enthalten ein `additionalEventData` Feld mit dem und. `customKeyStoreId` `backingKey` Der im `backingKeyId` Feld zurückgegebene Wert ist das `id` CloudHSM-Schlüsselattribut. Wir empfehlen, den Vorgang zum Löschen von Schlüsseln danach `id` zu filtern, dass das verwaiste Schlüsselmaterial gelöscht wird, das Sie in Ihren Protokollen identifiziert haben. CloudTrail

   AWS CloudHSM erkennt den `backingKeyId` Wert als Hexadezimalwert. Um nach zu filtern`id`, müssen Sie das mit voranstellen. `backingKeyId` `0x` Wenn das beispielsweise `backingKeyId` in Ihrem CloudTrail Protokoll steht`1a2b3c45678abcdef`, würden Sie nach **`0x`**`1a2b3c45678abcdef` filtern.

   Das folgende Beispiel löscht einen Schlüssel aus dem HSMs in Ihrem Cluster. Der `backing-key-id` ist im CloudTrail Protokolleintrag aufgeführt. Bevor Sie diesen Befehl ausführen, ersetzen Sie das Beispiel `backing-key-id` durch ein gültiges Beispiel aus Ihrem Konto.

```
aws-cloudhsm key delete --filter attr.id="0x<backing-key-id>"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. Melden Sie sich ab und stellen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher wieder her, wie unter beschriebenAbmelden und erneutes Verbinden.

Wiederherstellen von gelöschtem Schlüsselmaterial für einen KMS-Schlüssel

Wenn das Schlüsselmaterial für einen gelöscht AWS KMS key wird, ist der KMS-Schlüssel unbrauchbar und der gesamte Chiffretext, der unter dem KMS-Schlüssel verschlüsselt wurde, kann nicht entschlüsselt werden. Dies kann passieren, wenn das Schlüsselmaterial für einen KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher aus dem zugehörigen Cluster gelöscht wird. AWS CloudHSM Es kann jedoch möglich sein, das Schlüsselmaterial wiederherzustellen.

Wenn Sie einen AWS KMS key (KMS-Schlüssel) in einem AWS CloudHSM Schlüsselspeicher erstellen, AWS KMS meldet er sich beim zugehörigen AWS CloudHSM Cluster an und erstellt das Schlüsselmaterial für den KMS-Schlüssel. Außerdem wird das Passwort in einen Wert geändert, den nur er kennt, und er bleibt angemeldet, solange der AWS CloudHSM Schlüsselspeicher verbunden ist. Da nur der Schlüsselbesitzer, d. h. die CU, die einen Schlüssel erstellt hat, den Schlüssel löschen kann, ist es unwahrscheinlich, dass der Schlüssel HSMs versehentlich aus dem Schlüssel gelöscht wird.

Wenn jedoch das Schlüsselmaterial für einen KMS-Schlüssel aus dem HSMs in einem Cluster gelöscht wird, ändert sich der Schlüsselstatus des KMS-Schlüssels irgendwann auf`UNAVAILABLE`. Wenn Sie versuchen, den KMS-Schlüssel für eine kryptographische Operation zu verwenden, schlägt die Operation mit einer `KMSInvalidStateException`-Ausnahme fehl. Wichtig dabei ist, dass die mit dem KMS-Schlüssel verschlüsselten Daten nicht mehr entschlüsselt werden können.

Unter bestimmten Umständen können Sie das gelöschte Schlüsselmaterial wiederherstellen, indem Sie einen Cluster aus einem Backup erstellen, das das Schlüsselmaterial enthält. Diese Strategie funktioniert nur, wenn mindestens eine Sicherung erstellt wurde, während der Schlüssel vorhanden war und bevor er gelöscht wurde.

Gehen Sie wie folgt vor, um das Schlüsselmaterial wiederherzustellen.

1. Suchen Sie ein Cluster-Backup, das das Schlüsselmaterial enthält. Die Sicherung muss dazu alle Benutzer und Schlüssel enthalten, die zur Unterstützung des Clusters und seiner verschlüsselten Daten erforderlich sind.

   Verwenden Sie den DescribeBackupsVorgang, um die Backups für einen Cluster aufzulisten. Verwenden Sie dann den Sicherung-Zeitstempel, um eine Sicherung auszuwählen. Um die Ausgabe auf den Cluster zu beschränken, der dem AWS CloudHSM Schlüsselspeicher zugeordnet ist, verwenden Sie den `Filters` Parameter, wie im folgenden Beispiel gezeigt.

   ```
   $ aws cloudhsmv2 describe-backups --filters clusterIds=<cluster ID>
   ```

```
{
    "Backups": [
        {
            "ClusterId": "cluster-1a23b4cdefg",
            "BackupId": "backup-9g87f6edcba",
            "CreateTimestamp": 1536667238.328,
            "BackupState": "READY"
        },
             ...
    ]
}
```

2. Erstellen Sie einen Cluster aus der ausgewählten Sicherung. Prüfen Sie, ob die Sicherung den gelöschten Schlüssel sowie weitere Benutzer und Schlüssel enthält, die für den Cluster erforderlich sind.
3. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher, damit Sie seine Eigenschaften bearbeiten können.
4. Bearbeiten Sie die Cluster-ID des AWS CloudHSM Schlüsselspeichers. Geben Sie die Cluster-ID des Clusters ein, den Sie aus der Sicherung erstellt haben. Da der Cluster seinen Sicherungsverlauf mit dem ursprünglichen Cluster gemeinsam hat, sollte die neue Cluster-ID gültig sein.
5. Stellen Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher wieder her.

Anmeldung als **`kmsuser`**

AWS KMS Verwendet das `kmsuser`Crypto User (CU) -Konto, um das AWS CloudHSM Schlüsselmaterial im AWS CloudHSM Cluster für Ihren Schlüsselspeicher zu erstellen und zu verwalten. Sie erstellen das `kmsuser` CU-Konto in Ihrem Cluster und geben das zugehörige Passwort an, AWS KMS wenn Sie Ihren AWS CloudHSM Schlüsselspeicher erstellen.

AWS KMS Verwaltet im Allgemeinen das `kmsuser` Konto. Für einige Aufgaben müssen Sie jedoch die Verbindung zum AWS CloudHSM Schlüsselspeicher trennen, sich als `kmsuser` CU beim Cluster anmelden und die CloudHSM-Befehlszeilenschnittstelle (CLI) verwenden.

> **Note**
>
> Sämtliche Versuche, KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zu erstellen oder vorhandene KMS-Schlüssel in kryptografischen Produktionen zu nutzen,

schlagen fehl, während der benutzerdefinierte Schlüsselspeicher getrennt ist. Diese Aktion kann verhindern, dass Benutzer vertrauliche Daten speichern und darauf zugreifen.

In diesem Thema wird erklärt, wie Sie die Verbindung zum AWS CloudHSM Schlüsselspeicher trennen und sich anmeldenkmsuser, das AWS CloudHSM Befehlszeilentool ausführen, sich abmelden und den Schlüsselspeicher erneut verbinden. AWS CloudHSM

Themen

- Trennen und Anmelden
- Abmelden und erneutes Verbinden

Trennen und Anmelden

Gehen Sie jedes Mal wie folgt vor, wenn Sie sich als kmsuser Crypto-Benutzer bei einem zugehörigen Cluster anmelden müssen.

> **Hinweise**
>
> Die folgenden Verfahren verwenden das AWS CloudHSM Client SDK 5-Befehlszeilentool CloudHSM CLI. Die CloudHSM-CLI ersetzt durchkey-handle. key-reference
> Am 1. Januar 2025 AWS CloudHSM wird der Support für die Client SDK 3-Befehlszeilentools, das CloudHSM Management Utility (CMU) und das Key Management Utility (KMU) eingestellt. Weitere Informationen zu den Unterschieden zwischen den Client SDK 3-Befehlszeilentools und dem Client SDK 5-Befehlszeilentool finden Sie unter Migrieren von Client SDK 3 CMU und KMU zu Client SDK 5 CloudHSM CLI im Benutzerhandbuch.AWS CloudHSM

1. Trennen Sie die Verbindung AWS CloudHSM zum Schlüsselspeicher, falls er nicht bereits getrennt ist. Sie können die AWS KMS Konsole oder AWS KMS API verwenden.

   Solange Ihr AWS CloudHSM Schlüssel verbunden ist, AWS KMS ist er angemeldet alskmsuser. Dadurch wird verhindert, dass Sie sich als kmsuser anmelden oder das kmsuser-Passwort ändern.

Dieser Befehl dient beispielsweise dazu, DisconnectCustomKeyStoredie Verbindung zu einem Beispiel-Schlüsselspeicher zu trennen. Ersetzen Sie die AWS CloudHSM Beispiel-Schlüsselspeicher-ID durch eine gültige.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

2. Verwenden Sie den Login-Befehl, um sich als Administrator anzumelden. Verwenden Sie die im Abschnitt „CloudHSM CLI verwenden“ des AWS CloudHSM Benutzerhandbuchs beschriebenen Verfahren.

```
aws-cloudhsm > login --username admin --role admin
          Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

3. Verwenden Sie den Befehl user change-password in der CloudHSM-CLI, um das Passwort des `kmsuser` Kontos in ein Ihnen bekanntes zu ändern. (wechselt AWS KMS das Passwort, wenn Sie Ihren Schlüsselspeicher verbinden.) AWS CloudHSM Das Passwort muss 7 – 32 alphanumerische Zeichen umfassen. Bei der Angabe wird zwischen Groß- und Kleinschreibung unterschieden, Sonderzeichen sind nicht zulässig.

4. Melden Sie sich `kmsuser` mit dem von Ihnen festgelegten Passwort an. Eine ausführliche Anleitung finden Sie im Abschnitt „CloudHSM CLI verwenden“ des AWS CloudHSM Benutzerhandbuchs.

```
aws-cloudhsm > login --username kmsuser --role crypto-user
          Enter password:
{
  "error_code": 0,
  "data": {
    "username": "kmsuser",
    "role": "crypto-user"
  }
}
```

Abmelden und erneutes Verbinden

Gehen Sie jedes Mal wie folgt vor, wenn Sie sich als `kmsuser` Crypto-Benutzer abmelden und die Verbindung zu Ihrem Schlüsselspeicher erneut herstellen müssen.

> **Hinweise**
>
> Die folgenden Verfahren verwenden das AWS CloudHSM Client SDK 5-Befehlszeilentool CloudHSM CLI. Die CloudHSM-CLI ersetzt durch`key-handle`. `key-reference`
> Am 1. Januar 2025 AWS CloudHSM wird der Support für die Client SDK 3-Befehlszeilentools, das CloudHSM Management Utility (CMU) und das Key Management Utility (KMU) eingestellt. Weitere Informationen zu den Unterschieden zwischen den Client SDK 3-Befehlszeilentools und dem Client SDK 5-Befehlszeilentool finden Sie unter Migrieren von Client SDK 3 CMU und KMU zu Client SDK 5 CloudHSM CLI im Benutzerhandbuch.AWS CloudHSM

1. Führen Sie die Aufgabe aus und melden Sie sich dann mit dem Befehl logout in der CloudHSM-CLI ab. Wenn Sie sich nicht abmelden, schlagen Versuche fehl, die Verbindung zu Ihrem AWS CloudHSM Schlüsselspeicher wiederherzustellen.

   ```
   aws-cloudhsm  logout
   {
     "error_code": 0,
     "data": "Logout successful"
   }
   ```

2. Bearbeiten Sie die `kmsuser`-Passworteinstellung für den Custom Key Store.

   Dies gibt AWS KMS das aktuelle Passwort für den Cluster `kmsuser` an. Wenn Sie diesen Schritt auslassen, können AWS KMS Sie sich nicht als beim Cluster anmeldenkmsuser, und alle Versuche, die Verbindung zu Ihrem benutzerdefinierten Schlüsselspeicher wiederherzustellen, schlagen fehl. Sie können die AWS KMS Konsole oder den `KeyStorePassword` Parameter des Vorgangs verwenden. UpdateCustomKeyStore

   Dieser Befehl teilt beispielsweise mit AWS KMS , dass das aktuelle Passwort lautet`tempPassword`. Ersetzen Sie das Beispielpasswort durch das tatsächliche Passwort.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --
key-store-password tempPassword
```

3. Verbinden Sie den AWS KMS Schlüsselspeicher erneut mit seinem AWS CloudHSM Cluster. Ersetzen Sie die AWS CloudHSM Beispiel-Schlüsselspeicher-ID durch eine gültige. AWS KMS Ändert das `kmsuser` Passwort während des Verbindungsvorgangs in einen Wert, den nur das Unternehmen kennt.

   Der ConnectCustomKeyStoreVorgang kehrt schnell zurück, der Verbindungsvorgang kann jedoch längere Zeit in Anspruch nehmen. Die erste Reaktion ist kein Zeichen für den Erfolg des Verbindungsvorgangs.

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

4. Verwenden Sie den DescribeCustomKeyStoresVorgang, um zu überprüfen, ob der AWS CloudHSM Schlüsselspeicher verbunden ist. Ersetzen Sie die AWS CloudHSM Beispiel-Schlüsselspeicher-ID durch eine gültige.

   In diesem Beispiel zeigt das Feld für den Verbindungsstatus, dass der AWS CloudHSM Schlüsselspeicher jetzt verbunden ist.

```
$ aws kms describe-custom-key-stores --custom-key-store-
id cks-1234567890abcdef0
{
   "CustomKeyStores": [
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "CONNECTED"
   ],
}
```

Externe Schlüsselspeicher

Externe Schlüsselspeicher ermöglichen es Ihnen, Ihre AWS Ressourcen mithilfe kryptografischer Schlüssel außerhalb von AWS zu schützen. Dieses erweiterte Feature wurde für regulierte

Workloads entwickelt, die Sie mit Verschlüsselungsschlüsseln schützen müssen, die in einem externen System für die Schlüsselverwaltung gespeichert sind, das Sie kontrollieren. Externe Schlüsselspeicher unterstützen das Versprechen der AWS digitalen Souveränität und geben Ihnen die souveräne Kontrolle über Ihre Daten AWS, einschließlich der Möglichkeit, mit Schlüsselmaterial zu verschlüsseln, das Ihnen gehört und das Sie außerhalb kontrollieren. AWS

Ein externer Schlüsselspeicher ist ein benutzerdefinierter Schlüsselspeicher, der von einem externen Schlüsselmanager unterstützt wird, den Sie besitzen und den Sie außerhalb verwalten. AWS Ihr externer Schlüsselmanager kann ein physisches oder virtuelles Hardware-Sicherheitsmodul (HSMs) oder ein beliebiges hardware- oder softwarebasiertes System sein, das kryptografische Schlüssel generieren und verwenden kann. Verschlüsselungs- und Entschlüsselungsvorgänge, bei denen ein KMS-Schlüssel in einem externen Schlüsselspeicher verwendet wird, werden von Ihrem externen Schlüsselmanager unter Verwendung Ihres kryptografischen Schlüsselmaterials ausgeführt. Diese Funktion wird als Hold Your Own Keys () bezeichnet. HYOKs

AWS KMS interagiert nie direkt mit Ihrem externen Schlüsselmanager und kann Ihre Schlüssel nicht erstellen, anzeigen, verwalten oder löschen. AWS KMS Interagiert stattdessen nur mit der externen Schlüsselspeicher-Proxy-Software (XKS-Proxy), die Sie bereitstellen. Ihr externer Schlüsselspeicher-Proxy vermittelt die gesamte Kommunikation zwischen AWS KMS und Ihrem externen Schlüsselmanager. Er überträgt alle Anfragen von AWS KMS Ihrem externen Schlüsselmanager und überträgt die Antworten von Ihrem externen Schlüsselmanager zurück an AWS KMS. Der externe Schlüsselspeicher-Proxy übersetzt auch generische Anfragen AWS KMS in ein herstellerspezifisches Format, das Ihr externer Schlüsselmanager verstehen kann, sodass Sie externe Schlüsselspeicher mit Schlüsselmanagern verschiedener Anbieter verwenden können.

Sie können KMS-Schlüssel in einem externen Schlüsselspeicher für die clientseitige Verschlüsselung verwenden, auch mit dem AWS Encryption SDK. Externe Schlüsselspeicher sind jedoch eine wichtige Ressource für die serverseitige Verschlüsselung, sodass Sie Ihre AWS Ressourcen mehrfach AWS-Services mit Ihren kryptografischen Schlüsseln außerhalb von schützen können. AWS AWS-Services die vom Kunden verwaltete Schlüssel für symmetrische Verschlüsselung unterstützen, unterstützen auch KMS-Schlüssel in einem externen Schlüsselspeicher. Einzelheiten zum Servicesupport finden Sie unter AWS -Serviceintegration.

Externe Schlüsselspeicher ermöglichen die Verwendung AWS KMS für regulierte Workloads, bei denen Verschlüsselungsschlüssel gespeichert und außerhalb von verwendet werden müssen. AWS Sie weichen jedoch erheblich vom Standardmodell der übergreifenden Verantwortlichkeit ab und sind mit zusätzlichen operativen Belastungen verbunden. Das höhere Risiko für Verfügbarkeit und Latenz

wird für die meisten Kunden die wahrgenommenen Sicherheitsvorteile externer Schlüsselspeicher übersteigen.

Mit externen Schlüsselspeichern haben Sie die Kontrolle über den Vertrauensanker. Daten, die mit KMS-Schlüsseln in Ihrem externen Schlüsselspeicher verschlüsselt wurden, können nur mithilfe des von Ihnen kontrollierten externen Schlüsselmanagers entschlüsselt werden. Wenn Sie den Zugriff auf Ihren externen Schlüsselmanager vorübergehend entziehen, z. B. indem Sie die Verbindung zum externen Schlüsselspeicher trennen oder Ihren externen Schlüsselmanager vom externen Schlüsselspeicher-Proxy trennen, AWS geht jeglicher Zugriff auf Ihre kryptografischen Schlüssel verloren, bis Sie ihn wiederherstellen. Während dieses Intervalls kann der unter Ihren KMS-Schlüsseln verschlüsselte Geheimtext nicht entschlüsselt werden. Wenn Sie den Zugriff auf Ihren externen Schlüsselmanager dauerhaft widerrufen, kann der gesamte unter einem KMS-Schlüssel verschlüsselte Geheimtext in Ihrem externen Schlüsselspeicher nicht wiederhergestellt werden. Die einzigen Ausnahmen sind AWS Dienste, die die durch Ihre KMS-Schlüssel geschützten Datenschlüssel kurzzeitig zwischenspeichern. Diese Datenschlüssel funktionieren so lange, bis Sie die Ressource deaktivieren oder der Cache abläuft. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

Externe Schlüsselspeicher schließen die wenigen Anwendungsfälle für regulierte Workloads aus, bei denen Verschlüsselungsschlüssel ausschließlich unter Ihrer Kontrolle bleiben müssen und für die Sie keinen Zugriff haben müssen. AWS Dies ist jedoch eine große Änderung in der Art und Weise, wie Sie eine cloudbasierte Infrastruktur betreiben, und eine bedeutende Änderung des Modells der übergreifenden Verantwortlichkeit. Bei den meisten Workloads werden der zusätzliche Betriebsaufwand und die höheren Risiken für Verfügbarkeit und Leistung die wahrgenommenen Sicherheitsvorteile externer Schlüsselspeicher übersteigen.

Benötige ich einen externen Schlüsselspeicher?

Für die meisten Benutzer erfüllt der AWS KMS Standard-Schlüsselspeicher, der durch FIPS 140-3 Security Level 3-validierte Hardware-Sicherheitsmodule geschützt ist, ihre Sicherheits-, Kontroll- und behördlichen Anforderungen. Externe Benutzer von Schlüsselspeichern haben einen erheblichen Aufwand an Kosten, Wartung und Fehlerbehebung sowie Risiken in Bezug auf Latenz, Verfügbarkeit und Zuverlässigkeit.

Wenn Sie einen externen Schlüsselspeicher in Betracht ziehen, nehmen Sie sich etwas Zeit, um sich mit den Alternativen vertraut zu machen. Dazu gehören ein AWS CloudHSM Schlüsselspeicher, der von einem AWS CloudHSM Cluster unterstützt wird, den Sie besitzen und verwalten, und KMS-

Schlüssel mit importiertem Schlüsselmaterial, das Sie selbst generieren HSMs und bei Bedarf aus KMS-Schlüsseln löschen können. Insbesondere der Import von Schlüsselmaterial mit einem sehr kurzen Verfallsintervall könnte ein ähnliches Maß an Kontrolle bieten, ohne die Leistungs- oder Verfügbarkeitsrisiken einzugehen.

Ein externer Schlüsselspeicher könnte die richtige Lösung für Ihr Unternehmen sein, wenn Sie die folgenden Anforderungen erfüllen:

- Sie müssen kryptografische Schlüssel in Ihrem lokalen Schlüsselmanager oder in einem Schlüsselmanager verwenden AWS , den Sie nicht kontrollieren.
- Sie müssen nachweisen, dass Ihre kryptografischen Schlüssel außerhalb der Cloud ausschließlich unter Ihrer Kontrolle aufbewahrt werden.
- Sie müssen mithilfe kryptografischer Schlüssel mit unabhängiger Autorisierung verschlüsseln und entschlüsseln.
- Für Schlüsselmaterial muss ein unabhängiger sekundärer Prüfungspfad eingerichtet sein.

Wenn Sie sich für einen externen Schlüsselspeicher entscheiden, beschränken Sie dessen Verwendung auf Workloads, die außerhalb von AWS mit kryptografischen Schlüsseln geschützt werden müssen.

Modell der geteilten Verantwortung

Standardmäßige KMS-Schlüssel verwenden Schlüsselmaterial, das in dem generiert und verwendet wird HSMs , der AWS KMS Eigentümer und Verwalter ist. Sie legen die Zugriffskontrollrichtlinien für Ihre KMS-Schlüssel fest und konfigurieren AWS-Services , dass KMS-Schlüssel zum Schutz Ihrer Ressourcen verwendet werden. AWS KMS übernimmt die Verantwortung für die Sicherheit, Verfügbarkeit, Latenz und Haltbarkeit des Schlüsselmaterials in Ihren KMS-Schlüsseln.

KMS-Schlüssel in externen Schlüsselspeichern hängen von wichtigen Materialien und Vorgängen in Ihrem externen Schlüsselmanager ab. Somit verschiebt sich das Gleichgewicht der Verantwortung in Ihre Richtung. Sie sind für die Sicherheit, Zuverlässigkeit, Haltbarkeit und Leistung der kryptografischen Schlüssel in Ihrem externen Schlüsselmanager verantwortlich. AWS KMS ist dafür verantwortlich, umgehend auf Anfragen zu antworten und mit Ihrem externen Schlüsselspeicher-Proxy zu kommunizieren und unsere Sicherheitsstandards aufrechtzuerhalten. Um sicherzustellen, dass jeder externe Schlüssel den Chiffretext mindestens so stark speichert wie der AWS KMS Standard-Chiffretext, verschlüsselt er AWS KMS zunächst den gesamten Klartext mit AWS

KMS Schlüsselmaterial, das für Ihren KMS-Schlüssel spezifisch ist, und sendet ihn dann zur Verschlüsselung mit Ihrem externen Schlüssel an Ihren externen Schlüsselmanager. Dieses Verfahren wird als doppelte Verschlüsselung bezeichnet. Infolgedessen können weder AWS KMS noch der/die Besitzer:in des externen Schlüsselmaterials doppelt verschlüsselten Geheimtext alleine entschlüsseln.

Sie sind verantwortlich für die Pflege eines externen Schlüsselmanagers, der Ihren gesetzlichen und Leistungsstandards entspricht, für die Bereitstellung und Pflege eines externen Schlüsselspeicher-Proxys, der der API-Spezifikation für externe Schlüsselspeicher-Proxys von AWS KMS entspricht, und für die Gewährleistung der Verfügbarkeit und Beständigkeit Ihres Schlüsselmaterials. Sie müssen auch einen externen Schlüsselspeicher erstellen, konfigurieren und verwalten. Wenn Fehler auftreten, die durch Komponenten verursacht werden, die Sie warten, müssen Sie darauf vorbereitet sein, die Fehler zu identifizieren und zu beheben, damit die AWS Dienste ohne übermäßige Unterbrechung auf Ihre Ressourcen zugreifen können. AWS KMS bietet Anleitungen zur Problembehebung, mit deren Hilfe Sie die Ursache von Problemen und die wahrscheinlichsten Lösungen ermitteln können.

Überprüfen Sie die CloudWatch Amazon-Metriken und -Dimensionen, die für externe Schlüsselgeschäfte AWS KMS aufgezeichnet wurden. AWS KMS empfiehlt dringend, CloudWatch Alarme zur Überwachung Ihres externen Schlüsselspeichers einzurichten, damit Sie die ersten Anzeichen von Leistungs- und Betriebsproblemen erkennen können, bevor sie auftreten.

Was ändert sich?

Externe Schlüsselspeicher unterstützen nur KMS-Schlüssel mit symmetrischer Verschlüsselung. Innerhalb AWS KMS verwenden und verwalten Sie KMS-Schlüssel in einem externen Schlüsselspeicher auf die gleiche Weise wie andere vom Kunden verwaltete Schlüssel, einschließlich der Festlegung von Zugriffskontrollrichtlinien und der Überwachung der Schlüsselverwendung. Sie verwenden dasselbe APIs mit denselben Parametern, um eine kryptografische Operation mit einem KMS-Schlüssel in einem externen Schlüsselspeicher anzufordern, den Sie für einen beliebigen KMS-Schlüssel verwenden. Der Preis ist derselbe wie für die Standard-KMS-Schlüssel. Einzelheiten finden Sie unter KMS-Schlüssel in externen Schlüsselspeichern und AWS Key Management Service Preise.

Bei externen Schlüsselspeichern ändern sich jedoch die folgenden Prinzipien:

- Sie sind für die Verfügbarkeit, Beständigkeit und Latenz wichtiger Vorgänge verantwortlich.
- Sie sind für alle Kosten für die Entwicklung, den Erwerb, den Betrieb und die Lizenzierung Ihres externen Schlüsselmanagersystems verantwortlich.

- Sie können die unabhängige Autorisierung aller Anfragen AWS KMS an Ihren externen Schlüsselspeicher-Proxy implementieren.
- Sie können alle Vorgänge Ihres externen Schlüsselspeicher-Proxys sowie alle Vorgänge Ihres externen Schlüsselmanagers im Zusammenhang mit AWS KMS Anfragen überwachen, prüfen und protokollieren.

Wo fange ich an?

Um einen externen Schlüsselspeicher zu erstellen und zu verwalten, müssen Sie die Proxy-Konnektivitätsoption für Ihren externen Schlüsselspeicher auswählen, die Voraussetzungen erfüllen und Ihren externen Schlüsselspeicher erstellen und konfigurieren.

Kontingente

AWS KMS ermöglicht bis zu 10 benutzerdefinierte Schlüsselspeicher in jeder AWS-Konto Region, einschließlich sowohl AWS CloudHSM Schlüsselspeicher als auch externer Schlüsselspeicher, unabhängig von deren Verbindungsstatus. Darüber hinaus gibt es AWS KMS - Anforderungskontingente für die Nutzung von KMS-Schlüsseln in einem externen Schlüsselspeicher.

Wenn Sie VPC-Proxykonnektivität für Ihren externen Schlüsselspeicher-Proxy wählen, gibt es möglicherweise auch Kontingente für die erforderlichen Komponenten VPCs, wie Subnetze und Netzwerk-Loadbalancer. Informationen über diese Kontingente erhalten Sie in der Service Quotas-Konsole.

Regionen

Um die Netzwerklatenz zu minimieren, erstellen Sie Ihre externen Schlüsselspeicherkomponenten in der AWS-Region , die Ihrem externen Schlüsselmanager am nächsten liegt. Wählen Sie nach Möglichkeit eine Region mit einer Netzwerk-Round-Trip-Zeit (RTT) von maximal 35 Millisekunden.

Externe Schlüsselspeicher werden AWS-Regionen in allen Bereichen unterstützt, AWS KMS mit Ausnahme von China (Peking) und China (Ningxia).

Nicht unterstützte Funktionen

AWS KMS unterstützt die folgenden Funktionen in benutzerdefinierten Schlüsselspeichern nicht.

- Asymmetrische KMS-Schlüssel
- HMAC-KMS-Schlüssel

- KMS-Schlüssel mit importiertem Schlüsselmaterial
- Automatische Schlüsselrotation
- Multiregionale Schlüssel

Weitere Informationen:

- Ankündigung des AWS KMS External Key Store im AWS News-Blog.

Konzepte für externe Schlüsselspeicher

Lernen Sie die grundlegenden Begriffe und Konzepte kennen, die in externen Schlüsselspeichern verwendet werden.

Externer Schlüsselspeicher

Ein externer Schlüsselspeicher ist ein AWS KMS benutzerdefinierter Schlüsselspeicher, der von einem externen Schlüsselmanager unterstützt wird AWS , der nicht Ihnen gehört und von dem Sie verwaltet werden. Jeder KMS-Schlüssel in einem externen Schlüsselspeicher ist mit einem externen Schlüssel in Ihrem externen Schlüsselmanager verknüpft. Wenn Sie einen KMS-Schlüssel in einem externen Schlüsselspeicher für die Verschlüsselung oder Entschlüsselung verwenden, wird der Vorgang in Ihrem externen Schlüsselmanager unter Verwendung Ihres externen Schlüssels ausgeführt. Diese Anordnung wird als Hold Your Own Keys (HYOK) bezeichnet. Dieses Feature wurde für Organisationen entwickelt, die ihre kryptografischen Schlüssel in ihrem eigenen externen Schlüsselmanager verwalten müssen.

Externe Schlüsselspeicher stellen sicher, dass die kryptografischen Schlüssel und Operationen, die Ihre AWS Ressourcen schützen, in Ihrem externen Schlüsselmanager unter Ihrer Kontrolle verbleiben. AWS KMS sendet Anfragen an Ihren externen Schlüsselmanager zum Verschlüsseln und Entschlüsseln von Daten, AWS KMS kann jedoch keine externen Schlüssel erstellen, löschen oder verwalten. Alle Anfragen AWS KMS an Ihren externen Schlüsselmanager werden über eine externe Schlüsselspeicher-Proxy-Softwarekomponente vermittelt, die Sie bereitstellen, besitzen und verwalten.

AWS Dienste, die vom AWS KMS Kunden verwaltete Schlüssel unterstützen, können die KMS-Schlüssel in Ihrem externen Schlüsselspeicher verwenden, um Ihre Daten zu schützen. Infolgedessen werden Ihre Daten letztendlich durch Ihre Schlüssel geschützt, indem Sie Ihre Verschlüsselungsvorgänge in Ihrem externen Schlüsselmanager verwenden.

Für die KMS-Schlüssel in einem externen Schlüsselspeicher gelten grundsätzlich andere Vertrauensmodelle, Regelungen zur gemeinsamen Verantwortung und Leistungserwartungen als für Standard-KMS-Schlüssel. Bei externen Schlüsselspeichern sind Sie für die Sicherheit und Integrität des Schlüsselmaterials und der kryptografischen Operationen verantwortlich. Die Verfügbarkeit und Latenz von KMS-Schlüsseln in einem externen Schlüsselspeicher werden durch die Hardware, Software, Netzwerkkomponenten und die Entfernung zwischen AWS KMS und Ihrem externen Schlüsselmanager beeinflusst. Außerdem werden Ihnen wahrscheinlich zusätzliche Kosten für Ihren externen Schlüsselmanager und für die Netzwerk- und Lastenausgleichsinfrastruktur entstehen, mit der Ihr externer Schlüsselmanager kommunizieren muss AWS KMS

Sie können Ihren externen Schlüsselspeicher als Teil Ihrer umfassenderen Datenschutzstrategie verwenden. Für jede AWS Ressource, die Sie schützen, können Sie entscheiden, welche einen KMS-Schlüssel in einem externen Schlüsselspeicher erfordern und welche durch einen Standard-KMS-Schlüssel geschützt werden können. Dies gibt Ihnen die Flexibilität, KMS-Schlüssel für bestimmte Datenklassifizierungen, Anwendungen oder Projekte auszuwählen.

Externer Schlüsselmanager

Ein externer Schlüsselmanager ist eine Komponente außerhalb von AWS , die symmetrische 256-Bit-AES-Schlüssel generieren und eine symmetrische Verschlüsselung und Entschlüsselung durchführen kann. Der externe Schlüsselmanager für einen externen Schlüsselspeicher kann ein physisches Hardwaresicherheitsmodul (HSM), ein virtuelles HSM oder ein Software-Schlüsselmanager mit oder ohne HSM-Komponente sein. Es kann sich an einem beliebigen Ort außerhalb von befinden AWS, auch bei Ihnen vor Ort, in einem lokalen oder entfernten Rechenzentrum oder in einer beliebigen Cloud. Ihr externer Schlüsselspeicher kann von einem einzelnen externen Schlüsselmanager oder mehreren zugehörigen Schlüsselmanager-Instances unterstützt werden, die kryptografische Schlüssel gemeinsam nutzen, z. B. einen HSM-Cluster. Externe Schlüsselspeicher sind so konzipiert, dass sie eine Vielzahl von externen Managern verschiedener Anbieter unterstützen. Einzelheiten zum Herstellen einer Verbindung zu Ihrem externen Schlüsselmanager finden Sie unterWählen Sie eine Proxy-Konnektivitätsoption für externen Schlüsselspeicher.

Externer Schlüssel

Jeder KMS-Schlüssel in einem externen Schlüsselspeicher ist mit einem kryptografischen Schlüssel in Ihrem externen Schlüsselmanager verknüpft, der als externer Schlüssel bezeichnet wird. Wenn Sie mit einem KMS-Schlüssel in Ihrem externen Schlüsselspeicher verschlüsseln oder entschlüsseln, wird die kryptografische Operation in Ihrem externen Schlüsselmanager unter Verwendung Ihres externen Schlüssels durchgeführt.

> **Warning**
>
> Der externe Schlüssel ist für den Betrieb des KMS-Schlüssels unerlässlich. Wenn der externe Schlüssel verloren geht oder gelöscht wird, kann der unter dem zugehörigen KMS-Schlüssel verschlüsselte Geheimtext nicht wiederhergestellt werden.

Für externe Schlüsselspeicher muss es sich bei einem externen Schlüssel um einen 256-Bit-AES-Schlüssel handeln, der aktiviert ist und Verschlüsselung und Entschlüsselung durchführen kann. Ausführliche Informationen zu den Anforderungen an externe Schlüssel finden Sie unter Anforderungen an einen KMS-Schlüssel in einem externen Schlüsselspeicher.

AWS KMS kann keine externen Schlüssel erstellen, löschen oder verwalten. Ihr kryptografisches Schlüsselmaterial verlässt niemals Ihren externen Schlüsselmanager. Wenn Sie einen KMS-Schlüssel in einem externen Schlüsselspeicher erstellen, geben Sie die ID eines externen Schlüssels (`XksKeyId`) an. Sie können die externe Schlüssel-ID, die einem KMS-Schlüssel zugeordnet ist, nicht ändern, obwohl Ihr externer Schlüsselmanager das der externen Schlüssel-ID zugeordnete Schlüsselmaterial rotieren kann.

Zusätzlich zu Ihrem externen Schlüssel enthält ein KMS-Schlüssel in einem externen Schlüsselspeicher auch AWS KMS -Schlüsselmaterial. Durch den KMS-Schlüssel geschützte Daten werden zuerst AWS KMS mithilfe des AWS KMS Schlüsselmaterials und dann von Ihrem externen Schlüsselmanager mithilfe Ihres externen Schlüssels verschlüsselt. Dieser Prozess der doppelten Verschlüsselung stellt sicher, dass der durch Ihren KMS-Schlüssel geschützte Geheimtext immer mindestens so stark ist wie der nur durch AWS KMS geschützte Geheimtext.

Viele kryptografische Schlüssel haben unterschiedliche Arten von Identifikatoren. Wenn Sie einen KMS-Schlüssel in einem externen Schlüsselspeicher erstellen, geben Sie die ID des externen Schlüssels an, den der externe Schlüsselspeicher-Proxy verwendet, um auf den externen Schlüssel zu verweisen. Wenn Sie die falsche Kennung verwenden, schlägt Ihr Versuch, einen KMS-Schlüssel in Ihrem externen Schlüsselspeicher zu erstellen, fehl.

Externer Schlüsselspeicher-Proxy

Der externe Schlüsselspeicher-Proxy („XKS-Proxy“) ist eine kundeneigene und vom Kunden verwaltete Softwareanwendung, die die gesamte Kommunikation zwischen AWS KMS und Ihrem externen Schlüsselmanager vermittelt. Außerdem übersetzt er generische AWS KMS Anfragen in ein Format, das Ihr herstellerspezifischer externer Schlüsselmanager versteht. Für einen

externen Schlüsselspeicher ist ein externer Schlüsselspeicher-Proxy erforderlich. Jeder externe Schlüsselspeicher ist mit einem externen Schlüsselspeicher-Proxy verbunden.

AWS KMS kann keine externen Schlüssel erstellen, löschen oder verwalten. Ihr kryptographisches Schlüsselmaterial verlässt zu keiner Zeit Ihren externen Schlüsselmanager. Die gesamte Kommunikation zwischen AWS KMS und Ihrem externen Schlüsselmanager wird über Ihren externen Schlüsselspeicher-Proxy vermittelt. AWS KMS sendet Anfragen an den externen Schlüsselspeicher-Proxy und empfängt Antworten vom externen Schlüsselspeicher-Proxy. Der externe Schlüsselspeicher-Proxy ist dafür verantwortlich AWS KMS , Anfragen von Ihrem externen Schlüsselmanager und Antworten von Ihrem externen Schlüsselmanager zurück an AWS KMS

Sie besitzen und verwalten den externen Schlüsselspeicher-Proxy für Ihren externen Schlüsselspeicher und sind für dessen Wartung und Betrieb verantwortlich. Sie können Ihren externen Schlüsselspeicher-Proxy auf der Grundlage der Open-Source-Proxy-API-Spezifikation für externe Schlüsselspeicher entwickeln, mit der eine Proxyanwendung von einem Anbieter AWS KMS veröffentlicht oder gekauft wird. Ihr externer Schlüsselspeicher-Proxy ist möglicherweise in Ihrem externen Schlüsselmanager enthalten. Zur Unterstützung der Proxyentwicklung werden AWS KMS außerdem ein Beispiel für einen externen Schlüsselspeicher-Proxy (aws-kms-xks-proxy) und ein Testclient (xks-kms-xksproxy-test-client) bereitgestellt, der überprüft, ob Ihr externer Schlüsselspeicher-Proxy der Spezifikation entspricht.

Zur Authentifizierung verwendet der Proxy AWS KMS serverseitige TLS-Zertifikate. Um sich bei Ihrem Proxy zu authentifizieren, AWS KMS signieren Sie alle Anfragen an Ihren externen Schlüsselspeicher-Proxy mit einem SigV4-Proxy-Authentifizierungsnachweis. Optional kann Ihr Proxy Mutual TLS (mTLS) aktivieren, um zusätzlich sicherzustellen, dass er nur Anfragen von akzeptiert. AWS KMS

Ihr externer Schlüsselspeicher-Proxy muss HTTP/1.1 oder höher und TLS 1.2 oder höher mit mindestens einer der folgenden Verschlüsselungssammlungen unterstützen:

- TLS_AES_256_GCM_ SHA384 (TLS 1.3)
- TLS_0_ 05_ CHACHA2 (POLY13TLS SHA256 1.3)

> **Note**
>
> Das unterstützt AWS GovCloud (US) Region TLS_ 0_ 05_ nicht. CHACHA2 POLY13 SHA256

- TLS_ECDHE_RSA_WITH_AES_256_GCM_ (TLS 1.2SHA384)
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_ (TLS 1.2SHA384)

Verbinden Sie den externen Schlüsselspeicher zunächst mit dem externen Schlüsselspeicher-Proxy, um die KMS-Schlüssel in Ihrem externen Schlüsselspeicher zu erstellen und zu verwenden. Sie können Ihren externen Schlüsselspeicher bei Bedarf auch von seinem Proxy trennen. Wenn Sie dies tun, sind alle KMS-Schlüssel im externen Schlüsselspeicher nicht mehr verfügbar. Sie können nicht für kryptografische Operationen verwendet werden.

Konnektivität des externen Schlüsselspeicher-Proxys

Die externe Schlüsselspeicher-Proxykonnektivität („XKS-Proxykonnektivität“) beschreibt die Methode, die für die Kommunikation mit Ihrem externen Schlüsselspeicher-Proxy verwendet wird. AWS KMS

Sie geben Ihre Proxy-Konnektivitätsoption an, wenn Sie Ihren externen Schlüsselspeicher erstellen, und sie wird zu einer Eigenschaft des externen Schlüsselspeichers. Sie können Ihre Proxy-Konnektivitätsoption ändern, indem Sie die benutzerdefinierte Schlüsselspeichereigenschaft aktualisieren. Sie müssen jedoch sicherstellen, dass Ihr externer Schlüsselspeicher-Proxy weiterhin auf dieselben externen Schlüssel zugreifen kann.

AWS KMS unterstützt die folgenden Konnektivitätsoptionen:

- Öffentliche Endpunktkonnektivität — AWS KMS sendet Anfragen für Ihren externen Schlüsselspeicher-Proxy über das Internet an einen öffentlichen Endpunkt, den Sie kontrollieren. Diese Option ist einfach zu erstellen und zu verwalten, erfüllt jedoch möglicherweise nicht die Sicherheitsanforderungen für jede Installation.
- VPC-Endpunktservice-Konnektivität — AWS KMS sendet Anfragen an einen Amazon Virtual Private Cloud (Amazon VPC) -Endpunktservice, den Sie erstellen und verwalten. Sie können Ihren externen Schlüsselspeicher-Proxy in Ihrer Amazon VPC hosten oder Ihren externen Schlüsselspeicher-Proxy außerhalb hosten AWS und die Amazon VPC nur für die Kommunikation verwenden.

Einzelheiten zu den Verbindungsoptionen für den externen Schlüsselspeicher-Proxy finden Sie unter Wählen Sie eine Proxy-Konnektivitätsoption für externen Schlüsselspeicher.

Anmeldeinformation für die Proxy-Authentifizierung des externen Schlüsselspeichers

Um sich bei Ihrem externen Schlüsselspeicher-Proxy zu authentifizieren, AWS KMS signieren Sie alle Anfragen an Ihren externen Schlüsselspeicher-Proxy mit einem Signature V4 (SigV4) - Authentifizierungsnachweis. Sie erstellen und verwalten die Authentifizierungsdaten auf Ihrem Proxy und geben diese Anmeldeinformationen dann an, AWS KMS wenn Sie Ihren externen Speicher erstellen.

> **Note**
>
> Die SigV4-Anmeldeinformationen, die zum Signieren von Anfragen an den XKS-Proxy AWS KMS verwendet werden, haben nichts mit irgendwelchen SigV4-Anmeldeinformationen zu den Prinzipalen in Ihrem zu tun. AWS Identity and Access Management AWS-Konten Verwenden Sie keine IAM-SigV4-Anmeldeinformationen für Ihren externen Schlüsselspeicher-Proxy wieder.

Alle Anmeldeinformationen für die Proxy-Authentifizierung bestehen aus zwei Teilen. Sie müssen beide Teile angeben, wenn Sie einen externen Schlüsselspeicher erstellen oder die Anmeldeinformationen für die Authentifizierung für Ihren externen Schlüsselspeicher aktualisieren.

- Zugriffsschlüssel-ID: Identifiziert den geheimen Zugriffsschlüssel. Sie können diese ID in Klartext angeben.
- Geheimer Zugriffsschlüssel: Der geheime Teil der Anmeldeinformationen. AWS KMS verschlüsselt den geheimen Zugriffsschlüssel in den Anmeldeinformationen, bevor er gespeichert wird.

Sie können die Einstellung für die Anmeldeinformation jederzeit bearbeiten, z. B. wenn Sie falsche Werte eingeben, wenn Sie Ihre Anmeldeinformation auf dem Proxy ändern oder wenn Ihr Proxy die Anmeldeinformation rotiert. Technische Informationen zur AWS KMS Authentifizierung beim externen Schlüsselspeicher-Proxy finden Sie unter Authentifizierung in der API-Spezifikation für den AWS KMS externen Schlüsselspeicher-Proxy.

Damit Sie Ihre Anmeldeinformationen rotieren können, ohne die AWS-Services Verwendung von KMS-Schlüsseln in Ihrem externen Schlüsselspeicher zu unterbrechen, empfehlen wir, dass der externe Schlüsselspeicher-Proxy mindestens zwei gültige Authentifizierungsinformationen für unterstützt. AWS KMS Auf diese Weise wird sichergestellt, dass Ihre vorherige Anmeldeinformation weiterhin funktioniert, während Sie Ihre neue Anmeldeinformation für AWS KMS bereitstellen.

Damit Sie das Alter Ihrer Proxyauthentifizierungsdaten verfolgen können, AWS KMS definiert eine CloudWatch Amazon-Metrik, XksProxyCredentialAge. Sie können diese Metrik verwenden, um einen CloudWatch Alarm auszulösen, der Sie benachrichtigt, wenn das Alter Ihrer Anmeldeinformationen einen von Ihnen festgelegten Schwellenwert erreicht.

Um zusätzlich sicherzustellen, dass Ihr externer Schlüsselspeicher-Proxy nur auf AWS KMS antwortet, unterstützen einige externe Schlüssel-Proxys mTLS (mutual Transport Layer Security). Details hierzu finden Sie unter mTLS-Authentifizierung (optional).

Proxy APIs

Um einen AWS KMS externen Schlüsselspeicher zu unterstützen, muss ein externer Schlüsselspeicher-Proxy den erforderlichen Proxy implementieren, APIs wie in der API-Spezifikation für den AWS KMS externen Schlüsselspeicher beschrieben. Diese Proxy-API-Anfragen sind die einzigen Anfragen, die AWS KMS an den Proxy gesendet werden. Auch wenn Sie diese Anforderungen nie direkt senden, kann das Wissen darüber Ihnen helfen, Probleme zu beheben, die mit Ihrem externen Schlüsselspeicher oder seinem Proxy auftreten könnten. Nimmt beispielsweise AWS KMS Informationen über die Latenz und die Erfolgsraten dieser API-Aufrufe in seine CloudWatch Amazon-Metriken für externe Schlüsselgeschäfte auf. Details hierzu finden Sie unter Überwachen Sie externe Schlüsselspeicher.

In der folgenden Tabelle sind die einzelnen Proxys aufgeführt und beschrieben APIs. Sie umfasst auch die AWS KMS Operationen, die einen Aufruf der Proxy-API auslösen, sowie alle Ausnahmen für AWS KMS Operationen im Zusammenhang mit der Proxy-API.

Proxy-API	Beschreibung	Verwandte AWS KMS Operationen
Decrypt	AWS KMS sendet den zu entschlüsselnden Chiffretext und die ID des zu verwendenden externen Schlüssels. Der erforderliche Verschlüsselungsalgorithmus ist AES_GCM.	Entschlüsseln, ReEncrypt
Encrypt	AWS KMS sendet zu verschlüsselnde Daten und die ID des zu verwendenden externen Schlüssels. Der erforderliche Verschlüsselungsalgorithmus ist AES_GCM.	Verschlüsseln,, GenerateDataKey, GenerateDataKeyWithoutPlaintextReEncrypt

Proxy-API	Beschreibung	Verwandte AWS KMS Operationen
GetHealthStatus	AWS KMS fordert Informationen über den Status des Proxys und Ihres externen Schlüsselmanagers an. Der Status eines jeden externen Schlüsselmanagers kann einer der folgenden sein. • `Active`: Fehlerfrei; kann den Datenverkehr abwickeln • `Degraded`: Fehlerhaft, kann aber den Datenverkehr abwickeln • `Unavailable` : Fehlerhaft; kann keinen Datenverkehr abwickeln	CreateCustomKeyStore(für öffentliche Endpunktkonnektivität), ConnectCustomKeyStore(für VPC-Endpunktdienstkonnektivität) Wenn alle externen Schlüssel manager-Instances `Unavailable` sind, schlagen Versuche, den Schlüsselspeicher zu erstellen oder eine Verbindung herzustellen, mit `XksProxyUriUnreachableException` fehl.
GetKeyMetadata	AWS KMS fordert Informationen über den externen Schlüssel an, der einem KMS-Schlüssel in Ihrem externen Schlüsselspeicher zugeordnet ist. Die Antwort enthält die Schlüsselspezifikation (`AES_256`), die Schlüsselverwendung (`[ENCRYPT, DECRYPT]`) und die Angabe, ob der externe Schlüssel `ENABLED` oder `DISABLED` ist.	CreateKey Wenn die Schlüsselspezifikation nicht `AES_256` ist oder die Schlüsselverwendung nicht `[ENCRYPT, DECRYPT]` ist oder der Status `DISABLED` ist, schlägt der Vorgang `CreateKey` mit `XksKeyInvalidConfigurationException` fehl.

Doppelte Verschlüsselung

Daten, die mit einem KMS-Schlüssel in einem externen Schlüsselspeicher verschlüsselt wurden, werden zweimal verschlüsselt. AWS KMS Verschlüsselt zunächst die Daten mit AWS KMS Schlüsselmaterial, das für den KMS-Schlüssel spezifisch ist. Dann wird der mit AWS KMS verschlüsselte Geheimtext von Ihrem externen Schlüsselmanager mit Ihrem externen Schlüssel verschlüsselt. Dieser Vorgang wird als doppelte Verschlüsselung bezeichnet.

Die doppelte Verschlüsselung stellt sicher, dass Daten, die mit einem KMS-Schlüssel in einem externen Schlüsselspeicher verschlüsselt werden, mindestens so sicher sind wie mit einem Standard-KMS-Schlüssel verschlüsselter Geheimtext. Es schützt auch Ihren Klartext bei der Übertragung von AWS KMS zu Ihrem externen Schlüsselspeicher-Proxy. Bei der doppelten Verschlüsselung behalten Sie die volle Kontrolle über Ihre Geheimtexte. Wenn Sie AWS den Zugriff auf Ihren externen Schlüssel über Ihren externen Proxy dauerhaft entziehen, wird jeglicher in AWS verbleibende Geheimtext effektiv kryptografisch zerlegt.

Jeder KMS-Schlüssel verfügt in einem externen Schlüsselspeicher über zwei kryptografische Unterstützungsschlüssel, um eine doppelte Verschlüsselung zu ermöglichen:

- Ein AWS KMS Schlüsselmaterial, das nur für den KMS-Schlüssel gilt. Dieses Schlüsselmaterial wird generiert und nur in AWS KMS FIPS 140-3 Security Level 3-zertifizierten Hardware-Sicherheitsmodulen () verwendet. HSMs
- Ein externer Schlüssel in Ihrem externen Schlüsselmanager.

Die doppelte Verschlüsselung hat die folgenden Auswirkungen:

- AWS KMS kann keinen Chiffretext entschlüsseln, der mit einem KMS-Schlüssel in einem externen Schlüsselspeicher verschlüsselt wurde, ohne über Ihren externen Schlüsselspeicher-Proxy auf Ihre externen Schlüssel zuzugreifen.
- Sie können keinen mit einem KMS-Schlüssel verschlüsselten Chiffretext in einem externen Schlüsselspeicher außerhalb von entschlüsseln AWS, selbst wenn Sie über dessen externes Schlüsselmaterial verfügen.
- Sie können einen KMS-Schlüssel, der aus einem externen Schlüsselspeicher gelöscht wurde, nicht wiederherstellen, selbst wenn Sie über sein externes Schlüsselmaterial verfügen. Jeder KMS-Schlüssel verfügt über eindeutige Metadaten, die er in den symmetrischen Geheimtext einfügt. Ein neuer KMS-Schlüssel wäre nicht in der Lage, einen mit dem ursprünglichen Schlüssel verschlüsselten Geheimtext zu entschlüsseln, selbst wenn er dasselbe externe Schlüsselmaterial verwenden würde.

Ein Beispiel für doppelte Verschlüsselung in der Praxis finden Sie unter Funktionsweise externer Schlüsselspeicher.

Funktionsweise externer Schlüsselspeicher

Ihr externer Schlüsselspeicher, der externe Schlüsselspeicher-Proxy und der externe Schlüsselmanager arbeiten zusammen, um Ihre AWS -Ressourcen zu schützen. Das folgende Verfahren zeigt den Verschlüsselungsworkflow eines typischen AWS-Service , bei dem jedes Objekt mit einem eindeutigen Datenschlüssel verschlüsselt wird, der durch einen KMS-Schlüssel geschützt ist. In diesem Fall haben Sie zum Schutz des Objekts einen KMS-Schlüssel in einem externen Schlüsselspeicher gewählt. Das Beispiel zeigt, wie doppelte Verschlüsselung AWS KMS verwendet wird, um den Datenschlüssel bei der Übertragung zu schützen und sicherzustellen, dass Chiffretext, der durch einen KMS-Schlüssel in einem externen Schlüsselspeicher generiert wird, immer mindestens so stark ist wie Chiffretext, der mit einem standardmäßigen symmetrischen KMS-Schlüssel mit Schlüsselmaterial verschlüsselt wurde. AWS KMS

Die Verschlüsselungsmethoden, die von den einzelnen Integrationen verwendet werden, sind unterschiedlich AWS-Service . AWS KMS Weitere Informationen finden Sie unter dem Thema zum Datenschutz im Kapitel über Sicherheit in der AWS-Service -Dokumentation.

1. Sie fügen Ihrer AWS-Service Ressource ein neues Objekt hinzu. Um das Objekt zu verschlüsseln, AWS-Service sendet der eine GenerateDataKeyAnfrage zur AWS KMS Verwendung eines KMS-Schlüssels in Ihrem externen Schlüsselspeicher.
2. AWS KMS generiert einen symmetrischen 256-Bit-Datenschlüssel und bereitet den Versand einer Kopie des Klartext-Datenschlüssels über Ihren externen Schlüsselspeicher-Proxy an Ihren externen Schlüsselmanager vor. AWS KMS beginnt den doppelten Verschlüsselungsprozess, indem der Klartext-Datenschlüssel mit dem Schlüsselmaterial verschlüsselt wird, das dem AWS KMS KMS-Schlüssel im externen Schlüsselspeicher zugeordnet ist.
3. AWS KMS sendet eine Verschlüsselungsanforderung an den externen Schlüsselspeicher-Proxy, der dem externen Schlüsselspeicher zugeordnet ist. Die Anforderung enthält den zu verschlüsselnden Chiffretext des Datenschlüssels und die ID des externen Schlüssels, der dem KMS-Schlüssel zugeordnet ist. AWS KMS signiert die Anfrage mit den Proxyauthentifizierungsdaten für Ihren externen Schlüsselspeicher-Proxy.

   Die Klartextkopie des Datenschlüssels wird nicht an den externen Schlüsselspeicher-Proxy gesendet.
4. Der externe Schlüsselspeicher-Proxy authentifiziert die Anforderung und leitet die Verschlüsselungsanforderung dann an Ihren externen Schlüsselmanager weiter.

Einige externe Schlüsselspeicher-Proxys implementieren auch eine optionale Autorisierungsrichtlinie, die es nur ausgewählten Prinzipalen erlaubt, unter bestimmten Bedingungen Operationen durchzuführen.

5. Ihr externer Schlüsselmanager verschlüsselt den Geheimtext des Datenschlüssels mit dem angegebenen externen Schlüssel. Der externe Schlüsselmanager gibt den doppelt verschlüsselten Datenschlüssel an Ihren externen Schlüsselspeicher-Proxy zurück, der ihn an AWS KMS zurückgibt.
6. AWS KMS gibt den Klartext-Datenschlüssel und die doppelt verschlüsselte Kopie dieses Datenschlüssels an den zurück. AWS-Service
7. Der AWS-Service verwendet den Klartext-Datenschlüssel, um das Ressourcenobjekt zu verschlüsseln, zerstört den Klartext-Datenschlüssel und speichert den verschlüsselten Datenschlüssel zusammen mit dem verschlüsselten Objekt.

   In manchen AWS-Services Fällen wird der Klartext-Datenschlüssel möglicherweise zwischengespeichert, um ihn für mehrere Objekte zu verwenden oder um ihn wiederzuverwenden, während die Ressource verwendet wird. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

Um das verschlüsselte Objekt zu entschlüsseln, AWS-Service müssen sie den verschlüsselten Datenschlüssel AWS KMS in einer Entschlüsselungsanforderung an sie zurücksenden. Um den verschlüsselten Datenschlüssel zu entschlüsseln, AWS KMS müssen Sie den verschlüsselten Datenschlüssel mit der ID des externen Schlüssels an Ihren externen Schlüsselspeicher-Proxy zurücksenden. Wenn die Entschlüsselungsanforderung an den externen Schlüsselspeicher-Proxy aus irgendeinem Grund fehlschlägt, AWS KMS kann der verschlüsselte Datenschlüssel nicht entschlüsselt werden, und das verschlüsselte Objekt AWS-Service kann nicht entschlüsselt werden.

Steuern Sie den Zugriff auf Ihren externen Schlüsselspeicher

Alle AWS KMS Zugriffskontrollfunktionen — wichtige Richtlinien, IAM-Richtlinien und Grants —, die Sie mit Standard-KMS-Schlüsseln verwenden, funktionieren auf die gleiche Weise für KMS-Schlüssel in einem externen Schlüsselspeicher. Sie können IAM-Richtlinien verwenden, um den Zugriff auf die API-Operationen zur Erstellung und Verwaltung externer Schlüsselspeicher zu steuern. Sie verwenden IAM-Richtlinien und Schlüsselrichtlinien, um den Zugriff auf die AWS KMS keys in Ihrem externen Schlüsselspeicher gespeicherten Daten zu steuern. Sie können auch Dienststeuerungsrichtlinien für Ihre AWS Organisation und VPC-Endpunktrichtlinien verwenden, um den Zugriff auf KMS-Schlüssel in Ihrem externen Schlüsselspeicher zu steuern.

Sie sollten Benutzern und Rollen ausschließlich die Berechtigungen gewähren, die sie für die Aufgaben benötigen, die sie voraussichtlich ausführen werden.

Themen

- Autorisierung externer Schlüsselspeichermanager
- Autorisierung von Benutzern von KMS-Schlüsseln in externen Schlüsselspeichern
- Autorisieren Sie AWS KMS die Kommunikation mit Ihrem externen Schlüsselspeicher-Proxy
- Proxy-Autorisierung für externen Schlüsselspeicher (optional)
- mTLS-Authentifizierung (optional)

Autorisierung externer Schlüsselspeichermanager

Prinzipale, die einen externen Schlüsselspeicher erstellen und verwalten, benötigen Berechtigungen für die benutzerdefinierten Schlüsselspeicheroperationen. Die folgende Liste beschreibt die Mindestberechtigungen, die Manager externer Schlüsselspeicher benötigen. Da es sich bei einem benutzerdefinierten Schlüsselspeicher nicht um eine AWS Ressource handelt, können Sie keine Berechtigungen für einen externen Schlüsselspeicher für Prinzipale in anderen Ländern erteilen. AWS-Konten

- `kms:CreateCustomKeyStore`
- `kms:DescribeCustomKeyStores`
- `kms:ConnectCustomKeyStore`
- `kms:DisconnectCustomKeyStore`
- `kms:UpdateCustomKeyStore`
- `kms:DeleteCustomKeyStore`

Prinzipale, die einen externen Schlüsselspeicher erstellen, benötigen die Berechtigung, die externen Schlüsselspeicherkomponenten zu erstellen und zu konfigurieren. Prinzipale können externe Schlüsselspeicher nur in ihren eigenen Konten erstellen. Um einen externen Schlüsselspeicher mit der Konnektivität eines VPC-Endpunkt-Services zu erstellen, müssen die Prinzipale über die Berechtigung verfügen, die folgenden Komponenten zu erstellen:

- Eine Amazon VPC
- Öffentliche und private Subnetze
- Einen Network Load Balancer und eine Zielgruppe

- Einen VPC-Endpunkt-Service von Amazon

Einzelheiten finden Sie unter Identity and Access Management für Amazon VPC, Identity and Access Management für VPC-Endpunkte und VPC-Endpunktservices und Elastic Load Balancing-API-Berechtigungen.

Autorisierung von Benutzern von KMS-Schlüsseln in externen Schlüsselspeichern

Prinzipale, die AWS KMS keys in Ihrem externen Schlüsselspeicher etwas erstellen und verwalten, benötigen dieselben Berechtigungen wie diejenigen, die alle KMS-Schlüssel in erstellen und verwalten. AWS KMS Die Standard-Schlüsselrichtlinie für KMS-Schlüssel in einem externen Schlüsselspeicher ist identisch mit der Standard-Schlüsselrichtlinie für KMS-Schlüssel in AWS KMS. Attributbasierte Zugriffskontrolle (ABAC), das Tags und Aliasse verwendet, um den Zugriff auf KMS-Schlüssel zu steuern, ist auch für KMS-Schlüssel in externen Schlüsselspeichern wirksam.

Prinzipale, die die KMS-Schlüssel in Ihrem benutzerdefinierten Schlüsselspeicher für kryptografische Operationen verwenden, benötigen die Berechtigung zum Ausführen der kryptografischen Operation mit dem KMS-Schlüssel, z. B. kms:Decrypt. Sie können diese Berechtigungen in einer IAM-Richtlinie oder Schlüsselrichtlinie bereitstellen. Sie benötigen keine zusätzlichen Berechtigungen, um einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher verwenden zu können.

Um eine Berechtigung festzulegen, die nur für KMS-Schlüssel in einem externen Schlüsselspeicher gilt, verwenden Sie die Richtlinienbedingung `kms:KeyOrigin` mit einem Wert von `EXTERNAL_KEY_STORE`. Sie können diese Bedingung verwenden, um die KMS: CreateKey - Berechtigung oder jede andere Berechtigung, die für eine KMS-Schlüsselressource spezifisch ist, einzuschränken. Die folgende IAM-Richtlinie erlaubt es beispielsweise der Identität, der sie zugeordnet ist, die angegebenen Operationen für alle KMS-Schlüssel des Kontos aufzurufen, vorausgesetzt, die KMS-Schlüssel befinden sich in einem externen Schlüsselspeicher. Beachten Sie, dass Sie die Berechtigung auf KMS-Schlüssel in einem externen Schlüsselspeicher und KMS-Schlüssel in einem AWS-Konto, aber nicht auf einen bestimmten externen Schlüsselspeicher im Konto beschränken können.

```
{
  "Sid": "AllowKeysInExternalKeyStores",
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
```

```
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "EXTERNAL_KEY_STORE"
    }
  }
}
```

Autorisieren Sie AWS KMS die Kommunikation mit Ihrem externen Schlüsselspeicher-Proxy

AWS KMS kommuniziert mit Ihrem externen Schlüsselmanager nur über den externen Schlüsselspeicher-Proxy, den Sie bereitstellen. AWS KMS authentifiziert sich gegenüber Ihrem Proxy, indem es seine Anfragen mithilfe des Signature Version 4-Prozesses (Sigv4) mit den von Ihnen angegebenen Proxyauthentifizierungsdaten für den externen Schlüsselspeicher signiert. Wenn Sie öffentliche Endpunktkonnektivität für Ihren externen Schlüsselspeicher-Proxy verwenden, sind AWS KMS keine zusätzlichen Berechtigungen erforderlich.

Wenn Sie jedoch VPC-Endpunktservice-Konnektivität verwenden, müssen Sie die AWS KMS Erlaubnis erteilen, einen Schnittstellenendpunkt für Ihren Amazon VPC-Endpunktservice zu erstellen. Diese Berechtigung ist unabhängig davon erforderlich, ob sich der externe Schlüsselspeicher-Proxy in Ihrer VPC oder der externe Schlüsselspeicher-Proxy an einem anderen Ort befindet, aber den VPC-Endpunktdienst für die Kommunikation verwendet. AWS KMS

Um die Erstellung eines Schnittstellenendpunkts AWS KMS zu ermöglichen, verwenden Sie die Amazon VPC-Konsole oder den ModifyVpcEndpointServicePermissionsVorgang. Erlauben Sie Berechtigungen für den folgenden Prinzipal: `cks.kms.<region>.amazonaws.com`.

Mit dem folgenden AWS CLI Befehl können Sie beispielsweise eine Verbindung AWS KMS zum angegebenen VPC-Endpunktdienst in der Region USA West (Oregon) (us-west-2) herstellen. Bevor Sie diesen Befehl verwenden, ersetzen Sie die Amazon VPC Service ID und AWS-Region durch gültige Werte für Ihre Konfiguration.

```
modify-vpc-endpoint-service-permissions
--service-id vpce-svc-12abc34567def0987
--add-allowed-principals '["cks.kms.us-west-2.amazonaws.com"]'
```

Um diese Berechtigung zu entfernen, verwenden Sie die Amazon VPC-Konsole oder den ModifyVpcEndpointServicePermissionsmit dem `RemoveAllowedPrincipals` Parameter.

Proxy-Autorisierung für externen Schlüsselspeicher (optional)

Einige externe Schlüsselspeicher-Proxys implementieren Autorisierungsanforderungen für die Verwendung ihrer externen Schlüssel. Ein externer Schlüsselspeicher-Proxy ist berechtigt, aber nicht verpflichtet, ein Autorisierungsschema zu entwerfen und zu implementieren, das es bestimmten Benutzern erlaubt, bestimmte Operationen nur unter bestimmten Bedingungen anzufordern. Beispielsweise könnte ein Proxy so konfiguriert sein, dass er Benutzer:in A die Verschlüsselung mit einem bestimmten externen Schlüssel erlaubt, aber nicht die Entschlüsselung mit diesem Schlüssel.

Die Proxy-Autorisierung ist unabhängig von der SIGV4-basierten Proxyauthentifizierung, die für alle externen Schlüsselspeicher-Proxys AWS KMS erforderlich ist. Sie ist auch unabhängig von den Schlüsselrichtlinien, IAM-Richtlinien und Berechtigungen, die den Zugriff auf Operationen zulassen, die den externen Schlüsselspeicher oder seine KMS-Schlüssel betreffen.

Um die Autorisierung durch den externen Schlüsselspeicher-Proxy zu ermöglichen, AWS KMS enthält jede Proxy-API-Anfrage Metadaten, einschließlich des Aufrufers, des KMS-Schlüssels, des AWS KMS Vorgangs und (falls vorhanden). AWS-Service Die Anforderungsmetadaten für Version 1 (v1) der externen Schlüssel-Proxy-API lauten wie folgt.

```
"requestMetadata": {
    "awsPrincipalArn": string,
    "awsSourceVpc": string, // optional
    "awsSourceVpce": string, // optional
    "kmsKeyArn": string,
    "kmsOperation": string,
    "kmsRequestId": string,
    "kmsViaService": string // optional
}
```

Sie können Ihren Proxy beispielsweise so konfigurieren, dass Anfragen von einem bestimmten Principal (`awsPrincipalArn`) zugelassen werden, aber nur, wenn die Anfrage im Namen des Prinzipals von einem bestimmten AWS-Service (`kmsViaService`) gestellt wird.

Wenn die Proxyautorisierung fehlschlägt, schlägt der entsprechende AWS KMS Vorgang fehl und es wird eine Meldung angezeigt, die den Fehler erklärt. Details hierzu finden Sie unter Probleme mit der Proxy-Autorisierung.

mTLS-Authentifizierung (optional)

Damit Ihr externer Schlüsselspeicher-Proxy Anfragen authentifizieren kann AWS KMS, AWS KMS signiert er alle Anfragen an Ihren externen Schlüsselspeicher-Proxy mit den Signature V4 (Sigv4) - Proxyauthentifizierungsdaten für Ihren externen Schlüsselspeicher.

Um zusätzliche Sicherheit zu bieten, dass Ihr externer Schlüsselspeicher-Proxy nur auf AWS KMS Anfragen reagiert, unterstützen einige externe Schlüsselproxys Mutual Transport Layer Security (mTLS), bei der beide Parteien einer Transaktion Zertifikate verwenden, um sich gegenseitig zu authentifizieren. mTLS fügt der serverseitigen Authentifizierung, die Standard-TLS bietet, eine clientseitige Authentifizierung hinzu, bei der der externe Schlüsselspeicher-Proxyserver den AWS KMS Client authentifiziert. In dem seltenen Fall, dass Ihre Proxy-Authentifizierungsanmeldeinformation kompromittiert wird, verhindert mTLS, dass ein Dritter erfolgreiche API-Anforderungen an den externen Schlüsselspeicher-Proxy stellt.

Um mTLS zu implementieren, konfigurieren Sie Ihren externen Schlüsselspeicher-Proxy so, dass er nur clientseitige TLS-Zertifikate mit den folgenden Eigenschaften akzeptiert:

- Der Subject Common Name auf dem TLS-Zertifikat muss `cks.kms.<Region>.amazonaws.com` lauten, zum Beispiel `cks.kms.eu-west-3.amazonaws.com`.
- Das Zertifikat muss mit einer Zertifizierungsstelle verknüpft sein, die mit Amazon Trust Services verbunden ist.

Wählen Sie eine Proxy-Konnektivitätsoption für externen Schlüsselspeicher

Bevor Sie Ihren externen Schlüsselspeicher erstellen, wählen Sie die Verbindungsoption aus, die bestimmt, wie mit Ihren externen Schlüsselspeicherkomponenten AWS KMS kommuniziert wird. Der Rest des Planungsprozesses hängt davon ab, welche Konnektivitätsoption Sie auswählen.

Wenn Sie einen externen Schlüsselspeicher erstellen, müssen Sie festlegen, wie mit Ihrem externen Schlüsselspeicher-Proxy AWS KMS kommuniziert wird. Diese Auswahl bestimmt, welche Komponenten Sie benötigen und wie Sie sie konfigurieren. AWS KMS unterstützt die folgenden Konnektivitätsoptionen. Wählen Sie die Option aus, die Ihren Leistungs- und Sicherheitszielen entspricht.

Bevor Sie beginnen, vergewissern Sie sich, dass Sie einen externen Schlüsselspeicher benötigen. Die meisten Kunden können KMS-Schlüssel verwenden, die durch AWS KMS Schlüsselmaterial unterstützt werden.

> **Note**
>
> Wenn Ihr externer Schlüsselspeicher-Proxy in Ihren externen Schlüsselmanager integriert ist, ist die Konnektivität möglicherweise vordefiniert. Hilfreiche Informationen finden Sie in der Dokumentation für Ihren externen Schlüsselmanager oder externen Schlüsselspeicher-Proxy.

Sie können die Konnektivitätsoption für Ihren externen Schlüsselspeicher-Proxy auch dann ändern, wenn der entsprechende Schlüsselspeicher aktiv ist. Dieser Vorgang muss jedoch sorgfältig geplant und ausgeführt werden, um Unterbrechungen zu minimieren, Fehler zu vermeiden und einen unterbrechungsfreien Zugriff auf die kryptografischen Schlüssel sicherzustellen, mit denen Ihre Daten verschlüsselt werden.

Konnektivität eines öffentlichen Endpunkts

AWS KMS stellt über das Internet über einen öffentlichen Endpunkt eine Verbindung zum externen Schlüsselspeicher-Proxy (XKS-Proxy) her.

Diese Konnektivitätsoption ist einfacher einzurichten und zu verwalten und eignet sich gut für einige Schlüsselverwaltungsmodelle. Sie entspricht jedoch möglicherweise nicht den Sicherheitsanforderungen einiger Organisationen.

Voraussetzungen

Wenn Sie sich für die Konnektivität eines öffentlichen Endpunkts entscheiden, werden die folgenden Komponenten benötigt.

- Ihr externer Schlüsselspeicher-Proxy muss unter einem öffentlich routingfähigen Endpunkt erreichbar sein.
- Sie können einen öffentlichen Endpunkt für mehrere externe Schlüsselspeicher verwenden, sofern diese unterschiedliche Werte für den Proxy-URI-Pfad nutzen.
- Sie können nicht denselben Endpunkt für einen externen Schlüsselspeicher mit öffentlicher Endpunktkonnektivität und einen externen Schlüsselspeicher mit VPC-Endpunktdienst-Konnektivität im selben verwenden AWS-Region, auch wenn sich die Schlüsselspeicher in unterschiedlichen AWS-Konten Ordnern befinden.
- Sie benötigen ein TLS-Zertifikat von einer öffentlichen Zertifizierungsstelle, die für externe Schlüsselspeicher unterstützt wird. Eine Liste finden Sie unter Vertrauenswürdige Zertifizierungsstellen.

Der Subject Common Name (CN) auf dem TLS-Zertifikat muss mit dem Domainnamen im Proxy-URI-Endpunkt für den externen Schlüsselspeicher-Proxy identisch sein. Ist der öffentliche Endpunkt beispielsweise `https://myproxy.xks.example.com`, muss der CN auf dem TLS-Zertifikat `myproxy.xks.example.com` oder `*.xks.example.com` lauten.

- Stellen Sie sicher, dass alle Firewalls zwischen AWS KMS und dem externen Schlüsselspeicher-Proxy den Datenverkehr zu und von Port 443 auf dem Proxy zulassen. AWS KMS kommuniziert auf Port 443 über. IPv4 Dieser Wert kann nicht konfiguriert werden.

Alle Anforderungen für einen externen Schlüsselspeicher finden Sie in den Informationen zum Erfüllen der Voraussetzungen.

Konnektivität eines VPC-Endpunktservice

AWS KMS stellt eine Verbindung zum externen Schlüsselspeicher-Proxy (XKS-Proxy) her, indem ein Schnittstellenendpunkt zu einem von Ihnen erstellten und konfigurierten Amazon VPC-Endpunktservice erstellt wird. Sie müssen den VPC-Endpunktservice erstellen und Ihre VPC mit Ihrem externen Schlüsselmanager verbinden.

Ihr Endpunktdienst kann jede der unterstützten network-to-Amazon VPC-Optionen für die Kommunikation verwenden, darunter AWS Direct Connect.

Einrichtung und Verwaltung dieser Konnektivitätsoption sind komplizierter. Es wird jedoch verwendet AWS PrivateLink, sodass Sie eine private Verbindung AWS KMS zu Ihrer Amazon VPC und Ihrem externen Schlüsselspeicher-Proxy herstellen können, ohne das öffentliche Internet zu nutzen.

Sie können Ihren externen Schlüsselspeicher-Proxy in Ihrer Amazon VPC lokalisieren.

Oder suchen Sie Ihren externen Schlüsselspeicher-Proxy außerhalb von AWS und verwenden Sie Ihren Amazon VPC-Endpunktservice nur für die sichere Kommunikation mit AWS KMS.

Weitere Informationen:

- Sehen Sie sich den Prozess zum Erstellen eines externen Schlüsselspeichers an, einschließlich der Informationen zum Erfüllen der Voraussetzungen. Er stellt sicher, dass Sie über alle benötigten Komponenten verfügen, wenn Sie einen externen Schlüsselspeicher erstellen.

- Erfahren Sie, wie Sie den Zugriff auf Ihren externen Schlüsselspeicher steuern, einschließlich der Berechtigungen, die Administratoren und Benutzer von externen Schlüsselspeicheradministratoren benötigen.
- Erfahren Sie mehr über die CloudWatch Amazon-Kennzahlen und -Dimensionen, die für externe Schlüsselgeschäfte AWS KMS aufgezeichnet werden. Wir empfehlen Ihnen dringend, Alarme zur Überwachung Ihres externen Schlüsselspeichers zu erstellen, damit Sie erste Anzeichen von Leistungs- und Betriebsproblemen erkennen.

VPC-Endpunktdienstkonnektivität konfigurieren

Verwenden Sie die Anleitungen in diesem Abschnitt, um die AWS Ressourcen und zugehörigen Komponenten zu erstellen und zu konfigurieren, die für einen externen Schlüsselspeicher erforderlich sind, der VPC-Endpunktdienstkonnektivität verwendet. Die für diese Konnektivitätsoption aufgeführten Ressourcen sind zusätzlich zu den für alle externen Schlüsselspeicher benötigten Ressourcen erforderlich. Nachdem Sie die erforderlichen Ressourcen erstellt und konfiguriert haben, können Sie den externen Schlüsselspeicher erstellen.

Sie können Ihren externen Schlüsselspeicher-Proxy in Ihrer Amazon VPC oder den Proxy außerhalb von VPC lokalisieren AWS und Ihren VPC-Endpunktservice für die Kommunikation verwenden.

Bevor Sie beginnen, vergewissern Sie sich, dass Sie einen externen Schlüsselspeicher benötigen. Die meisten Kunden können KMS-Schlüssel verwenden, die durch AWS KMS Schlüsselmaterial unterstützt werden.

> **Note**
>
> Einige der Elemente, die für die Konnektivität eines VPC-Endpunktservice erforderlich sind, sind möglicherweise in Ihrem externen Schlüsselmanager enthalten. Außerdem gelten unter Umständen für Ihre Software zusätzliche Konfigurationsanforderungen. Bevor Sie die AWS Ressourcen in diesem Abschnitt erstellen und konfigurieren, sollten Sie die Dokumentation zu Ihrem Proxy und Key Manager lesen.

Themen

- Anforderungen für die Konnektivität eines VPC-Endpunktservice
- Schritt 1: Amazon-VPC und Subnetze erstellen
- Schritt 2: Erstellen Sie eine Zielgruppe

- Schritt 3: Erstellen Sie einen Netzwerk-Loadbalancer
- Schritt 4: Erstellen Sie einen VPC-Endpunktdienst
- Schritt 5: Überprüfen Sie Ihre private DNS-Namensdomäne
- Schritt 6: Autorisieren Sie die Verbindung AWS KMS zum VPC-Endpunktdienst

Anforderungen für die Konnektivität eines VPC-Endpunktservice

Wenn Sie sich für Ihren externen Schlüsselspeicher für die Konnektivität eines VPC-Endpunktservice entscheiden, sind die folgenden Ressourcen erforderlich.

Um die Netzwerklatenz zu minimieren, erstellen Sie Ihre AWS Komponenten in der unterstützten Version AWS-Region, die Ihrem externen Schlüsselmanager am nächsten kommt. Wählen Sie nach Möglichkeit eine Region mit einer Netzwerk-Round-Trip-Zeit (RTT) von maximal 35 Millisekunden.

- Eine Amazon VPC, die mit Ihrem externen Schlüsselmanager verbunden ist. Sie muss über mindestens zwei private Subnetze in zwei verschiedenen Availability Zones verfügen.

  Sie können eine vorhandene Amazon VPC für Ihren externen Schlüsselspeicher verwenden, sofern diese die Anforderungen für die Verwendung mit einem externen Schlüsselspeicher erfüllt. Mehrere externe Schlüsselspeicher können sich eine Amazon VPC teilen, aber jeder externe Schlüsselspeicher muss seinen eigenen VPC-Endpunktservice und privaten DNS-Namen haben.
- Ein Amazon VPC-Endpunktservice, der von AWS PrivateLink unterstützt wird, mit einem Network Load Balancer und einer Zielgruppe.

  Der Endpunktservice darf keine Akzeptanz verlangen. Außerdem müssen Sie AWS KMS als zulässigen Prinzipal hinzufügen. Auf diese Weise können AWS KMS Schnittstellen-Endpunkte erstellt werden, sodass sie mit Ihrem externen Schlüsselspeicher-Proxy kommunizieren können.
- Ein privater DNS-Name für den VPC-Endpunktservice, der in seiner AWS-Region eindeutig ist.

  Der private DNS-Name muss eine Subdomain einer öffentlichen Domain höherer Ebene sein. Wenn der private DNS-Name beispielsweise `myproxy-private.xks.example.com` lautet, muss er eine Subdomain einer öffentlichen Domain wie `xks.example.com` oder `example.com` sein.

  Sie müssen den Besitz der DNS-Domain für den privaten DNS-Namen bestätigen.
- Ein TLS-Zertifikat für Ihren externen Schlüsselspeicher-Proxy, das von einer unterstützten öffentlichen Zertifizierungsstelle ausgestellt wurde.

Der Subject Common Name (CN) auf dem TLS-Zertifikat muss mit dem privaten DNS-Namen übereinstimmen. Wenn der private DNS-Name beispielsweise `myproxy-private.xks.example.com` lautet, muss der CN auf dem TLS-Zertifikat `myproxy-private.xks.example.com` oder `*.xks.example.com` lauten.

Alle Anforderungen für einen externen Schlüsselspeicher finden Sie in den Informationen zum Erfüllen der Voraussetzungen.

Schritt 1: Amazon-VPC und Subnetze erstellen

Die Konnektivität eines VPC-Endpunktservice erfordert eine Amazon VPC, die mit Ihrem externen Schlüsselmanager mit mindestens zwei privaten Subnetzen verbunden ist. Sie können eine Amazon VPC erstellen oder eine vorhandene Amazon VPC verwenden, die die Anforderungen für externe Schlüsselspeicher erfüllt. Informationen zum Erstellen einer neuen Amazon VPC finden Sie unter Erstellen einer VPC im Amazon Virtual Private Cloud-Benutzerhandbuch.

Anforderungen für Ihre Amazon VPC

Damit Sie unter Verwendung der Konnektivität eines VPC-Endpunktservice mit externen Schlüsselspeichern arbeiten können, muss die Amazon VPC die folgenden Eigenschaften haben:

- Muss sich in derselben Region befinden AWS-Konto und unterstützt werden wie Ihr externer Schlüsselspeicher.
- Sie muss über mindestens zwei private Subnetze verfügen, die in verschiedenen Availability Zones sind.
- Der private IP-Adressbereich Ihrer Amazon VPC darf sich nicht mit dem privaten IP-Adressbereich des Rechenzentrums überschneiden, von dem Ihr externer Schlüsselmanager gehostet wird.
- Alle Komponenten müssen verwendet werden IPv4.

Sie haben zahlreiche Möglichkeiten, die Amazon VPC mit Ihrem externen Schlüsselspeicher-Proxy zu verbinden. Wählen Sie eine Option aus, die Ihre Leistungs- und Sicherheitsanforderungen erfüllt. Eine Liste finden Sie unter Connect Ihrer VPC mit anderen Netzwerken und Network-to-Amazon VPC-Konnektivitätsoptionen. Weitere Informationen finden Sie im AWS Direct Connect und im AWS Site-to-Site VPN -Benutzerhandbuch.

Erstellen einer Amazon VPC für Ihren externen Schlüsselspeicher

Verwenden Sie folgende Anweisungen zum Erstellen der Amazon VPC für Ihren externen Schlüsselspeicher. Eine Amazon VPC ist nur erforderlich, wenn Sie die Option Konnektivität eines VPC-Endpunktservice auswählen. Sie können eine vorhandene Amazon VPC verwenden, die die Anforderungen für einen externen Schlüsselspeicher erfüllt.

Folgen Sie den Anweisungen im Thema Erstellen einer VPC sowie von Subnetzen und anderen VPC-Ressourcen und verwenden Sie dabei die folgenden erforderlichen Werte. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Value (Wert)
IPv4 CIDR-Block	Geben Sie die IP-Adressen für Ihre VPC ein. Der private IP-Adressbereich Ihrer Amazon VPC darf sich nicht mit dem privaten IP-Adressbereich des Rechenzentrums überschneiden, von dem Ihr externer Schlüsselmanager gehostet wird.
Anzahl der Availability Zones () AZs	2 oder mehr
Number of public subnets (Anzahl der öffentlichen Subnetze)	Keine erforderlich (0)
Number of private subnets (Anzahl der privaten Subnetze)	Eines pro AZ
NAT gateways (NAT-Gateways)	Keine erforderlich

Feld	Value (Wert)
VPC endpoints (VPC-Endpunkte)	Keine erforderlich
Enable DNS hostnames (DNS-Hostnamen aktivieren)	Ja
Enable DNS resolution (DNS-Auflösung aktivieren)	Ja

Testen Sie unbedingt die VPC-Kommunikation. Wenn sich Ihr externer Schlüsselspeicher-Proxy beispielsweise nicht in Ihrer Amazon VPC befindet, erstellen Sie eine EC2 Amazon-Instance in Ihrer Amazon VPC und stellen Sie sicher, dass die Amazon VPC mit Ihrem externen Schlüsselspeicher-Proxy kommunizieren kann.

Verbinden der VPC mit dem externen Schlüsselmanager

Verbinden Sie die VPC mit dem Rechenzentrum, von dem Ihr externer Schlüsselmanager unter Verwendung einer der von Amazon VPC unterstützten Netzwerkkonnektivitätsoptionen gehostet wird. Stellen Sie sicher, dass die EC2 Amazon-Instance in der VPC (oder der externe Schlüsselspeicher-Proxy, falls er sich in der VPC befindet) mit dem Rechenzentrum und dem externen Schlüsselmanager kommunizieren kann.

Schritt 2: Erstellen Sie eine Zielgruppe

Bevor Sie den erforderlichen VPC-Endpunktservice erstellen, erstellen Sie die zugehörigen erforderlichen Komponenten, einen Network Load Balancer (NLB) und eine Zielgruppe. Der NLB verteilt die Anfragen an mehrere fehlerfreie Ziele, von denen jedes die Anfrage erfüllen kann. In diesem Schritt erstellen Sie eine Zielgruppe mit mindestens zwei Hosts für Ihren externen Schlüsselspeicher-Proxy und registrieren Ihre IP-Adressen bei der Zielgruppe.

Folgen Sie den Anweisungen im Thema Konfigurieren einer Zielgruppe und verwenden Sie dabei die folgenden erforderlichen Werte. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Wert
Target type (Zieltyp)	IP-Adressen
Protocol (Protokoll)	TCP
Port	443
IP address type (IP-Adresstyp)	IPv4
VPC	Wählen Sie die VPC aus, in der Sie den VPC-Endpunktservice für Ihren externen Schlüsselspeicher erstellen werden.
Health check protocol and path (Zustands prüfungs-Protokoll und -Pfad)	Ihr Zustandsprüfungs-Protokoll und Ihr Zustandsprüfungs-Pfad werden sich von der Konfiguration für Ihren externen Schlüsselspeicher-Proxy untersche iden. Weitere Informationen finden Sie in der Dokumentation für Ihren externen Schlüsselmanager oder externen Schlüsselspeicher-Proxy. Allgemeine Informationen zum Konfigurieren von Zustandsprüfungen für Ihre Zielgruppen finden Sie unter Zustandsprüfungen für Ihre Zielgruppen im Benutzerhandbuch zum Elastic Load Balancing für Network Load Balancer.
Network (Netzwerk)	Andere private IP-Adresse
IPv4 Adresse	Die privaten Adressen für Ihren externen Schlüsselspeicher-Proxy
Ports	443

Schritt 3: Erstellen Sie einen Netzwerk-Loadbalancer

Der Network Load Balancer verteilt die Netzwerkzugriffe, einschließlich Anfragen von AWS KMS an Ihren externen Schlüsselspeicher-Proxy, an die konfigurierten Ziele.

Folgen Sie den Anweisungen im Thema Konfigurieren eines Load Balancers und eines Listeners, um unter Verwendung der folgenden erforderlichen Werte einen Listener zu konfigurieren und hinzuzufügen und einen Load Balancer zu erstellen. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Wert
Scheme (Schema)	Intern
IP-Adresstyp	IPv4
Network mapping (Netzwerk zuordnung)	Wählen Sie die VPC aus, in der Sie den VPC-Endpunktservice für Ihren externen Schlüsselspeicher erstellen werden.
Mapping (Zuordnung)	Wählen Sie die beiden Availability Zones (mindestens zwei) aus, die Sie für Ihre VPC-Subnetze konfiguriert haben. Verifizieren Sie die Namen der Subnetze und die private IP-Adresse.
Protocol (Protokoll)	TCP
Port	443
Default action: Forward to (Standard aktion: Weiterleiten an)	Wählen Sie die Zielgruppe für Ihren Network Load Balancer aus.

Schritt 4: Erstellen Sie einen VPC-Endpunktdienst

Normalerweise erstellen Sie einen Endpunkt für einen Service. Wenn Sie jedoch einen VPC-Endpunktdienst erstellen, sind Sie der Anbieter und erstellen einen AWS KMS Endpunkt für Ihren Service. Für einen externen Schlüsselspeicher erstellen Sie einen VPC-Endpunktservice mit dem

Network Load Balancer, den Sie im vorherigen Schritt erstellt haben. Der VPC-Endpunktdienst muss sich in derselben AWS-Konto unterstützten Region wie Ihr externer Schlüsselspeicher befinden.

Mehrere externe Schlüsselspeicher können sich eine Amazon VPC teilen, aber jeder externe Schlüsselspeicher muss seinen eigenen VPC-Endpunktservice und privaten DNS-Namen haben.

Folgen Sie den Anweisungen im Thema Erstellen eines Endpunktservice, um Ihren VPC-Endpunktservice mit den folgenden erforderlichen Werten zu erstellen. Übernehmen Sie für andere Felder die Standardwerte und geben Sie Namen wie gewünscht an.

Feld	Wert
Load balancer type (Load Balancer-Typ)	Network (Netzwerk)
Available load balancers (Verfügbare Load Balancer)	Wählen Sie den Network Load Balancer aus, den Sie im vorherigen Schritt erstellt haben. Falls Ihr neuer Load Balancer nicht in der Liste aufgeführt wird, stellen Sie sicher, dass er aktiv ist. Es kann einige Minuten dauern, bis der Zustand des Load Balancers von der Bereitstellung in den aktiven Status ändert.
Acceptance required (Akzeptanz erforderlich)	Falsch. Wählen Sie dieses Kontrollkästchen ab. Erfordert keine Annahme. AWS KMS kann ohne eine manuelle Annahme keine Verbindung zum VPC-Endpunktdienst herstellen. Wenn die Akzeptanz erforderlich ist, schlägt das Erstellen des externen Schlüsselspeichers mit einer `XksProxyInvalidConfigurationException` -Ausnahme fehl.
Enable private DNS name (Privaten DNS-Namen aktivieren)	Ordnen Sie dem Service einen privaten DNS-Namen zu.
Private DNS name (Privater DNS-Name)	Geben Sie einen privaten DNS-Namen ein, der in seiner AWS-Region eindeutig ist.

Feld	Wert
	Der private DNS-Name muss eine Subdomain einer öffentlichen Domain höherer Ebene sein. Wenn der private DNS-Name beispielsweise `myproxy-private.xks.example.com` lautet, muss er eine Subdomain einer öffentlichen Domain wie `xks.example.com` oder `example.com` sein. Dieser private DNS-Name muss mit dem Subject Common Name (CN) im TLS-Zertifikat übereinstimmen, das auf Ihrem externen Schlüsselspeicher-Proxy konfiguriert ist. Wenn der private DNS-Name beispielsweise `myproxy-private.xks.example.com` lautet, muss der CN auf dem TLS-Zertifikat `myproxy-private.xks.example.com` oder `*.xks.example.com` lauten. Falls das Zertifikat und der private DNS-Name nicht übereinstimmen, schlagen Versuche, einen externen Schlüsselspeicher mit seinem externen Schlüssel speicher-Proxy zu verbinden, mit dem Verbindungsfehlercode `XKS_PROXY_INVALID_TLS_CONFIGURATION` fehl. Details hierzu finden Sie unter Allgemeine Konfigurationsfehler.
Supported IP address types (Unterstützte IP-Adresstypen)	IPv4

Schritt 5: Überprüfen Sie Ihre private DNS-Namensdomäne

Wenn Sie einen VPC-Endpunktservice erstellen, lautet der Verfizierungsstatus für die zugehörige Domain `pendingVerification`. Bevor Sie den VPC-Endpunktservice verwenden, um einen externen Schlüsselspeicher zu erstellen, muss dieser Status `verified` lauten. Um zu bestätigen, dass Sie der Besitzer der Domain sind, die Ihrem privaten DNS-Namen zugeordnet ist, müssen Sie einen TXT-Datensatz in einem öffentlichen DNS-Server erstellen.

Wenn der private DNS-Name für Ihren VPC-Endpunktdienst beispielsweise lautet`myproxy-private.xks.example.com`, müssen Sie einen TXT-Eintrag in einer öffentlichen Domain erstellen, z. B. `xks.example.com` oder, je nachdem`example.com`, was öffentlich ist. AWS PrivateLink sucht zuerst am `xks.example.com` und dann weiter nach dem TXT-Eintrag. `example.com`

> **Tip**
>
> Nachdem Sie einen TXT-Datensatz hinzugefügt haben, kann es einige Minuten dauern, bis sich der Wert von Domain verification status (Domain-Verifizierungsstatus) von `pendingVerification` in `verify` ändert.

Ermitteln Sie zuerst mithilfe einer der nachfolgenden Methoden, welchen Verifizierungsstatus Ihre Domain hat. Gültige Werte sind `verified`, `pendingVerification` und `failed`.

- Wählen Sie in der Amazon VPC-Konsole die Option Endpoint services (Endpunktservices) und dann Ihren Endpunktservice aus. Im Detailbereich wird Domain verification status (Domain-Verifizierungsstatus) angezeigt.
- Verwenden Sie die DescribeVpcEndpointServiceConfigurationsOperation. Der Wert für `State` ist im Feld `ServiceConfigurations.PrivateDnsNameConfiguration.State`.

Sollte der Verifizierungsstatus nicht `verified` lauten, folgen Sie den Anweisungen im Thema Domain-Eigentumsüberprüfung, um dem DNS-Server Ihrer Domain einen TXT-Datensatz hinzuzufügen und zu verifizieren, dass der TXT-Datensatz veröffentlicht wird. Überprüfen Sie den Verifizierungsstatus danach erneut.

Sie sind nicht dazu verpflichtet, einen A-Datensatz für den Namen Ihrer privaten DNS-Domain zu erstellen. Wenn AWS KMS Sie einen Schnittstellenendpunkt zu Ihrem VPC-Endpunktdienst erstellen, wird AWS PrivateLink automatisch eine gehostete Zone mit dem erforderlichen A-Eintrag für den privaten Domainnamen in der AWS KMS VPC erstellt. Für externe Schlüsselspeicher mit der Konnektivität eines VPC-Endpunktservice geschieht dies, wenn Sie Ihren externen Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy verbinden.

Schritt 6: Autorisieren Sie die Verbindung AWS KMS zum VPC-Endpunktdienst

Sie müssen der Liste der erlaubten Prinzipale für Ihren VPC-Endpunktdienst etwas hinzufügen AWS KMS . Dies ermöglicht AWS KMS die Erstellung von Schnittstellenendpunkten zu Ihrem VPC-Endpunktdienst. Wenn dies kein zulässiger Prinzipal AWS KMS ist, schlagen Versuche, einen externen Schlüsselspeicher zu erstellen, mit einer `XksProxyVpcEndpointServiceNotFoundException` Ausnahme fehl.

Folgen Sie den Anweisungen im Thema Verwalten von Berechtigungen im AWS PrivateLink - Benutzerhandbuch. Verwenden Sie den folgenden erforderlichen Wert.

Feld	Wert
ARN	`cks.kms.<region>.amazonaws.com` Beispiel: `cks.kms.us-east-1.amazonaws.com`

Weiter: Erstellen Sie einen externen Schlüsselspeicher

Erstellen Sie einen externen Schlüsselspeicher

Sie können in jeder AWS-Konto Region einen oder mehrere externe Schlüsselspeicher erstellen. Jeder externe Schlüsselspeicher muss einem externen Schlüsselmanager außerhalb von AWS und einem externen Schlüsselspeicher-Proxy (XKS-Proxy) zugeordnet sein, der die Kommunikation zwischen AWS KMS und Ihrem externen Schlüsselmanager vermittelt. Details hierzu finden Sie unter Wählen Sie eine Proxy-Konnektivitätsoption für externen Schlüsselspeicher. Bevor Sie beginnen, vergewissern Sie sich, dass Sie einen externen Schlüsselspeicher benötigen. Die meisten Kunden können KMS-Schlüssel verwenden, denen AWS KMS Schlüsselmaterial zugrunde liegt.

> **Tip**
>
> Einige externe Schlüsselmanager bieten eine einfachere Methode zum Erstellen eines externen Schlüsselspeichers. Weitere Informationen finden Sie in der Dokumentation zum externen Schlüsselmanager.

Vor dem Erstellen des externen Schlüsselspeichers müssen Sie einige Voraussetzungen erfüllen. Geben Sie während des Erstellungsprozesses die Eigenschaften Ihres externen Schlüsselspeichers an. Am wichtigsten ist, dass Sie angeben, ob Ihr externer Schlüsselspeicher einen öffentlichen Endpunkt oder einen VPC-Endpunktdienst AWS KMS verwendet, um eine Verbindung zu seinem externen Schlüsselspeicher-Proxy herzustellen. Sie geben auch die Verbindungsdetails an, einschließlich des URI-Endpunkts des Proxys und des Pfads innerhalb dieses Proxyendpunkts, über den API-Anfragen an den Proxy AWS KMS gesendet werden.

Überlegungen

- KMS kann nicht IPv6 mit externen Schlüsselspeichern kommunizieren.
- Wenn Sie öffentliche Endpunktkonnektivität verwenden, stellen Sie sicher, dass diese über eine HTTPS-Verbindung mit Ihrem Proxy über das Internet kommunizieren AWS KMS können.

Dazu gehört die Konfiguration von TLS auf dem externen Schlüsselspeicher-Proxy und die Sicherstellung, dass alle Firewalls zwischen AWS KMS und dem Proxy IPv4 Datenverkehr zu und von Port 443 auf dem Proxy zulassen. Beim Erstellen eines externen Schlüsselspeichers mit öffentlicher Endpunktkonnektivität wird die Verbindung AWS KMS getestet, indem eine Statusanfrage an den externen Schlüsselspeicher-Proxy gesendet wird. Mit diesem Test wird überprüft, ob der Endpunkt erreichbar ist und ob der externe Schlüsselspeicher-Proxy eine mit der Anmeldeinformation für die Proxy-Authentifizierung des externen Schlüsselspeichers signierte Anforderung akzeptiert. Wenn diese Testanforderung fehlschlägt, schlägt der Vorgang zum Erstellen des externen Schlüsselspeichers fehl.

- Wenn Sie die Konnektivität eines VPC-Endpunkt-Service verwenden, stellen Sie sicher, dass der Network Load Balancer, der private DNS-Name und der VPC-Endpunkt-Service korrekt konfiguriert und betriebsbereit sind. Wenn sich der externe Schlüsselspeicher-Proxy nicht in der VPC befindet, müssen Sie sicherstellen, dass der VPC-Endpunktdienst mit dem externen Schlüsselspeicher-Proxy kommunizieren kann. (AWS KMS testet die VPC-Endpunktdienst-Konnektivität, wenn Sie den externen Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy verbinden.)
- AWS KMS zeichnet CloudWatch Amazon-Metriken und -Dimensionen speziell für externe Schlüsselgeschäfte auf. Überwachungsdiagramme, die auf einigen dieser Metriken basieren, werden in der AWS KMS Konsole für jeden externen Schlüsselspeicher angezeigt. Wir empfehlen dringend, dass Sie diese Metriken verwenden, um Alarme zu erstellen, die Ihren externen Schlüsselspeicher überwachen. Diese Alarme warnen Sie vor ersten Anzeichen von Leistungs- und Betriebsproblemen, bevor sie auftreten. Detaillierte Anweisungen finden Sie unter Überwachen Sie externe Schlüsselspeicher.
- Externe Schlüsselspeicher unterliegen Ressourcenkontingenten. Die Verwendung von KMS-Schlüsseln in einem externen Schlüsselspeicher unterliegt Anforderungskontingenten. Überprüfen Sie diese Kontingente, bevor Sie Ihre externe Schlüsselspeicher-Implementierung entwerfen.

> **Note**
>
> Überprüfen Sie Ihre Konfiguration auf zirkuläre Abhängigkeiten, die möglicherweise verhindern, dass sie funktioniert.
> Wenn Sie beispielsweise Ihren externen Schlüsselspeicher-Proxy mithilfe von AWS Ressourcen erstellen, stellen Sie sicher, dass für den Betrieb des Proxys nicht die Verfügbarkeit eines KMS-Schlüssels in einem externen Schlüsselspeicher erforderlich ist, auf den über diesen Proxy zugegriffen wird.

Alle neuen externen Schlüsselspeicher werden in einem getrennten Zustand erstellt. Bevor Sie KMS-Schlüssel für Ihren externen Schlüsselspeicher erstellen können, müssen Sie ihn mit seinem externen Schlüsselspeicher-Proxy verbinden. Bearbeiten Sie die Einstellungen Ihres externen Schlüsselspeichers, um die Eigenschaften Ihres externen Schlüsselspeichers zu ändern.

Themen

- Erfüllen der Voraussetzungen
- Erstellen Sie einen neuen externen Schlüsselspeicher

Erfüllen der Voraussetzungen

Bevor Sie einen externen Schlüsselspeicher erstellen, müssen Sie die erforderlichen Komponenten zusammenstellen, einschließlich des externen Schlüsselmanagers, den Sie zur Unterstützung des externen Schlüsselspeichers verwenden werden, und des externen Schlüsselspeicher-Proxys, der AWS KMS Anfragen in ein Format übersetzt, das Ihr externer Schlüsselmanager verstehen kann.

Die folgenden Komponenten sind für alle externen Schlüsselspeicher erforderlich. Zusätzlich zu diesen Komponenten müssen Sie die Komponenten bereitstellen, die die von Ihnen gewählte Option der externen Schlüsselspeicher-Proxy-Konnektivität unterstützen.

> **Tip**
>
> Ihr externer Schlüsselmanager enthält möglicherweise einige dieser Komponenten, oder sie sind möglicherweise für Sie konfiguriert. Weitere Informationen finden Sie in der Dokumentation zum externen Schlüsselmanager.
> Wenn Sie Ihren externen Schlüsselspeicher in der AWS KMS Konsole erstellen, haben Sie die Möglichkeit, eine JSON-basierte Proxykonfigurationsdatei hochzuladen, die den Proxy-URI-Pfadund die Anmeldeinformationen für die Proxyauthentifizierung angibt. Einige externe Schlüsselspeicher-Proxys generieren diese Datei für Sie. Einzelheiten finden Sie in der Dokumentation für Ihren externen Schlüsselspeicher-Proxy oder externen Schlüsselmanager.

Externer Schlüsselmanager

Jeder externe Schlüsselspeicher benötigt mindestens eine externe Schlüsselmanager-Instance. Dies kann ein physisches oder virtuelles Hardwaresicherheitsmodul (HSM) oder eine Schlüsselverwaltungssoftware sein.

Sie können einen einzelnen Schlüsselmanager verwenden, wir empfehlen jedoch aus Redundanzgründen mindestens zwei verwandte Schlüsselmanager-Instances, die sich kryptografische Schlüssel teilen. Der externe Schlüsselspeicher erfordert keine ausschließliche Verwendung des externen Schlüsselmanagers. Der externe Schlüsselmanager muss jedoch in der Lage sein, die erwartete Häufigkeit von Verschlüsselungs- und Entschlüsselungsanforderungen von AWS Diensten zu verarbeiten, die KMS-Schlüssel im externen Schlüsselspeicher verwenden, um Ihre Ressourcen zu schützen. Ihr externer Schlüsselmanager sollte so konfiguriert sein, dass er bis zu 1 800 Anfragen pro Sekunde verarbeitet und innerhalb des Zeitlimits von 250 Millisekunden für jede Anforderung reagiert. Es wird empfohlen, den externen Schlüsselmanager in der Nähe von zu platzieren, AWS-Region sodass die Network Round-Trip Time (RTT) 35 Millisekunden oder weniger beträgt.

Wenn Ihr externer Schlüsselspeicher-Proxy dies zulässt, können Sie den externen Schlüsselmanager ändern, den Sie Ihrem externen Schlüsselspeicher-Proxy zuordnen. Der neue externe Schlüsselmanager muss jedoch ein Backup oder ein Snapshot mit demselben Schlüsselmaterial sein. Wenn der externe Schlüssel, den Sie einem KMS-Schlüssel zuordnen, für Ihren externen Schlüsselspeicher-Proxy nicht mehr verfügbar ist, AWS KMS kann der mit dem KMS-Schlüssel verschlüsselte Chiffretext nicht entschlüsselt werden.

Der externe Schlüsselmanager muss für den externen Schlüsselspeicher-Proxy zugänglich sein. Wenn in der GetHealthStatusAntwort des Proxys angegeben wird, dass alle externen Schlüsselmanager-Instanzen vorhanden sind`Unavailable`, schlagen alle Versuche, einen externen Schlüsselspeicher zu erstellen, mit einem fehl. `XksProxyUriUnreachableException`

Externer Schlüsselspeicher-Proxy

Sie müssen einen externen Schlüsselspeicher-Proxy (XKS-Proxy) angeben, der den Designanforderungen in der API-Spezifikation von AWS KMS für externe Schlüsselspeicher-Proxys entspricht. Sie können einen externen Schlüsselspeicher-Proxy entwickeln oder kaufen oder einen externen Schlüsselspeicher-Proxy verwenden, der von Ihrem externen Schlüsselmanager bereitgestellt oder in diesen integriert ist. AWS KMS empfiehlt, Ihren externen Schlüsselspeicher-Proxy so zu konfigurieren, dass er bis zu 1800 Anfragen pro Sekunde verarbeitet und innerhalb des Zeitlimits von 250 Millisekunden für jede Anfrage antwortet. Wir empfehlen, dass Sie den externen Schlüsselmanager in der Nähe von platzieren, AWS-Region sodass die Network Round-Trip Time (RTT) 35 Millisekunden oder weniger beträgt.

Sie können einen externen Schlüsselspeicher-Proxy für mehr als einen externen Schlüsselspeicher verwenden, aber jeder externe Schlüsselspeicher muss für seine Anforderungen einen eindeutigen URI-Endpunkt und -Pfad innerhalb des externen Schlüsselspeicher-Proxys haben.

Wenn Sie die Konnektivität eines VPC-Endpunkt-Service verwenden, können Sie Ihren externen Schlüsselspeicher-Proxy in Ihrer Amazon VPC lokalisieren, dies ist jedoch nicht erforderlich. Sie können Ihren Proxy außerhalb von AWS, z. B. in Ihrem privaten Rechenzentrum, lokalisieren und den VPC-Endpunktdienst nur für die Kommunikation mit dem Proxy verwenden.

Anmeldeinformationen für die Proxy-Authentifizierung

Um einen externen Schlüsselspeicher zu erstellen, müssen Sie Ihre Anmeldeinformationen für die Proxy-Authentifizierung für den externen Schlüsselspeicher (`XksProxyAuthenticationCredential`) angeben.

Sie müssen einen Authentifizierungsnachweis (`XksProxyAuthenticationCredential`) für Ihren externen AWS KMS Schlüsselspeicher-Proxy einrichten. AWS KMS authentifiziert sich bei Ihrem Proxy, indem es seine Anfragen mithilfe des Signature Version 4-Prozesses (Sigv4) mit den Anmeldeinformationen für die Proxyauthentifizierung des externen Schlüsselspeichers signiert. Sie geben die Anmeldeinformationen für die Authentifizierung an, wenn Sie Ihren externen Schlüsselspeicher erstellen, und Sie können sie jederzeit ändern. Wenn Ihr Proxy Ihre Anmeldeinformationen rotiert, müssen Sie die Anmeldeinformationswerte für Ihren externen Schlüsselspeicher aktualisieren.

Die Anmeldeinformationen für die Proxy-Authentifizierung bestehen aus zwei Teilen. Sie müssen beide Teile für Ihren externen Schlüsselspeicher bereitstellen.

- Zugriffsschlüssel-ID: Identifiziert den geheimen Zugriffsschlüssel. Sie können diese ID in Klartext angeben.
- Geheimer Zugriffsschlüssel: Der geheime Teil der Anmeldeinformationen. AWS KMS verschlüsselt den geheimen Zugriffsschlüssel in den Anmeldeinformationen, bevor er gespeichert wird.

Die SigV4-Anmeldeinformationen, die zum Signieren von Anfragen an den externen Schlüsselspeicher-Proxy AWS KMS verwendet werden, haben nichts mit SigV4-Anmeldeinformationen zu tun, die mit Prinzipalen in Ihren Konten verknüpft sind. AWS Identity and Access Management AWS Verwenden Sie keine IAM-SigV4-Anmeldeinformationen für Ihren externen Schlüsselspeicher-Proxy wieder.

Proxy-Konnektivität

Sie müssen die Proxy-Konnektivitätsoption des externen Schlüsselspeichers angeben (`XksProxyConnectivity`), um einen externen Schlüsselspeicher zu erstellen.

AWS KMS kann über einen öffentlichen Endpunkt oder einen Amazon Virtual Private Cloud (Amazon VPC) -Endpunktservice mit Ihrem externen Schlüsselspeicher-Proxy kommunizieren. Ein öffentlicher Endpunkt ist zwar einfacher zu konfigurieren und zu verwalten, erfüllt jedoch möglicherweise nicht die Sicherheitsanforderungen für jede Installation. Wenn Sie sich für die Option der Konnektivität des Amazon VPC-Endpunkt-Service entscheiden, müssen Sie die erforderlichen Komponenten erstellen und verwalten, einschließlich einer Amazon VPC mit mindestens zwei Subnetzen in zwei verschiedenen Availability Zones, einem VPC-Endpunkt-Service mit einem Network Load Balancer und einer Zielgruppe sowie einem privaten DNS-Namen für den VPC-Endpunkt-Service.

Für Ihren externen Schlüsselspeicher können Sie die Proxy-Konnektivitätsoption ändern. Sie müssen jedoch sicherstellen, dass das mit den KMS-Schlüsseln verbundene Schlüsselmaterial in Ihrem externen Schlüsselspeicher weiterhin verfügbar ist. Andernfalls AWS KMS kann kein mit diesen KMS-Schlüsseln verschlüsselter Chiffretext entschlüsselt werden.

Hilfe bei der Entscheidung, welche Proxy-Konnektivitätsoption für Ihren externen Schlüsselspeicher am besten geeignet ist, finden Sie unter Wählen Sie eine Proxy-Konnektivitätsoption für externen Schlüsselspeicher. Hilfe beim Erstellen und Konfigurieren der Konnektivität eines VPC-Endpunkt-Service finden Sie unter VPC-Endpunktdienstkonnektivität konfigurieren.

Proxy-URI-Endpunkt

Um einen externen Schlüsselspeicher zu erstellen, müssen Sie den Endpunkt (`XksProxyUriEndpoint`) angeben, über den Anfragen an AWS KMS den externen Schlüsselspeicher-Proxy gesendet werden.

Das Protokoll muss HTTPS sein. AWS KMS kommuniziert IPv4 über Port 443. Geben Sie den Port nicht im Wert des Proxy-URI-Endpunkts an.

- Konnektivität eines öffentlichen Endpunkts: Geben Sie den öffentlich verfügbaren Endpunkt für Ihren externen Schlüsselspeicher-Proxy an. Dieser Endpunkt muss erreichbar sein, bevor Sie Ihren externen Schlüsselspeicher erstellen.
- Konnektivität eines VPC-Endpunkt-Service: Geben Sie `https://` gefolgt von dem privaten DNS-Namen des VPC-Endpunkt-Service an.

Das TLS-Serverzertifikat, das auf dem externen Schlüsselspeicher-Proxy konfiguriert ist, muss mit dem Domainnamen im URI-Endpunkt des externen Schlüsselspeicher-Proxy übereinstimmen und von einer Zertifizierungsstelle ausgestellt sein, die für externe Schlüsselspeicher unterstützt wird. Eine Liste finden Sie unter Vertrauenswürdige Zertifizierungsstellen. Ihre Zertifizierungsstelle verlangt einen Nachweis über den Besitz der Domain, bevor sie das TLS-Zertifikat ausstellt.

Der Subject Common Name (CN) auf dem TLS-Zertifikat muss mit dem privaten DNS-Namen übereinstimmen. Wenn der private DNS-Name beispielsweise `myproxy-private.xks.example.com` lautet, muss der CN auf dem TLS-Zertifikat `myproxy-private.xks.example.com` oder `*.xks.example.com` lauten.

Sie können Ihren Proxy-URI-Endpunkt ändern, aber stellen Sie sicher, dass der externe Schlüsselspeicher-Proxy Zugriff auf das Schlüsselmaterial hat, das mit den KMS-Schlüsseln in Ihrem externen Schlüsselspeicher verbunden ist. Andernfalls AWS KMS kann kein mit diesen KMS-Schlüsseln verschlüsselter Chiffretext entschlüsselt werden.

Anforderungen an die Eindeutigkeit

- Der kombinierte Wert für den Proxy-URI-Endpunkt (`XksProxyUriEndpoint`) und den Proxy-URI-Pfad (`XksProxyUriPath`) muss in dem AWS-Konto und der Region eindeutig sein.
- Externe Schlüsselspeicher mit der Konnektivität eines öffentlichen Endpunkts können denselben Proxy-URI-Endpunkt verwenden, sofern sie unterschiedliche Proxy-URI-Pfadwerte haben.
- Ein externer Schlüsselspeicher mit öffentlicher Endpunktkonnektivität kann nicht denselben Proxy-URI-Endpunktwert verwenden wie jeder externe Schlüsselspeicher mit VPC-Endpunktdienstkonnektivität in demselben AWS-Region, auch wenn sich die Schlüsselspeicher in unterschiedlichen AWS-Konten Speichern befinden.
- Jeder externe Schlüsselspeicher mit der Konnektivität eines VPC-Endpunkts muss seinen eigenen privaten DNS-Namen haben. Der Proxy-URI-Endpunkt (privater DNS-Name) muss in der Region AWS-Konto und eindeutig sein.

Proxy-URI-Pfad

Um einen externen Schlüsselspeicher zu erstellen, müssen Sie den Basispfad in Ihrem externen Schlüsselspeicher-Proxy zum erforderlichen Proxy angeben APIs. Der Wert muss mit/kms/xks/v1 beginnen `/` und enden, wobei `v1` es sich um die Version der AWS KMS API für den externen Schlüsselspeicher-Proxy handelt. Dieser Pfad kann ein optionales Präfix zwischen den erforderlichen Elementen enthalten, z. B. `/example-prefix/kms/xks/v1`. Diesen Wert finden Sie in der Dokumentation für Ihren externen Schlüsselspeicher-Proxy.

AWS KMS sendet Proxyanfragen an die Adresse, die durch die Verkettung des Proxy-URI-Endpunkts und des Proxy-URI-Pfads angegeben wurde. Wenn beispielsweise der Proxy-URI-Endpunkt `https://myproxy.xks.example.com` und der Proxy-URI-Pfad sind`/kms/xks/v1`, AWS KMS sendet die Proxy-API-Anfragen an. `https://myproxy.xks.example.com/kms/xks/v1`

Sie können Ihren Proxy-URI-Pfad ändern, aber stellen Sie sicher, dass der externe Schlüsselspeicher-Proxy Zugriff auf das Schlüsselmaterial hat, das mit den KMS-Schlüsseln in Ihrem externen Schlüsselspeicher verbunden ist. Andernfalls AWS KMS kann kein mit diesen KMS-Schlüsseln verschlüsselter Chiffretext entschlüsselt werden.

Anforderungen an die Eindeutigkeit

- Der kombinierte Wert für den Proxy-URI-Endpunkt (`XksProxyUriEndpoint`) und den Proxy-URI-Pfad (`XksProxyUriPath`) muss in dem AWS-Konto und der Region eindeutig sein.

VPC-Endpunktservice

Gibt den Namen des Amazon VPC-Endpunkt-Services an, der für die Kommunikation mit dem externen Schlüsselspeicher-Proxy verwendet wird. Diese Komponente ist nur für externe Schlüsselspeicher erforderlich, die die Konnektivität eines VPC-Endpunkt-Service verwenden. Hilfe zum Einrichten und Konfigurieren Ihres VPC-Endpunkt-Services für einen externen Schlüsselspeicher finden Sie unter VPC-Endpunktdienstkonnektivität konfigurieren.

Der VPC-Endpunkt-Service muss über die folgenden Eigenschaften verfügen:

- Der VPC-Endpunktdienst muss sich in derselben AWS-Konto Region wie der externe Schlüsselspeicher befinden.
- Er muss über einen Network Load Balancer (NLB) verfügen, der mit mindestens zwei Subnetzen verbunden ist, die jeweils in einer anderen Availability Zone liegen.
- Die Liste der zugelassenen Prinzipale für den VPC-Endpunktdienst muss den AWS KMS Dienstprinzipal für die Region enthalten:`cks.kms.<region>.amazonaws.com`, z. B. `cks.kms.us-east-1.amazonaws.com`
- Er darf keine Annahme von Verbindungsanforderungen verlangen.
- Er muss einen privaten DNS-Namen innerhalb einer öffentlichen Domain höherer Ebene haben. Sie könnten beispielsweise den privaten DNS-Namen „myproxy-private.xks.example.com“ in der öffentlichen Domain `xks.example.com` haben.

  Der private DNS-Name für einen externen Schlüsselspeicher mit der Konnektivität eines VPC-Endpunkt-Service muss in seiner AWS-Region eindeutig sein.
- Der Domain-Verifizierungsstatus der privaten DNS-Namensdomain muss `verified` lauten.
- Das auf dem externen Schlüsselspeicher-Proxy konfigurierte TLS-Serverzertifikat muss den privaten DNS-Hostnamen angeben, unter dem der Endpunkt erreichbar ist.

Anforderungen an die Eindeutigkeit

- Externe Schlüsselspeicher mit der Konnektivität eines VPC-Endpunkts können sich eine `Amazon VPC` teilen, aber jeder externe Schlüsselspeicher muss seinen eigenen VPC-Endpunkt-Service und privaten DNS-Namen haben.

Proxy-Konfigurationsdatei

Eine Proxy-Konfigurationsdatei ist eine optionale JSON-basierte Datei, die Werte für den Proxy-URI-Pfad und die Eigenschaften der Proxy-Authentifizierungsanmeldeinformation Ihres externen Schlüsselspeichers enthält. Beim Erstellen oder Bearbeiten eines externen Schlüsselspeichers in der AWS KMS -Konsole können Sie eine Proxy-Konfigurationsdatei hochladen, um Konfigurationswerte für Ihren externen Schlüsselspeicher bereitzustellen. Durch die Verwendung dieser Datei werden Tipp- und Einfügefehler verhindert und es wird sichergestellt, dass die Werte in Ihrem externen Schlüsselspeicher mit den Werten in Ihrem externen Schlüsselspeicher-Proxy übereinstimmen.

Die Proxy-Konfigurationsdateien werden vom externen Schlüsselspeicher-Proxy generiert. Ob Ihr externer Schlüsselspeicher-Proxy eine Proxy-Konfigurationsdatei anbietet, können Sie der Dokumentation Ihres externen Schlüsselspeicher-Proxys entnehmen.

Im Folgenden finden Sie ein Beispiel für eine gültige Proxy-Konfigurationsdatei mit fiktiven Werten.

```
{
  "XksProxyUriPath": "/example-prefix/kms/xks/v1",
  "XksProxyAuthenticationCredential": {
    "AccessKeyId": "ABCDE12345670EXAMPLE",
    "RawSecretAccessKey": "0000EXAMPLEFA5FT0mCc3DrGUe2sti527BitkQ0Zr9MO9+vE="
  }
}
```

Sie können eine Proxy-Konfigurationsdatei nur hochladen, wenn Sie einen externen Schlüsselspeicher in der AWS KMS Konsole erstellen oder bearbeiten. Sie können sie nicht mit den UpdateCustomKeyStoreOperationen CreateCustomKeyStoreoder verwenden, aber Sie können die Werte in der Proxykonfigurationsdatei verwenden, um sicherzustellen, dass Ihre Parameterwerte korrekt sind.

Erstellen Sie einen neuen externen Schlüsselspeicher

Sobald Sie die erforderlichen Voraussetzungen zusammengestellt haben, können Sie in der AWS KMS Konsole oder mithilfe des CreateCustomKeyStoreVorgangs einen neuen externen Schlüsselspeicher erstellen.

Mithilfe der AWS KMS Konsole

Bevor Sie einen externen Schlüsselspeicher erstellen, wählen Sie Ihren Proxy-Konnektivitätstyp und stellen Sie sicher, dass Sie alle erforderlichen Komponenten erstellt und konfiguriert haben. Wenn Sie Hilfe bei der Suche nach einem der erforderlichen Werte benötigen, ziehen Sie die Dokumentation für Ihren externen Schlüsselspeicher-Proxy oder Ihre Schlüsselverwaltungssoftware zu Rate.

> **Note**
>
> Wenn Sie in der einen externen Schlüsselspeicher erstellen AWS Management Console, können Sie eine JSON-basierte Proxykonfigurationsdatei mit Werten für den Proxy-URI-Pfad und die Anmeldeinformationen für die Proxyauthentifizierung hochladen. Einige Proxys generieren diese Datei für Sie. Sie ist nicht erforderlich.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter /kms. https://console.aws.amazon.com
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (Benutzerdefinierte Schlüsselspeicher), External key stores (Externe Schlüsselspeicher) aus.
4. Wählen Sie Create external key store (Erstellen eines externen Schlüsselspeichers) aus.
5. Geben Sie einen Anzeigenamen für den externen Schlüsselspeicher ein. Der Name muss unter allen externen Schlüsselspeichern in Ihrem Konto eindeutig sein.

   > **Important**
   >
   > Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

6. Wählen Sie Ihren Proxy-Konnektivitätstyp aus.

Die von Ihnen gewählte Proxy-Konnektivität bestimmt die für Ihren externen Schlüsselspeicher-Proxy erforderlichen Komponenten. Hilfe bei dieser Auswahl finden Sie unter Wählen Sie eine Proxy-Konnektivitätsoption für externen Schlüsselspeicher.

7. Wählen Sie den Namen des VPC-Endpunkt-Service für diesen externen Schlüsselspeicher aus oder geben Sie ihn ein. Dieser Schritt wird nur angezeigt, wenn der Proxy-Konnektivitätstyp des externen Schlüsselspeichers VPC-Endpunkt-Service ist.

   Der VPC-Endpunktdienst und sein VPCs müssen die Anforderungen für einen externen Schlüsselspeicher erfüllen. Details hierzu finden Sie unter the section called “Erfüllen der Voraussetzungen”.

8. Geben Sie Ihren Proxy-URI-Endpunkt ein. Das Protokoll muss HTTPS sein. AWS KMS kommuniziert IPv4 über Port 443. Geben Sie den Port nicht im Wert des Proxy-URI-Endpunkts an.

   Wenn der VPC-Endpunktdienst AWS KMS erkannt wird, den Sie im vorherigen Schritt angegeben haben, füllt er dieses Feld für Sie aus.

   Für die Konnektivität eines öffentlichen Endpunkts geben Sie einen öffentlich verfügbaren Endpunkt-URI ein. Für die Konnektivität des VPC-Endpunkts geben Sie `https://` gefolgt vom privaten DNS-Namen des VPC-Endpunkt-Service ein.

9. Um die Werte für das Präfix des Proxy-URI-Pfads und die Proxy-Authentifizierungsanmeldeinformation einzugeben, laden Sie eine Proxy-Konfigurationsdatei hoch oder geben Sie die Werte manuell ein.

   - Wenn Sie über eine optionale Proxy-Konfigurationsdatei verfügen, die Werte für den Proxy-URI-Pfad und die Proxy-Authentifizierungsanmeldeinformation enthält, wählen Sie Upload configuration file (Konfigurationsdatei hochladen) aus. Folgen Sie den Schritten zum Hochladen der Datei.

     Wenn die Datei hochgeladen ist, zeigt die Konsole die Werte aus der Datei in bearbeitbaren Feldern an. Sie können die Werte jetzt ändern oder diese Werte bearbeiten, nachdem der externe Schlüsselspeicher erstellt wurde.

     Wählen Sie Show secret access key (Geheimen Zugriffsschlüssel anzeigen) aus, um den Wert des geheimen Zugriffsschlüssels anzuzeigen.

   - Wenn Sie über keine Proxy-Konfigurationsdatei verfügen, können Sie den Proxy-URI-Pfad und die Werte für die Proxy-Authentifizierungsanmeldeinformation manuell eingeben.

a. Wenn Sie nicht über eine Proxy-Konfigurationsdatei verfügen, können Sie den Proxy-URI manuell eingeben. Die Konsole stellt den erforderlichen Wert/kms/xks/v1 bereit.

   Wenn Ihr Proxy-URI-Pfad ein optionales Präfix enthält, wie z. B. das `example-prefix` in `/example-prefix/kms/xks/v1`, geben Sie das Präfix in das Feld Proxy URI path prefix (Proxy-URI-Pfadpräfix) ein. Andernfalls lassen Sie das Feld leer.

b. Wenn Sie nicht über eine Proxy-Konfigurationsdatei verfügen, können Sie die Proxy-Authentifizierungsanmeldeinformation manuell eingeben. Sowohl die Zugriffsschlüssel-ID als auch der geheime Zugriffsschlüssel sind erforderlich.

   - Geben Sie in Proxy credential: Access key ID (Proxy-Anmeldeinformationen: Zugriffsschlüssel-ID) die Zugriffsschlüssel-ID der Proxy-Authentifizierungsanmeldeinformation ein. Die Zugriffsschlüssel-ID identifiziert den geheimen Zugriffsschlüssel.
   - Geben Sie in Proxy credential: Secret access key (Proxy-Anmeldeinformationen: Geheimer Zugriffsschlüssel) den geheimen Zugriffsschlüssel der Proxy-Authentifizierungsanmeldeinformation ein.

   Wählen Sie Show secret access key (Geheimen Zugriffsschlüssel anzeigen) aus, um den Wert des geheimen Zugriffsschlüssels anzuzeigen.

   Mit diesem Verfahren wird die Authentifizierungsanmeldeinformation, die Sie auf Ihrem externen Schlüsselspeicher-Proxy eingerichtet haben, weder festgelegt noch geändert. Es verknüpft diese Werte lediglich mit Ihrem externen Schlüsselspeicher. Informationen zum Einrichten, Ändern und Rotieren der Proxy-Authentifizierungsanmeldeinformation finden Sie in der Dokumentation zu Ihrem externen Schlüsselspeicher-Proxy oder Ihrer Schlüsselverwaltungssoftware.

   Wenn sich die Proxy-Authentifizierungsanmeldeinformation ändert, bearbeiten Sie die Einstellungen für die Anmeldeinformation für Ihren externen Schlüsselspeicher.

10. Wählen Sie Create external key store (Erstellen eines externen Schlüsselspeichers) aus.

Wenn der Vorgang erfolgreich war, wird der neue externe Schlüsselspeicher in der Liste der externen Schlüsselspeicher im Konto und in der Region angezeigt. Bei nicht erfolgreicher Ausführung wird eine Fehlermeldung mit einer Beschreibung des Problems und Hilfestellung zur Fehlerbehebung

angezeigt. Wenn Sie weitere Hilfe benötigen, beachten Sie den Abschnitt CreateKey Fehler für den externen Schlüssel.

Nächster Schritt: Neue externe Schlüsselspeicher werden nicht automatisch verbunden. Bevor Sie AWS KMS keys in Ihrem externen Schlüsselspeicher etwas erstellen können, müssen Sie den externen Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy verbinden.

Verwenden der AWS KMS API

Sie können den CreateCustomKeyStoreVorgang verwenden, um einen neuen externen Schlüsselspeicher zu erstellen. Hilfe bei der Suche nach den Werten für die erforderlichen Parameter finden Sie in der Dokumentation zu Ihrem externen Schlüsselspeicher-Proxy oder Ihrer Schlüsselverwaltungssoftware.

> **Tip**
>
> Sie können keine Proxy-Konfigurationsdatei hochladen, wenn Sie den `CreateCustomKeyStore`-Vorgang verwenden. Sie können jedoch die Werte in der Proxy-Konfigurationsdatei verwenden, um sicherzustellen, dass Ihre Parameterwerte korrekt sind.

Zum Erstellen eines externen Schlüsselspeichers sind für den `CreateCustomKeyStore`-Vorgang die folgenden Parameterwerte erforderlich:

- `CustomKeyStoreName`: Ein Anzeigename für den externen Schlüsselspeicher, der im Konto eindeutig ist.

  > **Important**
  >
  > Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.

- `CustomKeyStoreType`: Geben Sie `EXTERNAL_KEY_STORE` an.
- `XksProxyConnectivity`: Geben Sie `PUBLIC_ENDPOINT` oder `VPC_ENDPOINT_SERVICE` an.
- `XksProxyAuthenticationCredential`: Geben Sie sowohl die Zugriffsschlüssel-ID als auch den geheimen Zugriffsschlüssel an.
- `XksProxyUriEndpoint`— Der Endpunkt, der für die Kommunikation mit Ihrem externen Schlüsselspeicher-Proxy AWS KMS verwendet wird.

- `XksProxyUriPath`— Der Pfad innerhalb des Proxys zum Proxy APIs.
- `XksProxyVpcEndpointServiceName`: Nur erforderlich, wenn Ihr `XksProxyConnectivity`-Wert `VPC_ENDPOINT_SERVICE` ist.

> **Note**
>
> Wenn Sie AWS CLI Version 1.0 verwenden, führen Sie den folgenden Befehl aus, bevor Sie einen Parameter mit einem HTTP- oder HTTPS-Wert angeben, z. B. den `XksProxyUriEndpoint` Parameter.
>
> ```
> aws configure set cli_follow_urlparam false
> ```
>
> Andernfalls ersetzt AWS CLI Version 1.0 den Parameterwert durch den Inhalt, der an dieser URI-Adresse gefunden wurde, was den folgenden Fehler verursacht:
>
> ```
> Error parsing parameter '--xks-proxy-uri-endpoint': Unable to retrieve
> https:// : received non 200 status code of 404
> ```

In den folgenden Beispielen werden fiktive Werte verwendet. Bevor Sie den Befehl ausführen, ersetzen Sie sie durch gültige Werte für Ihren externen Schlüsselspeicher.

Erstellen Sie einen externen Schlüsselspeicher mit der Konnektivität eines öffentlichen Endpunkts.

```
$ aws kms create-custom-key-store
        --custom-key-store-name ExampleExternalKeyStorePublic \
        --custom-key-store-type EXTERNAL_KEY_STORE \
        --xks-proxy-connectivity PUBLIC_ENDPOINT \
        --xks-proxy-uri-endpoint https://myproxy.xks.example.com \
        --xks-proxy-uri-path /kms/xks/v1 \
        --xks-proxy-authentication-credential
 AccessKeyId=<value>,RawSecretAccessKey=<value>
```

Erstellen Sie einen externen Schlüsselspeicher mit der Konnektivität eines VPC-Endpunkt-Service.

```
$ aws kms create-custom-key-store
        --custom-key-store-name ExampleExternalKeyStoreVPC \
        --custom-key-store-type EXTERNAL_KEY_STORE \
```

```
        --xks-proxy-connectivity VPC_ENDPOINT_SERVICE \
        --xks-proxy-vpc-endpoint-service-name com.amazonaws.vpce.us-east-1.vpce-svc-
example \
        --xks-proxy-uri-endpoint https://myproxy-private.xks.example.com \
        --xks-proxy-uri-path /kms/xks/v1 \
        --xks-proxy-authentication-credential
 AccessKeyId=<value>,RawSecretAccessKey=<value>
```

Bei einer erfolgreichen Produktion gibt `CreateCustomKeyStore` die ID des benutzerdefinierten Schlüsselspeichers zurück, wie in der folgenden Beispielantwort dargestellt.

```
{
    "CustomKeyStoreId": cks-1234567890abcdef0
}
```

Wenn die Produktion fehlschlägt, korrigieren Sie den in der Ausnahme angegebenen Fehler und versuchen Sie es erneut. Weitere Informationen finden Sie unter Fehlerbehebung bei externen Schlüsselspeichern.

Nächster Schritt: Um den externen Schlüsselspeicher zu verwenden, verbinden Sie ihn mit dem externen Schlüsselspeicher-Proxy.

Eigenschaften des externen Schlüsselspeichers bearbeiten

Sie können ausgewählte Eigenschaften eines vorhandenen externen Schlüsselspeichers bearbeiten.

Sie können einige Eigenschaften bearbeiten, während der externe Schlüsselspeicher verbunden oder getrennt ist. Für andere Eigenschaften müssen Sie zuerst Ihren externen Schlüsselspeicher vom Proxy des externen Schlüsselspeichers trennen. Der Verbindungsstatus des externen Schlüsselspeichers muss `DISCONNECTED` sein. Während ein externer Schlüsselspeicher getrennt ist, können Sie den Schlüsselspeicher und seine KMS-Schlüssel verwalten, aber Sie können keine KMS-Schlüssel im externen Schlüsselspeicher erstellen oder verwenden. Um den Verbindungsstatus Ihres externen Schlüsselspeichers zu ermitteln, verwenden Sie den DescribeCustomKeyStoresVorgang oder lesen Sie den Abschnitt Allgemeine Konfiguration auf der Detailseite für den externen Schlüsselspeicher.

Bevor Sie die Eigenschaften Ihres externen Schlüsselspeichers aktualisieren, AWS KMS sendet er unter Verwendung der neuen Werte eine GetHealthStatusAnfrage an den externen Schlüsselspeicher-Proxy. Wenn die Anforderung erfolgreich ist, bedeutet dies, dass Sie eine

Verbindung zu einem Proxy des externen Schlüsselspeichers herstellen und sich mit den aktualisierten Eigenschaftswerten authentifizieren können. Schlägt die Anforderung fehl, schlägt der Bearbeitungsvorgang mit einer Ausnahme fehl, die den Fehler identifiziert.

Wenn der Bearbeitungsvorgang abgeschlossen ist, werden die aktualisierten Eigenschaftswerte für Ihren externen Schlüsselspeicher in der AWS KMS Konsole und in der `DescribeCustomKeyStores` Antwort angezeigt. Es kann jedoch bis zu fünf Minuten dauern, bis die Änderungen vollständig wirksam werden.

Wenn Sie Ihren externen Schlüsselspeicher in der AWS KMS Konsole bearbeiten, haben Sie die Möglichkeit, eine JSON-basierte Proxy-Konfigurationsdatei hochzuladen, die den Proxy-URI-Pfad und die Anmeldeinformationen für die Proxyauthentifizierung angibt. Einige externe Schlüsselspeicher-Proxys generieren diese Datei für Sie. Einzelheiten finden Sie in der Dokumentation für Ihren externen Schlüsselspeicher-Proxy oder externen Schlüsselmanager.

> **Warning**
>
> Die aktualisierten Eigenschaftswerte müssen Ihren externen Schlüsselspeicher mit einem Proxy für denselben externen Schlüsselmanager wie die vorherigen Werte oder für eine Sicherung oder einen Snapshot des externen Schlüsselmanagers mit denselben kryptografischen Schlüsseln verbinden. Wenn Ihr externer Schlüsselspeicher dauerhaft den Zugriff auf die externen Schlüssel verliert, die mit seinen KMS-Schlüsseln verknüpft sind, kann der mit diesen externen Schlüsseln verschlüsselte Geheimtext nicht wiederhergestellt werden. Insbesondere kann durch eine Änderung der Proxykonnektivität eines externen Schlüsselspeichers der Zugriff auf Ihre externen Schlüssel AWS KMS verhindert werden.

> **Tip**
>
> Einige externe Schlüsselmanager bieten eine einfachere Methode zur Bearbeitung der Eigenschaften externer Schlüsselspeicher. Weitere Informationen finden Sie in der Dokumentation zum externen Schlüsselmanager.

Sie können die folgenden Eigenschaften eines externen Schlüsselspeichers ändern:

Bearbeitbare Eigenschaften eines externen Schlüsselspeichers	Beliebiger Verbindungsstatus	Status „Disconnected“ (Verbindung getrennt) erforderlich
Name des benutzerdefinierten Schlüsselspeichers Ein erforderlicher Anzeigename für einen benutzerdefinierten Schlüsselspeicher. **Important** Geben Sie keine vertraulichen oder sensiblen Informationen in dieses Feld ein. Dieses Feld kann in CloudTrail Protokollen und anderen Ausgaben im Klartext angezeigt werden.		
Anmeldeinformationen für die Proxyauthentifizierung () XksProxyAuthenticationCredential (Sie müssen sowohl die Zugriffsschlüssel-ID als auch den geheimen Zugriffsschlüssel angeben, auch wenn Sie nur ein Element ändern.)		
Proxy-URI-Pfad () XksProxyUriPath		
Proxy-Konnektivität (XksProxyConnectivity) (Sie müssen auch den Proxy-URI-Endpunkt aktualisieren. Wenn Sie zur Konnektivität eines VPC-Endpunkt-Service wechseln, müssen Sie einen Namen für den Proxy-VPC-Endpunkt-Service angeben.)		
Proxy-URI-Endpunkt (XksProxyUriEndpoint)		

Bearbeitbare Eigenschaften eines externen Schlüsselspeichers	Beliebiger Verbindungsstatus	Status „Disconnected“ (Verbindung getrennt) erforderlich
Wenn Sie den Proxy-Endpunkt-URI ändern, müssen Sie möglicherweise auch das zugehörige TLS-Zertifikat ändern.		
Proxy-VPC-Endpunktdienstname () XksProxyVpcEndpointServiceName (Dieses Feld ist für die Konnektivität eines VPC-Endpunkt-Service erforderlich)		

Bearbeiten Sie die Eigenschaften Ihres externen Schlüsselspeichers

Sie können die Eigenschaften Ihres externen Schlüsselspeichers in der AWS KMS Konsole oder mithilfe des UpdateCustomKeyStoreVorgangs bearbeiten.

Mithilfe der AWS KMS Konsole

Wenn Sie einen Schlüsselspeicher bearbeiten, können Sie die bearbeitbaren Werte beliebig ändern. Einige Änderungen erfordern, dass der externe Schlüsselspeicher von seinem externen Schlüsselspeicher-Proxy getrennt wird.

Wenn Sie den Proxy-URI-Pfad oder die Anmeldeinformation für die Proxy-Authentifizierung bearbeiten, können Sie die neuen Werte eingeben oder eine Proxykonfigurationsdatei für den externen Schlüsselspeicher hochladen, die die neuen Werte enthält.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (Benutzerdefinierte Schlüsselspeicher), External key stores (Externe Schlüsselspeicher) aus.
4. Wählen Sie die Zeile des externen Schlüsselspeichers aus, den Sie bearbeiten möchten.

5. Falls erforderlich, trennen Sie den externen Schlüsselspeicher von seinem externen Schlüsselspeicher-Proxy. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Disconnect (Verbindung trennen) aus.
6. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Edit (Bearbeiten) aus.
7. Ändern Sie eine oder mehrere der bearbeitbaren Eigenschaften des externen Schlüsselspeichers. Sie können auch eine Proxy-Konfigurationsdatei für den externen Schlüsselspeicher mit Werten für den Proxy-URI-Pfad und der Anmeldeinformation für die Proxy-Authentifizierung hochladen. Sie können eine Proxy-Konfigurationsdatei auch dann verwenden, wenn sich einige in der Datei angegebene Werte nicht geändert haben.
8. Wählen Sie Update external key store (Aktualisieren des externen Schlüsselspeichers).
9. Überprüfen Sie die Warnung, und wenn Sie fortfahren möchten, bestätigen Sie die Warnung und wählen Sie dann Update external key store (Aktualisieren des externen Schlüsselspeichers).

   Wenn der Vorgang erfolgreich ist, wird eine Meldung mit einer Beschreibung der von Ihnen bearbeiteten Eigenschaften angezeigt. Wenn der Vorgang nicht erfolgreich ist, wird Ihnen eine Fehlermeldung mit einer Beschreibung des Problems und Hilfestellung zur Fehlerbehebung angezeigt.
10. Falls erforderlich, verbinden Sie den externen Schlüsselspeicher wieder. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Connect (Verbinden) aus.

    Sie können die Verbindung des externen Schlüsselspeichers getrennt lassen. Solange die Verbindung getrennt ist, können Sie jedoch keine KMS-Schlüssel im externen Schlüsselspeicher erstellen oder die KMS-Schlüssel im externen Schlüsselspeicher für kryptografische Vorgänge verwenden.

Verwenden der API AWS KMS

Verwenden Sie den UpdateCustomKeyStoreVorgang, um die Eigenschaften eines externen Schlüsselspeichers zu ändern. Sie können mehrere Eigenschaften eines externen Schlüsselspeichers mit demselben Vorgang ändern. Wenn der Vorgang erfolgreich ist, werden eine HTTP 200-Antwort und ein JSON-Objekt ohne Eigenschaften AWS KMS zurückgegeben.

Verwenden Sie den `CustomKeyStoreId`-Parameter, um den externen Schlüsselspeicher zu identifizieren. Verwenden Sie die anderen Parameter, um die Eigenschaften zu ändern. Sie können für den `UpdateCustomKeyStore`-Vorgang keine Proxy-Konfigurationsdatei verwenden. Die Proxy-Konfigurationsdatei wird nur von der AWS KMS Konsole unterstützt. Sie können aber

die Proxy-Konfigurationsdatei verwenden, um die richtigen Parameterwerte für Ihren externen Schlüsselspeicher-Proxy zu ermitteln.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Bevor Sie beginnen, trennen Sie ggf. die Verbindung des externen Schlüsselspeichers mit dem Proxy des externen Schlüsselspeichers. Nach der Aktualisierung können Sie den externen Schlüsselspeicher bei Bedarf wieder mit dem Proxy des externen Schlüsselspeichers verbinden. Sie können den externen Schlüsselspeicher im getrennten Zustand belassen, müssen ihn aber wieder verbinden, bevor Sie neue KMS-Schlüssel im Schlüsselspeicher erstellen oder vorhandene KMS-Schlüssel im Schlüsselspeicher für kryptografische Vorgänge verwenden können.

> **Note**
>
> Wenn Sie AWS CLI Version 1.0 verwenden, führen Sie den folgenden Befehl aus, bevor Sie einen Parameter mit einem HTTP- oder HTTPS-Wert angeben, z. B. den `XksProxyUriEndpoint` Parameter.
>
> ```
> aws configure set cli_follow_urlparam false
> ```
>
> Andernfalls ersetzt AWS CLI Version 1.0 den Parameterwert durch den Inhalt, der an dieser URI-Adresse gefunden wurde, was den folgenden Fehler verursacht:
>
> ```
> Error parsing parameter '--xks-proxy-uri-endpoint': Unable to retrieve
> https:// : received non 200 status code of 404
> ```

Ändern des Namens des externen Schlüsselspeichers

Das erste Beispiel verwendet die UpdateCustomKeyStoreOperation, um den Anzeigenamen des externen Schlüsselspeichers in zu ändern`XksKeyStore`. Der Befehl verwendet den Parameter `CustomKeyStoreId`, um den benutzerdefinierten Schlüsselspeicher zu identifizieren, und den Parameter `CustomKeyStoreName`, um den neuen Namen für den benutzerdefinierten Schlüsselspeicher anzugeben. Ersetzen Sie alle Beispielwerte durch tatsächliche Werte für Ihren externen Schlüsselspeicher.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --new-
custom-key-store-name XksKeyStore
```

Ändern der Anmeldeinformation für die Proxy-Authentifizierung

Im folgenden Beispiel werden die Anmeldeinformationen für die Proxyauthentifizierung aktualisiert, die für die Authentifizierung beim externen Schlüsselspeicher-Proxy AWS KMS verwendet werden. Sie können einen Befehl wie diesen verwenden, um die Anmeldeinformation zu aktualisieren, wenn sie auf Ihrem Proxy rotiert werden.

Aktualisieren Sie zuerst die Anmeldeinformation auf Ihrem externen Schlüsselspeicher-Proxy. Verwenden Sie dann dieses Feature, um die Änderung an AWS KMS zu melden. (Ihr Proxy unterstützt kurzzeitig sowohl die alten als auch die neuen Anmeldeinformationen, sodass Sie Zeit haben, Ihre Anmeldeinformationen zu aktualisieren.) AWS KMS

Sie müssen immer sowohl die Zugriffsschlüssel-ID als auch den geheimen Zugriffsschlüssel in der Anmeldeinformation angeben, auch wenn nur ein Wert geändert wird.

Die ersten beiden Befehle legen Variablen fest, die die Anmeldeinformationswerte enthalten. Der Vorgang `UpdateCustomKeyStore` verwendet den Parameter `CustomKeyStoreId`, um den externen Schlüsselspeicher zu identifizieren. Er verwendet den Parameter `XksProxyAuthenticationCredential` mit seinen Feldern `AccessKeyId` und `RawSecretAccessKey`, um die neue Anmeldeinformation festzulegen. Ersetzen Sie alle Beispielwerte durch tatsächliche Werte für Ihren externen Schlüsselspeicher.

```
$ accessKeyID=access key id
$ secretAccessKey=secret access key

$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
        --xks-proxy-authentication-credential \
            AccessKeyId=$accessKeyId,RawSecretAccessKey=$secretAccessKey
```

Ändern des Proxy-URI-Pfads

Im folgenden Beispiel wird der Proxy-URI-Pfad (`XksProxyUriPath`) aktualisiert. Die Kombination aus dem Proxy-URI-Endpunkt und dem Proxy-URI-Pfad muss für das AWS-Konto und die Region eindeutig sein. Ersetzen Sie alle Beispielwerte durch tatsächliche Werte für Ihren externen Schlüsselspeicher.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
```

```
            --xks-proxy-uri-path /kms/xks/v1
```

Umstellen auf Konnektivität eines VPC-Endpunkt-Service

Im folgenden Beispiel wird der UpdateCustomKeyStoreVorgang verwendet, um den Proxykonnektivitätstyp des externen Schlüsselspeichers auf zu ändern. `VPC_ENDPOINT_SERVICE` Um diese Änderung vorzunehmen, müssen Sie die erforderlichen Werte für die Konnektivität eines VPC-Endpunkt-Service angeben, einschließlich des Namens des VPC-Endpunktservice (`XksProxyVpcEndpointServiceName`) und eines Proxy-URI-Endpunktwerts (`XksProxyUriEndpoint`), der den privaten DNS-Namen für den VPC-Endpunktservice enthält. Ersetzen Sie alle Beispielwerte durch tatsächliche Werte für Ihren externen Schlüsselspeicher.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
            --xks-proxy-connectivity "VPC_ENDPOINT_SERVICE" \
            --xks-proxy-uri-endpoint https://myproxy-private.xks.example.com \
            --xks-proxy-vpc-endpoint-service-name com.amazonaws.vpce.us-east-1.vpce-
svc-example
```

Umstellen auf Konnektivität eines öffentlichen Endpunkts

Im folgenden Beispiel wird der Proxy-Konnektivitätstyp des externen Schlüsselspeichers in `PUBLIC_ENDPOINT` geändert. Wenn Sie diese Änderung vornehmen, müssen Sie den Wert des Proxy-URI-Endpunkts (`XksProxyUriEndpoint`) aktualisieren. Ersetzen Sie alle Beispielwerte durch tatsächliche Werte für Ihren externen Schlüsselspeicher.

> **Note**
>
> VPC-Endpunktkonnektivität bietet mehr Sicherheit als öffentliche Endpunktkonnektivität. Bevor Sie auf öffentliche Endpunktkonnektivität umstellen, sollten Sie andere Optionen in Betracht ziehen, z. B. den Proxy für den externen Schlüsselspeicher On-Premises zu platzieren und die VPC nur für die Kommunikation zu nutzen.

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
            --xks-proxy-connectivity "PUBLIC_ENDPOINT" \
            --xks-proxy-uri-endpoint https://myproxy.xks.example.com
```

Externe Schlüsselspeicher anzeigen

Sie können externe Schlüsselspeicher in jedem Konto und jeder Region anzeigen, indem Sie die AWS KMS Konsole oder den DescribeCustomKeyStoresVorgang verwenden.

Beim Anzeigen eines externen Schlüsselspeichers können Sie folgende Informationen sehen:

- Grundlegende Informationen über den Schlüsselspeicher, einschließlich seines Benutzernamens, seiner ID, seines Schlüsselspeichertyps und seines Erstellungsdatums
- Konfigurationsinformationen für den Proxy des externen Schlüsselspeichers, einschließlich des Konnektivitätstyps, des Proxy-URI-Endpunkts und -pfads sowie der Zugriffsschlüssel-ID Ihrer aktuellen Proxy-Anmeldeinformation.
- Wenn der Proxy des externen Schlüsselspeichers die Konnektivität des VPC-Endpunkt-Service verwendet, zeigt die Konsole den Namen des VPC-Endpunktservice an.
- Den aktuellen Verbindungsstatus

> **Note**
>
> Der Verbindungsstatus Disconnected (Verbindung getrennt) bedeutet, dass der externe Schlüsselspeicher noch nie verbunden war oder dass die Verbindung zum Proxy des externen Schlüsselspeichers absichtlich getrennt wurde. Wenn Ihre Versuche, einen KMS-Schlüssel in einem verbundenen externen Schlüsselspeicher zu verwenden, jedoch fehlschlagen, kann dies auf ein Problem mit dem externen Schlüsselspeicher oder seinem Proxy hinweisen. Weitere Informationen dazu finden Sie unter Fehler bei der Verbindung mit dem externen Schlüsselspeicher.

- Ein Monitoring-Bereich mit Grafiken von CloudWatch Amazon-Metriken, die Ihnen helfen sollen, Probleme mit Ihrem externen Schlüsselspeicher zu erkennen und zu lösen. Hilfe bei der Interpretation der Grafiken, ihrer Verwendung bei der Planung und Problembehebung und der Erstellung von CloudWatch Alarmen auf der Grundlage der Kennzahlen in den Diagrammen finden Sie unterÜberwachen Sie externe Schlüsselspeicher.

Eigenschaften des externen Schlüsselspeichers

Die folgenden Eigenschaften eines externen Schlüsselspeichers sind in der AWS KMS Konsole und in der DescribeCustomKeyStoresAntwort sichtbar.

Eigenschaften eines benutzerdefinierten Schlüsselspeichers

Die folgenden Werte werden im Abschnitt Allgemeine Konfiguration der Detailseite für jeden benutzerdefinierten Schlüsselspeicher angezeigt. Diese Eigenschaften gelten für alle benutzerdefinierten Schlüsselspeicher, einschließlich AWS CloudHSM Schlüsselspeicher und externer Schlüsselspeicher.

ID des benutzerdefinierten Schlüsselspeichers

Eine eindeutige ID, die AWS KMS dem benutzerdefinierten Schlüsselspeicher zugewiesen wird.

Name des benutzerdefinierten Schlüsselspeichers

Ein Anzeigename, den Sie dem benutzerdefinierten Schlüsselspeicher bei dessen Erstellung zuweisen. Sie können diesen Wert jederzeit ändern.

Typ des benutzerdefinierten Schlüsselspeichers

Der Typ des benutzerdefinierten Schlüsselspeichers. Gültige Werte sind AWS CloudHSM (`AWS_CLOUDHSM`) oder „Externer Schlüsselspeicher“ (`EXTERNAL_KEY_STORE`). Der Typ kann nach der Erstellung des benutzerdefinierten Schlüsselspeichers nicht mehr geändert werden.

Erstellungsdatum

Das Datum, an dem der benutzerdefinierte Schlüsselspeicher erstellt wurde. Dieses Datum wird in Ortszeit für die AWS-Region angezeigt.

Verbindungsstatus

Zeigt an, ob der benutzerdefinierte Schlüsselspeicher mit dem Unterstützungsschlüsselspeicher verbunden ist. Der Verbindungsstatus ist nur dann `DISCONNECTED`, wenn der benutzerdefinierte Schlüsselspeicher noch nie mit dem Unterstützungsschlüsselspeicher verbunden war oder die Verbindung absichtlich getrennt wurde. Details hierzu finden Sie unter the section called “Verbindungsstatus”.

Konfigurationseigenschaften des externen Schlüsselspeichers

Die folgenden Werte werden im Abschnitt Konfiguration des externen Schlüsselspeichers auf der Detailseite für jeden externen Schlüsselspeicher und im `XksProxyConfiguration` Element der DescribeCustomKeyStoresAntwort angezeigt. Eine ausführliche Beschreibung der einzelnen Felder, einschließlich der Anforderungen an die Eindeutigkeit und Hilfe bei der Bestimmung des richtigen Werts für jedes Feld, finden Sie unter the section called “Erfüllen der Voraussetzungen” im Thema Erstellen eines externen Schlüsselspeichers.

Proxy-Konnektivität

Gibt an, ob der externe Schlüsselspeicher Konnektivität eines öffentlichen Endpunkts oder Konnektivität eines VPC-Endpunkt-Service verwendet.

Proxy-URI-Endpunkt

Der Endpunkt, der AWS KMS verwendet wird, um eine Verbindung zu Ihrem externen Schlüsselspeicher-Proxy herzustellen.

Proxy-URI-Pfad

Der Pfad vom Proxy-URI-Endpunkt, an den Proxy-API-Anfragen AWS KMS gesendet werden.

Proxy-Anmeldeinformation: Zugriffsschlüssel-ID

Teil der Anmeldeinformation für die Proxy-Authentifizierung, die Sie für den Proxy Ihres externen Schlüsselspeichers einrichten. Die Zugriffsschlüssel-ID identifiziert den geheimen Zugriffsschlüssel in der Anmeldeinformation.

AWS KMS verwendet den SigV4-Signaturprozess und die Anmeldeinformationen für die Proxyauthentifizierung, um seine Anfragen an Ihren externen Schlüsselspeicher-Proxy zu signieren. Die Anmeldeinformationen in der Signatur ermöglichen es dem externen Schlüsselspeicher-Proxy, Anfragen in Ihrem Namen von zu authentifizieren. AWS KMS

Name des VPC-Endpunkt-Service

Der Name des VPC-Endpunkt-Service von Amazon, der Ihren externen Schlüsselspeicher unterstützt. Dieser Wert wird nur angezeigt, wenn der externe Schlüsselspeicher die Konnektivität eines VPC-Endpunkt-Service nutzt. Sie können den Proxy Ihres externen Schlüsselspeichers in der VPC finden oder den VPC-Endpunkt-Service verwenden, um sicher mit dem Proxy Ihres externen Schlüsselspeichers zu kommunizieren.

Sehen Sie sich die Eigenschaften Ihres externen Schlüsselspeichers an

Sie können Ihren externen Schlüsselspeicher und die zugehörigen Eigenschaften in der AWS KMS Konsole oder mithilfe des DescribeCustomKeyStoresVorgangs anzeigen.

Mithilfe der AWS KMS Konsole

Gehen Sie wie folgt vor, um die externen Schlüsselspeicher in einem bestimmten Konto und einer bestimmten Region anzuzeigen:

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (Benutzerdefinierte Schlüsselspeicher), External key stores (Externe Schlüsselspeicher) aus.
4. Zum Anzeigen von detaillierten Informationen über einen externen Schlüsselspeicher wählen Sie den Namen des Schlüsselspeichers aus.

Verwenden der API AWS KMS

Verwenden Sie den DescribeCustomKeyStoresVorgang, um Ihre externen Schlüsselspeicher anzuzeigen. Standardmäßig gibt diese Operation alle benutzerdefinierten Schlüsselspeicher im Konto und in der Region zurück. Sie können jedoch entweder den Parameter `CustomKeyStoreId` oder `CustomKeyStoreName` (aber nicht beide) verwenden, um die Ausgabe auf einen bestimmten benutzerdefinierten Schlüsselspeicher zu begrenzen.

Bei benutzerdefinierten Schlüsselspeichern besteht die Ausgabe aus der ID, dem Namen und dem Typ des benutzerdefinierten Schlüsselspeichers sowie dem Verbindungsstatus des Schlüsselspeichers. Wenn der Verbindungsstatus `FAILED` ist, enthält die Ausgabe auch einen `ConnectionErrorCode`, der den Grund für den Fehler beschreibt. Hilfe bei der Interpretation des `ConnectionErrorCode` für einen externen Schlüsselspeicher finden Sie unter Verbindungsfehlercodes für externe Schlüsselspeicher.

Für externe Schlüsselspeicher enthält die Ausgabe auch das `XksProxyConfiguration`-Element. Dieses Element enthält den Konnektivitätstyp, den Proxy-URI-Endpunkt, den Proxy-URI-Pfad und die Zugriffsschlüssel-ID der Anmeldeinformation für die Proxy-Authentifizierung.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Beispielsweise gibt der folgende Befehl alle benutzerdefinierten Schlüsselspeicher im Konto und in der Region zurück. Sie können die Parameter `Marker` und `Limit` verwenden, um durch die benutzerdefinierten Schlüsselspeicher in der Ausgabe zu blättern.

```
$ aws kms describe-custom-key-stores
```

Der folgende Befehl verwendet den Parameter `CustomKeyStoreName`, um nur den externen Beispiel-Schlüsselspeicher mit dem Anzeigenamen `ExampleXksPublic` abzurufen. Dieser Beispiel-Schlüsselspeicher verwendet Konnektivität eines öffentlichen Endpunkts. Er ist mit dem Proxy seines externen Schlüsselspeichers verbunden.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksPublic
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleXksPublic",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-14T20:17:36.419000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE12345670EXAMPLE",
        "Connectivity": "PUBLIC_ENDPOINT",
        "UriEndpoint": "https://xks.example.com:6443",
        "UriPath": "/example/prefix/kms/xks/v1"
      }
    }
  ]
}
```

Mit dem folgenden Befehl wird ein Beispiel für einen externen Schlüsselspeicher mit Konnektivität eines VPC-Endpunkt-Service abgerufen. In diesem Beispiel ist der externe Schlüsselspeicher mit dem Proxy seines externen Schlüsselspeichers verbunden.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
```

```
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

Der `ConnectionState` `Disconnected` bedeutet, dass ein externer Schlüsselspeicher nie verbunden war oder dass er absichtlich vom Proxy seines externen Schlüsselspeichers getrennt wurde. Wenn jedoch der Versuch, einen KMS-Schlüssel in einem verbundenen externen Schlüsselspeicher zu verwenden, fehlschlägt, kann dies auf ein Problem mit dem Proxy des externen Schlüsselspeichers oder anderen externen Komponenten hinweisen.

Wenn der `ConnectionState` des externen Schlüsselspeichers `FAILED` lautet, enthält die `DescribeCustomKeyStores`-Antwort ein `ConnectionErrorCode`-Element, das den Grund für den Fehler angibt.

In der folgenden Ausgabe gibt der `XKS_PROXY_TIMED_OUT` Wert beispielsweise an, dass eine Verbindung zum externen Schlüsselspeicher-Proxy hergestellt werden AWS KMS kann, die Verbindung jedoch fehlgeschlagen ist, weil der externe Schlüsselspeicher-Proxy nicht innerhalb AWS KMS der vorgesehenen Zeit reagiert hat. Wenn Sie diesen Verbindungsfehlercode wiederholt sehen, benachrichtigen Sie den Proxy-Anbieter Ihres externen Schlüsselspeichers. Weitere Informationen zu diesem und anderen Verbindungsfehlern finden Sie unter Fehlerbehebung bei externen Schlüsselspeichern.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "FAILED",
      "ConnectionErrorCode": "XKS_PROXY_TIMED_OUT",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
```

```
    }
  ]
}
```

Überwachen Sie externe Schlüsselspeicher

AWS KMS sammelt Metriken für jede Interaktion mit einem externen Schlüsselspeicher und veröffentlicht sie in Ihrem CloudWatch Konto. Diese Metriken werden verwendet, um die Diagramme im Überwachungsabschnitt der Detailseite für jeden externen Schlüsselspeicher zu erstellen. Im folgenden Thema wird beschrieben, wie Sie die Diagramme verwenden, um Betriebs- und Konfigurationsprobleme zu identifizieren und zu beheben, die sich auf Ihren externen Schlüsselspeicher auswirken. Wir empfehlen, anhand der CloudWatch Metriken Alarme einzurichten, die Sie benachrichtigen, wenn Ihr externer Schlüsselspeicher nicht wie erwartet funktioniert. Weitere Informationen finden Sie unter Überwachung mit Amazon CloudWatch.

Themen

- Anzeigen der Diagramme
- Interpretieren der Diagramme

Anzeigen der Diagramme

Sie können die Diagramme auf verschiedenen Detailebenen anzeigen. Standardmäßig verwendet jedes Diagramm einen Zeitbereich von drei Stunden und einen Aggregationszeitraum von fünf Minuten. Sie können die Diagrammansicht innerhalb der Konsole anpassen, aber Ihre Änderungen werden auf die Standardeinstellungen zurückgesetzt, wenn die Detailseite des externen Schlüsselspeichers geschlossen oder der Browser aktualisiert wird. Hilfe zur CloudWatch Amazon-Terminologie finden Sie unter CloudWatch Amazon-Konzepte.

Anzeigen von Datenpunktdetails

Die Daten in jedem Diagramm werden von AWS KMS -Metriken gesammelt. Um weitere Informationen zu einem bestimmten Datenpunkt anzuzeigen, halten Sie die Maus auf den Datenpunkt im Liniendiagramm. Daraufhin wird ein Popup-Fenster mit weiteren Informationen zu der Metrik angezeigt, von der das Diagramm abgeleitet wurde. Jedes Listenelement zeigt den an diesem Datenpunkt aufgezeichneten Dimensionswert an. Das Pop-up zeigt einen Nullwert (–) an, wenn für den Dimensionswert an diesem Datenpunkt keine Metrikdaten verfügbar sind. Einige Diagramme zeichnen mehrere Dimensionen und Werte für einen einzelnen Datenpunkt auf. Andere Diagramme,

wie das Zuverlässigkeitsdiagramm, verwenden die von der Metrik erfassten Daten, um einen eindeutigen Wert zu berechnen. Jedes Listenelement ist einer anderen Farbe des Liniendiagramms zugeordnet.

Ändern des Zeitbereichs

Zum Ändern des Zeitbereichs wählen Sie im Überwachungsbereich oben rechts einen der vordefinierten Zeitbereiche aus. Die vordefinierten Zeitbereiche reichen von 1 Stunde bis 1 Woche (1h, 3h, 12h, 1d, 3d oder 1w). Dadurch wird der Zeitbereich für alle Diagramme angepasst. Wenn Sie ein bestimmtes Diagramm in einem anderen Zeitraum anzeigen oder einen benutzerdefinierten Zeitraum festlegen möchten, vergrößern Sie das Diagramm oder zeigen Sie es in der CloudWatch Amazon-Konsole an.

Vergrößern von Diagrammen

Mit der Mini-Map-Zoomfunktion können Sie sich auf Abschnitte von Liniendiagrammen und gestapelte Teile der Diagramme konzentrieren, ohne zwischen vergrößerter und verkleinerter Ansicht zu wechseln. Sie können die Mini-Map-Zoomfunktion beispielsweise verwenden, um sich auf einen Spitzenwert in einem Diagramm zu konzentrieren, sodass Sie den Spitzenwert mit anderen Diagrammen im Überwachungsabschnitt auf derselben Zeitachse vergleichen können.

1. Wählen Sie den Bereich des Diagramms aus, auf den Sie sich konzentrieren möchten, und lassen Sie die Maustaste los.
2. Um den Zoom zurückzusetzen, wählen Sie das Symbol Zoom zurücksetzen aus, das wie eine Lupe mit einem Minuszeichen (-) darin aussieht.

Vergrößern eines Diagramms

Wenn Sie ein Diagramm vergrößern möchten, wählen Sie darin das Menüsymbol oben rechts aus und wählen Sie dann Enlarge (Vergrößern). Sie können auch das Vergrößerungssymbol auswählen, das neben dem Menüsymbol angezeigt wird, wenn Sie den Mauszeiger über ein Diagramm bewegen.

Durch die Vergrößerung eines Diagramms können Sie die Ansicht eines Diagramms weiter verändern, indem Sie einen anderen Zeitraum, einen benutzerdefinierten Zeitbereich oder ein Aktualisierungsintervall angeben. Diese Änderungen werden auf die Standardeinstellungen zurückgesetzt, wenn Sie die vergrößerte Ansicht schließen.

Ändern des Zeitraums

1. Wählen Sie das Optionsmenü für den Zeitraum aus. In diesem Menü wird standardmäßig folgender Wert angezeigt: 5 Minuten.
2. Wählen Sie einen Zeitraum. Die vordefinierten Zeiträume reichen von 1 Sekunde bis 30 Tage.

   Sie können beispielsweise eine Ein-Minuten-Ansicht auswählen. Dies kann nützlich sein, wenn Sie Fehler beheben. Sie können auch eine weniger detaillierte Ein-Stunden-Ansicht auswählen. Dies kann nützlich sein, wenn Sie einen größeren Zeitraum anzeigen (z. B. 3 Tage), damit Sie Trends über die Zeit anzeigen können. Weitere Informationen finden Sie unter Perioden im CloudWatch Amazon-Benutzerhandbuch.

Ändern des Zeitraums oder Zeitzone

1. Wählen Sie einen der vordefinierten Zeitbereiche, die von 1 Stunde bis 1 Woche reichen (1h, 3h, 12h, 1d, 3d oder 1w). Alternativ können Sie Custom (Benutzerdefiniert) auswählen, um Ihren eigenen Zeitraum festzulegen.
2. Wählen Sie Custom (Benutzerdefiniert) aus.
   a. Zeitraum: Wählen Sie die Registerkarte Absolute (Absolut) im Feld oben links aus. Verwenden Sie die Kalenderauswahl oder die Textfelder, um einen Zeitraum festzulegen.
   b. Zeitzone: Wählen Sie das Dropdown-Menü im Feld oben rechts aus. Sie können die Zeitzone auf UTC oder Local time zone (lokale Zeitzone) ändern.
3. Wählen Sie nach dem Festlegen eines Zeitraums Apply (Anwenden) aus.

Ändern der Aktualisierungshäufigkeit der Daten in Ihrem Diagramm

1. Wählen Sie oben rechts das Menü Refresh options (Aktualisierungsoptionen) aus.
2. Wählen Sie ein Aktualisierungsintervall (Aus, 10 Sekunden, 1 Minute, 2 Minuten, 5 Minuten oder 15 Minuten).

Grafiken in der CloudWatch Amazon-Konsole anzeigen

Die Grafiken im Monitoring-Bereich werden aus vordefinierten Kennzahlen abgeleitet, die auf Amazon AWS KMS veröffentlicht CloudWatch werden. Sie können sie in der CloudWatch Konsole öffnen und in CloudWatch Dashboards speichern. Wenn Sie über mehrere externe Schlüsselspeicher verfügen, können Sie die jeweiligen Diagramme öffnen CloudWatch und sie in einem einzigen Dashboard speichern, um deren Zustand und Nutzung zu vergleichen.

Zum CloudWatch Dashboard hinzufügen

Wählen Sie in der oberen rechten Ecke Zum Dashboard hinzufügen aus, um alle Grafiken zu einem CloudWatch Amazon-Dashboard hinzuzufügen. Sie können entweder ein vorhandenes Dashboard auswählen oder ein neues erstellen. Informationen zur Verwendung dieses Dashboards zur Erstellung benutzerdefinierter Ansichten der Grafiken und Alarme finden Sie unter Verwenden von CloudWatch Amazon-Dashboards im CloudWatch Amazon-Benutzerhandbuch.

In Metriken CloudWatch anzeigen

Wählen Sie das Menüsymbol in der oberen rechten Ecke eines einzelnen Diagramms aus und wählen Sie In Kennzahlen anzeigen, um dieses Diagramm in der CloudWatch Amazon-Konsole anzuzeigen. Von der CloudWatch Konsole aus können Sie dieses einzelne Diagramm zu einem Dashboard hinzufügen und Zeitbereiche, Zeiträume und Aktualisierungsintervalle ändern. Weitere Informationen finden Sie unter Metriken grafisch darstellen im CloudWatch Amazon-Benutzerhandbuch.

Interpretieren der Diagramme

AWS KMS bietet mehrere Grafiken zur Überwachung des Zustands Ihres externen Schlüsselspeichers in der AWS KMS Konsole. Diese Diagramme werden automatisch konfiguriert und von AWS KMS -Metriken abgeleitet.

Die Diagrammdaten werden als Teil der Aufrufe gesammelt, die Sie an Ihren externen Schlüsselspeicher und Ihre externen Schlüssel richten. Möglicherweise werden während eines Zeitraums, in dem Sie keine Anrufe getätigt haben, Daten in den Diagrammen angezeigt. Diese Daten stammen aus den regelmäßigen `GetHealthStatus` Aufrufen, die in Ihrem Namen getätigt werden, AWS KMS um den Status Ihres externen Schlüsselspeicher-Proxys und Ihres externen Schlüsselmanagers zu überprüfen. Wenn Ihre Diagramme die Meldung No data available (Keine Daten verfügbar) anzeigen, wurden in diesem Zeitraum keine Anrufe aufgezeichnet oder Ihr externer Schlüsselspeicher befindet sich im Status `DISCONNECTED`. Möglicherweise können Sie den Zeitpunkt der Trennung Ihres externen Schlüsselspeichers ermitteln, indem Sie Ihre Ansicht auf einen breiteren Zeitbereich einstellen.

Themen

- Anfragen insgesamt
- Zuverlässigkeit
- Latency
- 5 häufigste Ausnahmen

- Gültigkeitsdauer des Zertifikats in Tagen

Anfragen insgesamt

Die Gesamtzahl der AWS KMS Anfragen, die in einem bestimmten Zeitraum für einen bestimmten externen Schlüsselspeicher eingegangen sind. Anhand dieses Diagramms können Sie feststellen, ob das Risiko einer Drosselung besteht.

AWS KMS empfiehlt, dass Ihr externer Schlüsselmanager bis zu 1800 Anfragen für kryptografische Operationen pro Sekunde verarbeiten kann. Wenn Sie innerhalb von fünf Minuten 540 000 Aufrufe erreichen, besteht das Risiko einer Drosselung.

Sie können die Anzahl der Anfragen für kryptografische Operationen mit KMS-Schlüsseln in Ihrem externen Schlüsselspeicher überwachen, der mit der Metrik AWS KMS gedrosselt wird. ExternalKeyStoreThrottle

Wenn Sie sehr häufig `KMSInvalidStateException`-Fehler mit einer Meldung erhalten, in der erklärt wird, dass die Anforderung „aufgrund einer sehr hohen Anforderungsrate“ abgelehnt wurde, könnte dies darauf hinweisen, dass Ihr externer Schlüsselmanager oder externer Schlüsselspeicher-Proxy mit der aktuellen Anforderungsrate nicht Schritt halten kann. Verringen Sie wenn möglich Ihre Anforderungsrate. Sie können auch eine Verringerung des Anforderungskontingentwerts für benutzerdefinierte Schlüsselspeicher anfordern. Eine Verringerung dieses Kontingentwerts kann die Drosselung erhöhen, deutet aber darauf hin, dass AWS KMS überschüssige Anfragen schnell zurückgewiesen werden, bevor sie an Ihren externen Schlüsselspeicher-Proxy oder externen Schlüsselmanager gesendet werden. Um eine Reduzierung des Kontingents zu beantragen, besuchen Sie bitte das AWS -Support Center und erstellen Sie einen Fall.

Das Diagramm für die gesamten Anforderungen wird von der XksProxyErrors-Metrik abgeleitet, die Daten über die erfolgreichen und nicht erfolgreichen Antworten erfasst, die AWS KMS von Ihrem externen Schlüsselspeicher-Proxy erhält. Wenn Sie sich einen bestimmten Datenpunkt ansehen, zeigt das Pop-up den Wert der `CustomKeyStoreId` Dimension zusammen mit der Gesamtzahl der an diesem Datenpunkt aufgezeichneten AWS KMS Anfragen an. Die `CustomKeyStoreId` bleibt immer gleich.

Zuverlässigkeit

Der Prozentsatz der AWS KMS Anfragen, bei denen der externe Schlüsselspeicher-Proxy entweder eine erfolgreiche Antwort oder einen Fehler, der nicht erneut versucht werden konnte, zurückgegeben

hat. Bewerten Sie anhand dieses Diagramms den Betriebszustand Ihres externen Schlüsselspeicher-Proxys.

Wenn das Diagramm einen Wert von weniger als 100 % anzeigt, weist dies auf Fälle hin, in denen der Proxy entweder nicht oder mit einem wiederholbaren Fehler geantwortet hat. Dies kann auf Probleme mit dem Netzwerk, Langsamkeit des externen Schlüsselspeicher-Proxys oder des externen Schlüsselmanagers oder auf Implementierungsfehler hindeuten.

Wenn die Anforderung eine falsche Anmeldeinformation enthält und Ihr Proxy mit einer `AuthenticationFailedException` antwortet, zeigt das Diagramm trotzdem 100 % Zuverlässigkeit an, weil der Proxy einen falschen Wert in der API-Anforderung des externen Schlüsselspeicher-Proxys identifiziert hat und der Fehler daher erwartet wurde. Wenn der Prozentsatz Ihres Zuverlässigkeitsdiagramms 100 % beträgt, reagiert Ihr externer Schlüsselspeicher-Proxy wie erwartet. Wenn das Diagramm einen Wert von weniger als 100 % anzeigt, hat der Proxy entweder mit einem wiederholbaren Fehler geantwortet oder eine Zeitüberschreitung verursacht. Wenn der Proxy beispielsweise aufgrund einer sehr hohen Anzahl von Anforderungen mit einer `ThrottlingException` antwortet, wird ein niedrigerer Zuverlässigkeitsgrad angezeigt, da der Proxy nicht in der Lage war, ein spezifisches Problem in der Anforderung zu identifizieren, das zum Fehlschlagen führte. Dies liegt daran, dass es sich bei wiederholbaren Fehlern wahrscheinlich um vorübergehende Probleme handelt, die durch einen erneuten Versuch der Anforderung behoben werden können.

Die folgenden Fehlerantworten senken den Zuverlässigkeitsgrad. Mit dem Diagramm 5 häufigste Ausnahmen und der Metrik XksProxyErrors können Sie weiter überwachen, wie häufig Ihr Proxy jeden wiederholbaren Fehler zurückgibt.

- `InternalException`
- `DependencyTimeoutException`
- `ThrottlingException`
- `XksProxyUnreachableException`

Das Zuverlässigkeitsdiagramm wird aus der XksProxyErrors Metrik abgeleitet, die Daten sowohl zu den erfolgreichen als auch zu den erfolglosen Antworten sammelt, die Sie von Ihrem externen Schlüsselspeicher-Proxy AWS KMS erhalten. Der Zuverlässigkeitsprozentsatz sinkt nur, wenn die Antwort den `ErrorType`-Wert `Retryable` hat. Wenn Sie sich einen bestimmten Datenpunkt ansehen, zeigt das Pop-up den Wert der `CustomKeyStoreId` Dimension zusammen mit dem

Zuverlässigkeitsprozentsatz für AWS KMS Anfragen an, die an diesem Datenpunkt aufgezeichnet wurden. Die `CustomKeyStoreId` bleibt immer gleich.

Wir empfehlen, die XksProxyErrors Metrik zu verwenden, um einen CloudWatch Alarm zu erstellen, der Sie über potenzielle Netzwerkprobleme informiert, indem er Sie benachrichtigt, wenn innerhalb einer Minute mehr als fünf wiederholbare Fehler aufgezeichnet werden. Weitere Informationen finden Sie unter Erstellen Sie einen Alarm für Fehler, die wiederholt werden können.

Latency

Die Anzahl der Millisekunden, die ein externer Schlüsselspeicher-Proxy benötigt, um auf eine Anfrage zu antworten. AWS KMS Verwenden Sie dieses Diagramm, um die Leistung Ihres externen Schlüsselspeicher-Proxys und externen Schlüsselmanagers zu bewerten.

AWS KMS erwartet, dass der externe Schlüsselspeicher-Proxy auf jede Anfrage innerhalb von 250 Millisekunden reagiert. Im Falle von Netzwerk-Timeouts AWS KMS wird die Anfrage einmal wiederholt. Wenn der Proxy ein zweites Mal fehlschlägt, entspricht die aufgezeichnete Latenz der kombinierten Timeout-Grenze für beide Anforderungen und das Diagramm zeigt etwa 500 Millisekunden an. In allen anderen Fällen, in denen der Proxy nicht innerhalb der Timeout-Grenze von 250 Millisekunden antwortet, beträgt die aufgezeichnete Latenzzeit 250 Millisekunden. Wenn der Proxy bei Verschlüsselungs- und Entschlüsselungsvorgängen häufig eine Zeitüberschreitung aufweist, wenden Sie sich an Ihren externen Proxy-Administrator. Hilfe zur Behebung von Latenzproblemen finden Sie unter Latenz- und Zeitüberschreitungsfehler.

Langsame Antworten können auch darauf hindeuten, dass Ihr externer Schlüsselmanager den aktuellen Anforderungsverkehr nicht bewältigen kann. AWS KMS empfiehlt, dass Ihr externer Schlüsselmanager bis zu 1800 Anfragen für kryptografische Operationen pro Sekunde verarbeiten kann. Wenn Ihr externer Schlüsselmanager die Rate von 1 800 Anforderungen pro Sekunde nicht bewältigen kann, sollten Sie eine Verringerung Ihrer Anforderungsquote für KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher anfordern. Anforderungen für kryptografische Vorgänge, die die KMS-Schlüssel in Ihrem externen Schlüsselspeicher verwenden, schlagen schnell mit einer Drosselungsausnahme fehl, anstatt verarbeitet und später von Ihrem externen Schlüsselspeicher-Proxy oder externen Schlüsselmanager abgelehnt zu werden.

Das Latenzdiagramm wird von der XksProxyLatency-Metrik abgeleitet. Wenn Sie einen bestimmten Datenpunkt anzeigen, zeigt das Popup-Fenster die entsprechenden Werte der Dimensionen `KmsOperation` und `XksOperation` sowie die durchschnittliche Latenz an, die für die Vorgänge an diesem Datenpunkt aufgezeichnet wurde. Die Listenelemente sind von der höchsten zur niedrigsten Latenz sortiert.

Wir empfehlen, die XksProxyLatency Metrik zu verwenden, um einen CloudWatch Alarm auszulösen, der Sie benachrichtigt, wenn sich Ihre Latenz dem Timeout-Limit nähert. Weitere Informationen finden Sie unter Erstellen Sie einen Alarm für das Antwort-Timeout.

5 häufigste Ausnahmen

Die fünf häufigsten Ausnahmen für fehlgeschlagene kryptografische und verwaltungstechnische Vorgänge innerhalb eines bestimmten Zeitraums. Verwenden Sie dieses Diagramm, um die häufigsten Fehler zu verfolgen, sodass Sie Ihre Entwicklungsarbeit nach Prioritäten ordnen können.

Diese Anzahl umfasst Ausnahmen, die vom externen Schlüsselspeicher-Proxy AWS KMS empfangen wurden, und Ausnahmen`XksProxyUnreachableException`, die intern AWS KMS zurückgegeben werden, wenn keine Kommunikation mit dem externen Schlüsselspeicher-Proxy hergestellt werden kann.

Hohe Raten wiederholbarer Fehler können auf Netzwerkfehler hinweisen, während hohe Raten nicht wiederholbarer Fehler auf ein Problem mit der Konfiguration Ihres externen Schlüsselspeichers hinweisen können. Ein Anstieg `AuthenticationFailedExceptions` deutet beispielsweise auf eine Diskrepanz zwischen den im konfigurierten Authentifizierungsdaten AWS KMS und dem externen Schlüsselspeicher-Proxy hin. Informationen zur Konfiguration Ihres externen Schlüsselspeichers finden Sie unter Externe Schlüsselspeicher anzeigen. Informationen zum Bearbeiten der Einstellungen Ihres externen Schlüsselspeichers finden Sie unter Eigenschaften des externen Schlüsselspeichers bearbeiten.

Die Ausnahmen, die vom externen Schlüsselspeicher-Proxy AWS KMS empfangen werden, unterscheiden sich von den Ausnahmen, die AWS KMS zurückgegeben werden, wenn ein Vorgang fehlschlägt. AWS KMS Bei kryptografischen Vorgängen wird ein `KMSInvalidStateException` für alle Fehler zurückgegeben, die mit der externen Konfiguration oder dem Verbindungsstatus des externen Schlüsselspeichers zusammenhängen. Identifizieren Sie das Problem anhand des zugehörigen Fehlermeldungstexts.

In der folgenden Tabelle sind die Ausnahmen aufgeführt, die im Diagramm mit den fünf häufigsten Ausnahmen angezeigt werden können, sowie die entsprechenden Ausnahmen, die AWS KMS Ihnen angezeigt werden.

Fehlertyp	Im Diagramm angezeigte Ausnahme	Ausnahme, die zu Ihnen AWS KMS zurückgekehrt ist
Nicht wiederholbar	**`AccessDeniedException`** Hilfe zur Problembehebung finden Sie unter Probleme mit der Proxy-Autorisierung.	**`CustomKeyStoreInvalidStateException`** als Antwort auf `CreateKey` -Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Nicht wiederholbar	**`AuthenticationFailedException`** Hilfe zur Problembehebung finden Sie unter Fehler mit der Anmeldeinformation für die Authentifizierung.	**`XksProxyIncorrectAuthenticationCredentialException`** als Antwort auf `CreateCustomKeyStore` - und `UpdateCustomKeyStore` -Vorgänge. **`CustomKeyStoreInvalidStateException`** als Antwort auf `CreateKey` -Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Wiederholbar	**`DependencyTimeoutException`** Hilfe zur Problembehebung finden Sie unter Latenz- und Zeitüberschreitungsfehler.	**`XksProxyUriUnreachableException`** als Antwort auf `CreateCustomKeyStore` - und `UpdateCustomKeyStore` -Vorgänge. **`CustomKeyStoreInvalidStateException`** als

Fehlertyp	Im Diagramm angezeigte Ausnahme	Ausnahme, die zu Ihnen AWS KMS zurückgekehrt ist
		Antwort auf `CreateKey` - Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Wiederholbar	**`InternalException`** Der Proxy des externen Schlüsselspeichers hat die Anforderung abgelehnt, weil er nicht mit dem externen Schlüsselmanager kommunizieren kann. Stellen Sie sicher, dass die Proxykonfiguration für den externen Schlüsselspeicher korrekt ist und dass der externe Schlüsselmanager verfügbar ist.	**`XksProxyInvalidResponseException`** als Antwort auf `CreateCustomKeyStore` - und `UpdateCustomKeyStore` - Vorgänge. **`CustomKeyStoreInvalidStateException`** als Antwort auf `CreateKey` - Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Nicht wiederholbar	**`InvalidCiphertextException`** Hilfe zur Problembehebung finden Sie unter Entschlüsselungsfehler.	**`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.

Fehlertyp	Im Diagramm angezeigte Ausnahme	Ausnahme, die zu Ihnen AWS KMS zurückgekehrt ist
Nicht wiederholbar	**`InvalidKeyUsageException`** Hilfe zur Problembehebung finden Sie unter Kryptografische Operationsfehler für den externen Schlüssel.	**`XksKeyInvalidConfigurationException`** als Antwort auf `CreateKey` -Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Nicht wiederholbar	**`InvalidStateException`** Hilfe zur Problembehebung finden Sie unter Kryptografische Operationsfehler für den externen Schlüssel.	**`XksKeyInvalidConfigurationException`** als Antwort auf `CreateKey` -Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Nicht wiederholbar	**`InvalidUriPathException`** Hilfe zur Problembehebung finden Sie unter Allgemeine Konfigurationsfehler.	**`XksProxyInvalidConfigurationException`** als Antwort auf `CreateCustomKeyStore` - und `UpdateCustomKeyStore` -Vorgänge. **`CustomKeyStoreInvalidStateException`** als Antwort auf `CreateKey` -Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.

Fehlertyp	Im Diagramm angezeigte Ausnahme	Ausnahme, die zu Ihnen AWS KMS zurückgekehrt ist
Nicht wiederholbar	**`KeyNotFoundException`** Hilfe zur Problembehebung finden Sie unter Fehler mit externen Schlüsseln.	**`XksKeyNotFoundException`** als Antwort auf `CreateKey` -Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Wiederholbar	**`ThrottlingException`** Der Proxy des externen Schlüsselspeichers hat die Anforderung aufgrund einer sehr hohen Anzahl von Anforderungen abgelehnt. Verringern Sie die Häufigkeit Ihrer Aufrufe mit KMS-Schlüsseln in diesem externen Schlüsselspeicher.	**`XksProxyUriUnreachableException`** als Antwort auf `CreateCustomKeyStore` - und `UpdateCustomKeyStore` -Vorgänge. **`CustomKeyStoreInvalidStateException`** als Antwort auf `CreateKey` -Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Nicht wiederholbar	**`UnsupportedOperationException`** Hilfe zur Problembehebung finden Sie unter Kryptografische Operationsfehler für den externen Schlüssel.	**`XksKeyInvalidResponseException`** als Antwort auf `CreateKey` -Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.

Fehlertyp	Im Diagramm angezeigte Ausnahme	Ausnahme, die zu Ihnen AWS KMS zurückgekehrt ist
Nicht wiederholbar	**`ValidationException`** Hilfe zur Problembehebung finden Sie unter Proxy-Probleme.	**`XksProxyInvalidResponseException`** als Antwort auf `CreateCustomKeyStore` - und `UpdateCustomKeyStore` - Vorgänge. **`CustomKeyStoreInvalidStateException`** als Antwort auf `CreateKey` - Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.
Wiederholbar	**`XksProxyUnreachableException`** Wenn dieser Fehler wiederholt auftritt, überprüfen Sie, ob Ihr externer Schlüsselspeicher-Proxy aktiv und mit dem Netzwerk verbunden ist und ob sein URI-Pfad und Endpunkt-URI oder der Name des VPC-Service in Ihrem externen Schlüsselspeicher korrekt sind.	**`XksProxyUriUnreachableException`** als Antwort auf `CreateCustomKeyStore` - und `UpdateCustomKeyStore` - Vorgänge. **`CustomKeyStoreInvalidStateException`** als Antwort auf `CreateKey` - Vorgänge. **`KMSInvalidStateException`** als Antwort auf kryptografische Vorgänge.

Das Diagramm mit den fünf wichtigsten Ausnahmen wird von der XksProxyErrors-Metrik abgeleitet. Wenn Sie einen bestimmten Datenpunkt anzeigen, zeigt das Popup-Fenster den Wert der Dimension

`ExceptionName` zusammen mit der Anzahl, wie oft die Ausnahme an diesem Datenpunkt aufgezeichnet wurde. Die fünf Elemente der Liste sind in der Reihenfolge der häufigsten Ausnahme bis zur geringsten geordnet.

Wir empfehlen, mithilfe der XksProxyErrors Metrik einen CloudWatch Alarm zu erstellen, der Sie über potenzielle Konfigurationsprobleme informiert, indem er Sie benachrichtigt, wenn innerhalb einer Minute mehr als fünf Fehler registriert werden, die nicht erneut versucht werden können. Weitere Informationen finden Sie unter Erstellen Sie einen Alarm für Fehler, die nicht wiederholt werden können.

Gültigkeitsdauer des Zertifikats in Tagen

Die Anzahl der Tage, bis das TLS-Zertifikat für den Proxy-Endpunkt Ihres externen Schlüsselspeichers (`XksProxyUriEndpoint`) abläuft. Verwenden Sie dieses Diagramm zur Überwachung des bevorstehenden Ablaufs Ihres TLS-Zertifikats.

Wenn das Zertifikat abläuft, AWS KMS kann nicht mit dem externen Schlüsselspeicher-Proxy kommuniziert werden. Der Zugriff auf alle durch KMS-Schlüssel geschützten Daten in Ihrem externen Schlüsselspeicher ist dann erst wieder möglich, wenn Sie das Zertifikat erneuern.

Das Diagramm mit der Gültigkeitsdauer des Zertifikats in Tagen wird von der XksProxyCertificateDaysToExpire-Metrik abgeleitet. Wir empfehlen dringend, diese Metrik zu verwenden, um einen CloudWatch Alarm auszulösen, der Sie über den bevorstehenden Ablauf informiert. Der Ablauf des Zertifikats kann den Zugriff auf Ihre verschlüsselten Ressourcen verhindern. Stellen Sie den Alarm so ein, dass Ihre Organisation Zeit hat, das Zertifikat zu erneuern, bevor es abläuft. Weitere Informationen finden Sie unter Erstellen Sie einen Alarm für den Ablauf des Zertifikats.

Externe Schlüsselspeicher Connect und trennen

Neue externe Schlüsselspeicher sind nicht verbunden. Um Ihren externen Schlüsselspeicher zu erstellen und AWS KMS keys in Ihrem externen Schlüsselspeicher zu verwenden, müssen Sie Ihren externen Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy verbinden. Sie können Ihren externen Schlüsselspeicher jederzeit verbinden und trennen und seinen Verbindungsstatus anzeigen.

Solange Ihr externer Schlüsselspeicher nicht verbunden ist, AWS KMS kann er nicht mit Ihrem externen Schlüsselspeicher-Proxy kommunizieren. Daher können Sie Ihren externen Schlüsselspeicher und seine vorhandenen KMS-Schlüssel anzeigen und verwalten. Sie können

jedoch keine KMS-Schlüssel in Ihrem externen Schlüsselspeicher erstellen oder dessen KMS-Schlüssel in kryptografischen Vorgängen verwenden. Es kann sein, dass Sie die Verbindung Ihres externen Schlüsselspeichers irgendwann trennen müssen, z. B. wenn Sie seine Eigenschaften bearbeiten. Wenn Sie die Verbindung zum Schlüsselspeicher trennen, kann der Betrieb von AWS Diensten, die dessen KMS-Schlüssel verwenden, gestört werden.

Sie müssen Ihren externen Schlüsselspeicher nicht verbinden. Sie können die Verbindung eines externen Schlüsselspeichers auf unbestimmte Zeit getrennt lassen und die Verbindung nur herstellen, wenn Sie den Schlüsselspeicher verwenden müssen. Möglicherweise möchten Sie die Verbindung jedoch von Zeit zu Zeit testen, um zu prüfen, ob die Einstellungen korrekt sind und eine Verbindungsherstellung möglich ist.

Wenn Sie die Verbindung zu einem benutzerdefinierten Schlüsselspeicher trennen, werden die KMS-Schlüssel im Schlüsselspeicher sofort unbrauchbar (je nach letztendlicher Konsistenz). Ressourcen, die mit durch den KMS-Schlüssel geschützten Datenschlüsseln verschlüsselt wurden, sind jedoch nicht betroffen, bis der KMS-Schlüssel erneut verwendet wird, z. B. zur Entschlüsselung des Datenschlüssels. Dieses Problem betrifft AWS-Services, von denen viele Datenschlüssel verwenden, um Ihre Ressourcen zu schützen. Details hierzu finden Sie unter Auswirkung von unbrauchbaren KMS-Schlüsseln auf Datenschlüssel.

> **Note**
>
> Externe Schlüsselspeicher haben nur dann den Status `DISCONNECTED`, wenn der Schlüsselspeicher noch nie verbunden wurde oder Sie ihn explizit trennen. Der Status `CONNECTED` bedeutet nicht, dass der externe Schlüsselspeicher oder seine unterstützenden Komponenten effizient funktionieren. Informationen über die Leistung der Komponenten Ihres externen Schlüsselspeichers finden Sie in den Diagrammen im Abschnitt Monitoring (Überwachung) auf der Detailseite für jeden externen Schlüsselspeicher. Details hierzu finden Sie unter Überwachen Sie externe Schlüsselspeicher.
> Ihr externer Schlüsselmanager bietet möglicherweise zusätzliche Methoden zum Beenden und Neustarten der Kommunikation zwischen Ihrem AWS KMS externen Schlüsselspeicher und Ihrem externen Schlüsselspeicher-Proxy oder zwischen Ihrem externen Schlüsselspeicher-Proxy und dem externen Schlüsselmanager. Weitere Informationen finden Sie in der Dokumentation zum externen Schlüsselmanager.

Themen

- Verbindungsstatus

- Einen externen Schlüsselspeicher verbinden
- Trennen Sie die Verbindung zu einem externen Schlüsselspeicher

Verbindungsstatus

Beim Verbinden und Trennen der Verbindung wird der Verbindungsstatus Ihres benutzerdefinierten Schlüsselspeichers geändert. Die Werte für den Verbindungsstatus sind für AWS CloudHSM Schlüsselspeicher und externe Schlüsselspeicher identisch.

Um den Verbindungsstatus Ihres benutzerdefinierten Schlüsselspeichers anzuzeigen, verwenden Sie das DescribeCustomKeyStoresBetriebssystem oder die AWS KMS Konsole. Der Verbindungsstatus wird in jeder Tabelle des benutzerdefinierten Schlüsselspeichers, im Abschnitt General configuration (Allgemeine Konfiguration) der Detailseite jedes benutzerdefinierten Schlüsselspeichers und auf der Registerkarte Cryptographic configuration (Kryptografische Konfiguration) der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher angezeigt. Details dazu finden Sie unter Sehen Sie sich einen AWS CloudHSM Schlüsselspeicher an und Externe Schlüsselspeicher anzeigen.

Ein benutzerdefinierter Schlüsselspeicher kann einen der folgenden Verbindungsstatus haben:

- `CONNECTED`: Der benutzerdefinierte Schlüsselspeicher ist mit seinem Unterstützungsschlüsselspeicher verbunden. Sie können KMS-Schlüssel im benutzerdefinierten Schlüsselspeicher erstellen und verwenden.

  Der Backing-Schlüsselspeicher für einen AWS CloudHSM Schlüsselspeicher ist der zugehörige AWS CloudHSM Cluster. Der Unterstützungsschlüsselspeicher für einen externen Schlüsselspeicher ist der externe Schlüsselspeicher-Proxy und der externe Schlüsselmanager, den er unterstützt.

  Der Status „CONNECTED“ (VERBUNDEN) bedeutet, dass eine Verbindung erfolgreich war und der benutzerdefinierte Schlüsselspeicher nicht absichtlich getrennt wurde. Er bedeutet nicht, dass die Verbindung ordnungsgemäß funktioniert. Informationen zum Status des AWS CloudHSM Clusters, der Ihrem AWS CloudHSM Schlüsselspeicher zugeordnet ist, finden Sie AWS CloudHSM im AWS CloudHSM Benutzerhandbuch unter CloudWatch Metriken abrufen für. Informationen zum Status und Betrieb Ihres externen Schlüsselspeichers finden Sie in den Diagrammen im Abschnitt Überwachung auf der Detailseite für jeden externen Schlüsselspeicher. Details hierzu finden Sie unter Überwachen Sie externe Schlüsselspeicher.
- `CONNECTING`: Der Prozess der Verbindung eines benutzerdefinierten Schlüsselspeichers ist im Gange. Dies ist ein vorübergehender Zustand.

- `DISCONNECTED`: Der benutzerdefinierte Schlüsselspeicher wurde noch nie mit seinem Backing-Speicher verbunden, oder er wurde absichtlich mithilfe der AWS KMS Konsole oder des DisconnectCustomKeyStoreVorgangs getrennt.
- `DISCONNECTING`: Der Prozess der Trennung eines benutzerdefinierten Schlüsselspeichers ist im Gange. Dies ist ein vorübergehender Zustand.
- `FAILED`: Ein Versuch, den benutzerdefinierten Schlüsselspeicher zu verbinden, ist fehlgeschlagen. Das `ConnectionErrorCode` in der DescribeCustomKeyStoresAntwort weist auf das Problem hin.

Um einen benutzerdefinierten Schlüsselspeicher zu verbinden, muss sein Verbindungsstatus `DISCONNECTED` sein. Wenn der Verbindungsstatus `FAILED` lautet, identifizieren und lösen Sie das Problem anhand des `ConnectionErrorCode`. Trennen Sie dann den benutzerdefinierten Schlüsselspeicher, bevor Sie versuchen, die Verbindung wieder herzustellen. Hilfestellung bei fehlgeschlagenen Verbindungen finden Sie unter Fehler bei der Verbindung mit dem externen Schlüsselspeicher. Hilfe beim Beantworten eines Verbindungsfehlercodes finden Sie unter Verbindungsfehlercodes für externe Schlüsselspeicher.

Anzeigen des Verbindungsfehlercodes:

- Sehen Sie sich in der DescribeCustomKeyStoresAntwort den Wert des `ConnectionErrorCode` Elements an. Die `DescribeCustomKeyStores`-Antwort enthält dieses Element nur, wenn der `ConnectionState` `FAILED` ist.
- Um den Verbindungsfehlercode in der AWS KMS Konsole auf der Detailseite für den externen Schlüsselspeicher anzuzeigen, zeigen Sie mit der Maus auf den Wert Fehlgeschlagen.

Einen externen Schlüsselspeicher verbinden

Wenn Ihr externer Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy verbunden ist, können Sie KMS-Schlüssel in Ihrem externen Schlüsselspeicher erstellen und seine vorhandenen KMS-Schlüssel in kryptografischen Vorgängen verwenden.

Der Prozess, der einen externen Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy verbindet, unterscheidet sich je nach der Konnektivität des externen Schlüsselspeichers.

- Wenn Sie einen externen Schlüsselspeicher mit öffentlicher Endpunktkonnektivität verbinden, AWS KMS sendet er eine GetHealthStatus Anfrage an den externen Schlüsselspeicher-Proxy, um den Proxy-URI-Endpunkt, den Proxy-URI-Pfad und die Anmeldeinformationen für die

Proxyauthentifizierung zu überprüfen. Eine erfolgreiche Antwort vom Proxy bestätigt, dass der Proxy-URI-Endpunkt und der Proxy-URI-Pfad korrekt und zugänglich sind und dass der Proxy die mit der Proxy-Authentifizierungsanmeldeinformation für den externen Schlüsselspeicher signierte Anforderung authentifiziert hat.

- Wenn Sie einen externen Schlüsselspeicher mit VPC-Endpunktdienst-Konnektivität mit seinem externen Schlüsselspeicher-Proxy verbinden, AWS KMS geht Folgendes vor:
  - Bestätigt, dass die Domain für den im Proxy-URI-Endpunkt angegebenen privaten DNS-Namen verifiziert ist.
  - Erstellt einen Schnittstellenendpunkt von einer AWS KMS VPC zu Ihrem VPC-Endpunktdienst.
  - Erstellt eine private gehostete Zone für den im Proxy-URI-Endpunkt angegebenen privaten DNS-Namen.
  - Sendet eine GetHealthStatusAnfrage an den externen Schlüsselspeicher-Proxy.
    Eine erfolgreiche Antwort vom Proxy bestätigt, dass der Proxy-URI-Endpunkt und der Proxy-URI-Pfad korrekt und zugänglich sind und dass der Proxy die mit der Proxy-Authentifizierungsanmeldeinformation für den externen Schlüsselspeicher signierte Anforderung authentifiziert hat.

Der Verbindungsvorgang beginnt mit der Verbindung Ihres benutzerdefinierten Schlüsselspeichers, die Verbindung eines externen Schlüsselspeichers mit seinem externen Proxy dauert jedoch etwa fünf Minuten. Eine Erfolgsmeldung des Verbindungsvorgangs bedeutet nicht, dass der externe Schlüsselspeicher verbunden ist. Um zu überprüfen, ob die Verbindung erfolgreich war, verwenden Sie die AWS KMS Konsole oder den DescribeCustomKeyStoresVorgang, um den Verbindungsstatus Ihres externen Schlüsselspeichers anzuzeigen.

Wenn der Verbindungsstatus lautet`FAILED`, wird ein Verbindungsfehlercode in der AWS KMS Konsole angezeigt und der `DescribeCustomKeyStore` Antwort hinzugefügt. Hilfe zur Interpretation von Verbindungsfehlercodes finden Sie unter Verbindungsfehlercodes für externe Schlüsselspeicher.

Connect zu Ihrem externen Schlüsselspeicher her und stellen Sie erneut eine Verbindung her

Sie können Ihren externen Schlüsselspeicher in der AWS KMS Konsole oder mithilfe des Vorgangs verbinden oder erneut verbinden. ConnectCustomKeyStore

Mithilfe der Konsole AWS KMS

Sie können die AWS KMS Konsole verwenden, um einen externen Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy zu verbinden.

1. Melden Sie sich bei der Konsole AWS Key Management Service (AWS KMS) an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (Benutzerdefinierte Schlüsselspeicher), External key stores (Externe Schlüsselspeicher) aus.
4. Wählen Sie die Zeile des externen Schlüsselspeichers aus, den Sie verbinden möchten.

   Wenn der Verbindungsstatus des externen Schlüsselspeichers FAILED (FEHLGESCHLAGEN) ist, müssen Sie die Verbindung des externen Schlüsselspeichers trennen, bevor Sie ihn verbinden.
5. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Connect (Verbinden) aus.

Der Verbindungsvorgang dauert in der Regel etwa fünf Minuten. Wenn der Vorgang abgeschlossen ist, ändert sich der Verbindungsstatus in CONNECTED (VERBUNDEN).

Wenn der Verbindungsstatus Failed (Fehlgeschlagen) ist, bewegen Sie den Mauszeiger über den Verbindungsstatus, um den Verbindungsfehlercode zu sehen, der die Ursache des Fehlers erklärt. Hilfe beim Beantworten eines Verbindungsfehlercodes finden Sie unter Verbindungsfehlercodes für externe Schlüsselspeicher. Um einen externen Schlüsselspeicher mit dem Verbindungsstatus Failed (Fehlgeschlagen) zu verbinden, müssen Sie zuerst den benutzerdefinierten Schlüsselspeicher trennen.

Verwenden der API AWS KMS

Verwenden Sie den ConnectCustomKeyStoreVorgang, um eine Verbindung zu einem getrennten externen Schlüsselspeicher herzustellen.

Vor der Verbindung muss der Verbindungsstatus des externen Schlüsselspeichers `DISCONNECTED` sein. Wenn der Verbindungsstatus `FAILED` lautet, trennen Sie den externen Schlüsselspeicher und stellen Sie anschließend die Verbindung her.

Der Verbindungsvorgang dauert etwa fünf Minuten. Wenn er nicht schnell fehlschlägt, gibt `ConnectCustomKeyStore` eine HTTP 200-Antwort und ein JSON-Objekt ohne Eigenschaften zurück. Diese erste Antwort gibt jedoch nicht an, dass die Verbindung erfolgreich war. Um festzustellen, ob der externe Schlüsselspeicher verbunden ist, sehen Sie sich den Verbindungsstatus in der DescribeCustomKeyStoresAntwort an.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

Identifizieren Sie den externen Schlüsselspeicher anhand der ID des benutzerdefinierten Schlüsselspeichers. Sie finden die ID auf der Seite Benutzerdefinierte Schlüsselspeicher in der Konsole oder mithilfe des DescribeCustomKeyStoresVorgangs. Ersetzen Sie vor Ausführung dieses Beispiels die Beispiel-ID durch eine gültige ID.

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

Der `ConnectCustomKeyStore`-Vorgang gibt den `ConnectionState` nicht in seiner Antwort zurück. Verwenden Sie den DescribeCustomKeyStoresVorgang, um zu überprüfen, ob der externe Schlüsselspeicher angeschlossen ist. Diese Produktion gibt standardmäßig alle benutzerdefinierten Schlüsselspeicher innerhalb des Kontos und der Region zurück. Sie können jedoch entweder den Parameter `CustomKeyStoreId` oder `CustomKeyStoreName` (aber nicht beide) verwenden, um die Antwort auf bestimmte benutzerdefinierte Schlüsselspeicher zu begrenzen. Der `ConnectionState`-Wert `CONNECTED` bedeutet, dass der externe Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy verbunden ist.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

Wenn der `ConnectionState`-Wert in der `DescribeCustomKeyStores`-Antwort `FAILED` lautet, gibt das `ConnectionErrorCode`-Element den Grund für den Fehler an.

Im folgenden Beispiel `ConnectionErrorCode` gibt der `XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND` Wert für an, dass der VPC-Endpunktdienst, den er für die Kommunikation mit dem externen Schlüsselspeicher-Proxy verwendet, nicht gefunden werden AWS KMS kann. Stellen Sie sicher, `XksProxyVpcEndpointServiceName` dass der AWS KMS Service Principal ein zulässiger Principal auf dem Amazon VPC-Endpunktservice ist und dass der VPC-Endpunktservice keine Annahme von Verbindungsanfragen erfordert. Hilfe beim Beantworten eines Verbindungsfehlercodes finden Sie unter Verbindungsfehlercodes für externe Schlüsselspeicher.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "FAILED",
      "ConnectionErrorCode": "XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

Trennen Sie die Verbindung zu einem externen Schlüsselspeicher

Wenn Sie einen externen Schlüsselspeicher mit Konnektivität eines VPC-Endpunkt-Service von seinem externen Schlüsselspeicher-Proxy trennen, löscht AWS KMS seinen Schnittstellen-Endpunkt zum VPC-Endpunkt-Service und entfernt die Netzwerkinfrastruktur, die zur Unterstützung der Verbindung erstellt wurde. Für externe Schlüsselspeicher mit öffentlicher Endpunktkonnektivität ist kein entsprechender Prozess erforderlich. Diese Aktion wirkt sich weder auf den VPC-Endpunkt-

Service oder eine seiner unterstützenden Komponenten noch auf den externen Schlüsselspeicher-Proxy oder externe Komponenten aus.

AWS KMS Sendet keine Anfragen an den externen Schlüsselspeicher-Proxy, solange die Verbindung zum externen Schlüsselspeicher getrennt ist. Der Verbindungsstatus des externen Schlüsselspeichers ist `DISCONNECTED`. Die KMS-Schlüssel im getrennten externen Schlüsselspeicher befinden sich in einem `UNAVAILABLE`-Schlüsselzustand (es sei denn, sie sind zum Löschen vorgesehen), was bedeutet, dass sie nicht für kryptografische Vorgänge verwendet werden können. Sie können Ihren externen Schlüsselspeicher und die vorhandenen KMS-Schlüssel jedoch weiterhin anzeigen und verwalten.

Der getrennte Zustand ist als vorübergehend und umkehrbar konzipiert. Sie können die Verbindung Ihres externen Schlüsselspeichers jederzeit wieder herstellen. Normalerweise ist keine Neukonfiguration erforderlich. Wenn sich jedoch Eigenschaften des zugehörigen externen Schlüsselspeicher-Proxys geändert haben, während die Verbindung getrennt war, z. B. die Rotation der Anmeldeinformation für die Proxy-Authentifizierung, müssen Sie die Einstellungen des externen Schlüsselspeichers vor der erneuten Verbindung bearbeiten.

> **Note**
>
> Sämtliche Versuche, KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zu erstellen oder vorhandene KMS-Schlüssel in kryptografischen Produktionen zu nutzen, schlagen fehl, während der benutzerdefinierte Schlüsselspeicher getrennt ist. Diese Aktion kann verhindern, dass Benutzer vertrauliche Daten speichern und darauf zugreifen.

Um die Auswirkung einer Trennung der Verbindung Ihres externen Schlüsselspeichers besser beurteilen zu können, ermitteln Sie die KMS-Schlüssel im externen Schlüsselspeicher und ihre bisherige Verwendung.

Die Verbindung eines externen Schlüsselspeichers könnte beispielsweise aus folgenden Gründen getrennt werden:

- Zum Bearbeiten seiner Eigenschaften. Sie können den Namen des benutzerdefinierten Schlüsselspeichers, den Proxy-URI-Pfad und die Anmeldeinformation für die Proxy-Authentifizierung bearbeiten, während der externe Schlüsselspeicher verbunden ist. Um jedoch den Typ der Proxy-Konnektivität, den Proxy-URI-Endpunkt oder den Namen des VPC-Endpunkt-Services zu bearbeiten, müssen Sie zunächst die Verbindung des externen Schlüsselspeichers

trennen. Details hierzu finden Sie unter Eigenschaften des externen Schlüsselspeichers bearbeiten.

- Um die gesamte Kommunikation zwischen AWS KMS und dem externen Schlüsselspeicher-Proxy zu beenden. Sie können auch die Kommunikation zwischen AWS KMS und Ihrem Proxy beenden, indem Sie Ihren Endpunkt oder VPC-Endpunktdienst deaktivieren. Darüber hinaus bietet Ihr externer Schlüsselspeicher-Proxy oder Ihre Schlüsselverwaltungssoftware möglicherweise zusätzliche Mechanismen, um die Kommunikation mit AWS KMS dem Proxy oder den Zugriff des Proxys auf Ihren externen Schlüsselmanager zu verhindern.
- Zum Deaktivieren aller KMS-Schlüssel im externen Schlüsselspeicher. Sie können KMS-Schlüssel in einem externen Schlüsselspeicher mithilfe der AWS KMS Konsole oder des DisableKeyVorgangs deaktivieren und erneut aktivieren. Diese Vorgänge werden schnell abgeschlossen (vorbehaltlich einer letztendlichen Konsistenz), beziehen sich jedoch immer nur auf jeweils einen KMS-Schlüssel. Wenn die Verbindung zum externen Schlüsselspeicher unterbrochen wird, ändert sich der Schlüsselstatus aller KMS-Schlüssel im externen Schlüsselspeicher in `Unavailable`, sodass sie in keinem kryptografischen Vorgang verwendet werden können.
- Zur Behebung eines fehlgeschlagenen Verbindungsversuchs. Wenn ein Versuch, eine Verbindung für einen externen Schlüsselspeicher herzustellen, fehlschlägt (Verbindungsstatus des benutzerdefinierten Schlüsselspeichers ist `FAILED`), müssen Sie die Verbindung des externen Schlüsselspeichers trennen, bevor Sie erneut versuchen, eine Verbindung herzustellen.

Trennen Sie die Verbindung zu Ihrem externen Schlüsselspeicher

Sie können die Verbindung zu Ihrem externen Schlüsselspeicher in der AWS KMS Konsole oder mithilfe des DisconnectCustomKeyStoreVorgangs trennen.

Mithilfe der AWS KMS Konsole

Sie können die AWS KMS Konsole verwenden, um einen externen Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy zu verbinden. Dieser Vorgang dauert etwa 5 Minuten.

1. Melden Sie sich bei der Konsole AWS Key Management Service (AWS KMS) an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (Benutzerdefinierte Schlüsselspeicher), External key stores (Externe Schlüsselspeicher) aus.

4. Wählen Sie die Zeile des externen Schlüsselspeichers aus, mit dem Sie die Verbindung trennen möchten.
5. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Disconnect (Verbindung trennen) aus.

Nach Abschluss der Produktion ändert sich der Verbindungsstatus von DISCONNECTING (VERBINDUNG WIRD GETRENNT) in DISCONNECTED (VERBINDUNG GETRENNT). Wenn die Produktion fehlschlägt, wird eine Fehlermeldung mit einer Beschreibung des Problems und Hilfestellung zur Fehlerbehebung angezeigt. Wenn Sie weitere Hilfe benötigen, beachten Sie den Abschnitt Fehler bei der Verbindung mit dem externen Schlüsselspeicher.

Verwenden der API AWS KMS

Verwenden Sie den DisconnectCustomKeyStoreVorgang, um die Verbindung zu einem verbundenen externen Schlüsselspeicher zu trennen. Wenn der Vorgang erfolgreich ist, werden eine HTTP 200-Antwort und ein JSON-Objekt ohne Eigenschaften AWS KMS zurückgegeben. Der Vorgang dauert etwa fünf Minuten. Verwenden Sie den DescribeCustomKeyStoresVorgang, um den Verbindungsstatus des externen Schlüsselspeichers zu ermitteln.

Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen.

In diesem Beispiel wird die Verbindung eines externen Schlüsselspeichers mit Konnektivität eines VPC-Endpunkt-Service getrennt. Vor der Ausführung dieses Beispiels müssen Sie die Beispiel-ID des benutzerdefinierten Schlüsselspeichers durch eine gültige ID ersetzen.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

Verwenden Sie den DescribeCustomKeyStoresVorgang, um zu überprüfen, ob der externe Schlüsselspeicher getrennt ist. Diese Produktion gibt standardmäßig alle benutzerdefinierten Schlüsselspeicher innerhalb des Kontos und der Region zurück. Sie können jedoch entweder den Parameter `CustomKeyStoreId` oder `CustomKeyStoreName` (aber nicht beide) verwenden, um die Antwort auf bestimmte benutzerdefinierte Schlüsselspeicher zu begrenzen. Der `ConnectionState`-Wert `DISCONNECTED` bedeutet, dass in diesem Beispiel der externe Schlüsselspeicher nicht mehr mit seinem externen Schlüsselspeicher-Proxy verbunden ist.

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
```

```
    "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",
      "ConnectionState": "DISCONNECTED",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

Löschen Sie einen externen Schlüsselspeicher

Wenn Sie einen externen Schlüsselspeicher AWS KMS löschen, werden alle Metadaten über den externen Schlüsselspeicher gelöscht AWS KMS, einschließlich Informationen über seinen externen Schlüsselspeicher-Proxy. Dieser Vorgang hat keine Auswirkungen auf den externen Schlüsselspeicher-Proxy, den externen Schlüsselmanager, externe Schlüssel oder andere AWS Ressourcen, die Sie zur Unterstützung des externen Schlüsselspeichers erstellt haben, wie z. B. eine Amazon VPC oder einen VPC-Endpunktservice.

Bevor Sie einen externen Schlüsselspeicher löschen, müssen Sie alle KMS-Schlüssel aus dem Schlüsselspeicher löschen und die Verbindung des Schlüsselspeichers mit seinem externen Schlüsselspeicher-Proxy trennen. Andernfalls schlagen Versuche, den Schlüsselspeicher zu löschen, fehl.

Das Löschen eines externen Schlüsselspeichers ist nicht rückgängig zu machen, aber Sie können einen neuen externen Schlüsselspeicher erstellen und ihn mit demselben externen Schlüsselspeicher-Proxy und externen Schlüsselmanager verknüpfen. Sie können die KMS-Schlüssel für die symmetrische Verschlüsselung jedoch nicht im externen Schlüsselspeicher neu erstellen, selbst wenn Sie Zugriff auf dasselbe externe Schlüsselmaterial haben. AWS KMS schließt Metadaten im symmetrischen Chiffretext ein, die für jeden KMS-Schlüssel einzigartig sind. Diese Sicherheitsfunktion sorgt dafür, dass nur der KMS-Schlüssel, der die Daten verschlüsselt hat, sie entschlüsseln kann.

Anstatt den externen Schlüsselspeicher zu löschen, sollten Sie dessen Verbindung trennen. Solange die Verbindung zu einem externen Schlüsselspeicher unterbrochen ist, können Sie den externen Schlüsselspeicher und dessen Inhalt verwalten, AWS KMS keys aber Sie können keine KMS-Schlüssel im externen Schlüsselspeicher erstellen oder verwenden. Sie können den externen Schlüsselspeicher jederzeit wieder verbinden und seine KMS-Schlüssel zum Ver- und Entschlüsseln von Daten verwenden. Für einen nicht verbundenen externen Schlüsselspeicher-Proxy oder seine nicht verfügbaren KMS-Schlüssel fallen keine Kosten an.

Sie können Ihren externen Schlüsselspeicher in der AWS KMS Konsole oder mithilfe des DeleteCustomKeyStoreVorgangs löschen.

Mithilfe der AWS KMS Konsole

Sie können die AWS KMS Konsole verwenden, um einen externen Schlüsselspeicher zu löschen.

1. Melden Sie sich bei der AWS Key Management Service (AWS KMS) -Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/kms.
2. Um das zu ändern AWS-Region, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Custom key stores (Benutzerdefinierte Schlüsselspeicher), External key stores (Externe Schlüsselspeicher) aus.
4. Suchen Sie die Zeile, die den externen Schlüsselspeicher darstellt, den Sie löschen möchten. Wenn der Verbindungsstatus des externen Schlüsselspeichers nicht DISCONNECTED (VERBINDUNG GETRENNT) ist, müssen Sie die Verbindung des externen Schlüsselspeichers trennen, bevor Sie ihn löschen.
5. Wählen Sie im Menü Key store actions (Schlüsselspeicheraktionen) die Option Delete (Löschen) aus.

Nach Abschluss des Vorgangs wird eine Erfolgsmeldung angezeigt und der externe Schlüsselspeicher wird nicht mehr in der Schlüsselspeicherliste angezeigt. Wenn die Produktion nicht erfolgreich ist, wird eine Fehlermeldung mit einer Beschreibung des Problems und Hilfestellung zur Fehlerbehebung angezeigt. Wenn Sie weitere Hilfe benötigen, beachten Sie den Abschnitt Fehlerbehebung bei externen Schlüsselspeichern.

Verwenden der API AWS KMS

Verwenden Sie den DeleteCustomKeyStoreVorgang, um einen externen Schlüsselspeicher zu löschen. Wenn der Vorgang erfolgreich ist, werden eine HTTP 200-Antwort und ein JSON-Objekt ohne Eigenschaften AWS KMS zurückgegeben.

Trennen Sie zunächst die Verbindung des externen Schlüsselspeichers. Vor der Ausführung dieses Befehls müssen Sie die Beispiel-ID des benutzerdefinierten Schlüsselspeichers durch eine gültige ID ersetzen.

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

Nachdem der externe Schlüsselspeicher getrennt wurde, können Sie ihn mithilfe des DeleteCustomKeyStoreVorgangs löschen.

```
$ aws kms delete-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

Verwenden Sie den DescribeCustomKeyStoresVorgang, um zu bestätigen, dass der externe Schlüsselspeicher gelöscht wurde.

```
$ aws kms describe-custom-key-stores

{
    "CustomKeyStores": []
}
```

Wenn Sie einen benutzerdefinierten Schlüsselspeicher-Namen oder eine ID angeben, die nicht mehr existiert, wird eine `CustomKeyStoreNotFoundException` Ausnahme AWS KMS zurückgegeben.

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0

An error occurred (CustomKeyStoreNotFoundException) when calling the
 DescribeCustomKeyStore operation:
```

Fehlerbehebung bei externen Schlüsselspeichern

Die meisten Probleme mit externen Schlüsselspeichern lassen sich anhand der Fehlermeldung beheben, die bei jeder Ausnahme AWS KMS angezeigt wird, oder anhand des

Verbindungsfehlercodes, der AWS KMS zurückgegeben wird, wenn ein Versuch, den externen Schlüsselspeicher mit seinem externen Schlüsselspeicher-Proxy zu verbinden, fehlschlägt. Einige Probleme sind jedoch etwas komplexer.

Wenn Sie ein Problem mit einem externen Schlüsselspeicher diagnostizieren, ermitteln Sie zunächst die Ursache. Dadurch wird die Bandbreite der potenziellen Abhilfemaßnahmen eingeschränkt und Ihre Fehlersuche wird effizienter.

- AWS KMS — Das Problem liegt möglicherweise darin AWS KMS, z. B. ein falscher Wert in der Konfiguration Ihres externen Schlüsselspeichers.
- Extern — Das Problem kann außerhalb von liegen AWS KMS, einschließlich Problemen mit der Konfiguration oder dem Betrieb des externen Schlüsselspeicher-Proxys, des externen Schlüsselmanagers, der externen Schlüssel oder des VPC-Endpunktdienstes.
- Netzwerk — Möglicherweise handelt es sich um ein Konnektivitäts- oder Netzwerkproblem, z. B. ein Problem mit Ihrem Proxy-Endpunkt, Port, IP-Stack oder Ihrem privaten DNS-Namen oder Ihrer privaten DNS-Domäne.

> **Note**
>
> Wenn Verwaltungsoperationen für externe Schlüsselspeicher fehlschlagen, generieren sie verschiedene Ausnahmen. AWS KMS Kryptografische Operationen werden jedoch `KMSInvalidStateException` für alle Fehler zurückgegeben, die mit der externen Konfiguration oder dem Verbindungsstatus des externen Schlüsselspeichers zusammenhängen. Identifizieren Sie das Problem anhand des zugehörigen Fehlermeldungstexts.
>
> Der ConnectCustomKeyStoreVorgang ist schnell erfolgreich, bevor der Verbindungsvorgang abgeschlossen ist. Um festzustellen, ob der Verbindungsvorgang erfolgreich ausgeführt wird, sehen Sie sich den Verbindungsstatus des externen Schlüsselspeichers an. Wenn der Verbindungsvorgang fehlschlägt, gibt AWS KMS einen Verbindungsfehlercode zurück, der Aufschluss über die Ursache gibt und Abhilfemaßnahmen vorschlägt.

Themen

- Tools zur Fehlerbehebung bei externen Schlüsselspeichern
- Konfigurationsfehler
- Fehler bei der Verbindung mit dem externen Schlüsselspeicher

- Latenz- und Zeitüberschreitungsfehler
- Fehler mit der Anmeldeinformation für die Authentifizierung
- Fehler mit dem Schlüsselstatus
- Entschlüsselungsfehler
- Fehler mit externen Schlüsseln
- Proxy-Probleme
- Probleme mit der Proxy-Autorisierung

Tools zur Fehlerbehebung bei externen Schlüsselspeichern

AWS KMS stellt mehrere Tools bereit, mit denen Sie Probleme mit Ihrem externen Schlüsselspeicher und seinen Schlüsseln identifizieren und lösen können. Verwenden Sie diese Tools zusammen mit den Tools, die mit Ihrem externen Schlüsselspeicher-Proxy und Ihrem externen Schlüsselmanager bereitgestellt werden.

> **Note**
>
> Ihr externer Schlüsselspeicher-Proxy und Ihr externer Schlüsselmanager bieten möglicherweise einfachere Methoden zum Erstellen und Verwalten Ihres externen Schlüsselspeichers und seiner KMS-Schlüssel. Weitere Informationen finden Sie in der Dokumentation Ihrer externen Tools.

AWS KMS Ausnahmen und Fehlermeldungen

AWS KMS bietet eine detaillierte Fehlermeldung zu allen auftretenden Problemen. Weitere Informationen zu AWS KMS Ausnahmen finden Sie in der AWS Key Management Service API-Referenz und AWS SDKs. Auch wenn Sie die AWS KMS Konsole verwenden, könnten diese Verweise für Sie hilfreich sein. Sehen Sie sich zum Beispiel die Fehlerliste für die `CreateCustomKeyStores`-Operation an.

Um die Leistung Ihres externen Schlüsselspeicher-Proxys zu optimieren, AWS KMS gibt es Ausnahmen auf der Grundlage der Zuverlässigkeit Ihres Proxys innerhalb eines bestimmten Aggregationszeitraums von 5 Minuten zurück. Bei einem internen Serverfehler vom Typ 500, 503 Service Unavailable oder einem Verbindungs-Timeout kehrt ein Proxy mit hoher Zuverlässigkeit zurück `KMSInternalException` und löst einen automatischen Wiederholungsversuch aus, um

sicherzustellen, dass Anfragen letztendlich erfolgreich sind. Ein Proxy mit niedriger Zuverlässigkeit kehrt jedoch zurück. `KMSInvalidStateException` Weitere Informationen finden Sie unter Überwachen eines externen Schlüsselspeichers.

Wenn das Problem in einem anderen AWS Dienst auftritt, z. B. wenn Sie einen KMS-Schlüssel in Ihrem externen Schlüsselspeicher verwenden, um eine Ressource in einem anderen AWS Dienst zu schützen, stellt der AWS Dienst möglicherweise zusätzliche Informationen zur Verfügung, mit denen Sie das Problem identifizieren können. Wenn der AWS Dienst die Meldung nicht bereitstellt, können Sie die Fehlermeldung in den CloudTrail Protokollen einsehen, in denen die Verwendung Ihres KMS-Schlüssels aufgezeichnet wird.

CloudTrail logs

Jeder AWS KMS API-Vorgang, einschließlich Aktionen in der AWS KMS Konsole, wird in AWS CloudTrail Protokollen aufgezeichnet. AWS KMS zeichnet einen Protokolleintrag für erfolgreiche und fehlgeschlagene Operationen auf. Bei fehlgeschlagenen Operationen enthält der Protokolleintrag den Namen der AWS KMS -Ausnahme (`errorCode`) und die Fehlermeldung (`errorMessage`). Sie können diese Informationen verwenden, um den Fehler zu identifizieren und zu beheben. Ein Beispiel finden Sie unter Fehler bei der Entschlüsselung mit einem KMS-Schlüssel in einem externen Schlüsselspeicher.

Der Protokolleintrag enthält auch die Anfrage-ID. Wenn die Anfrage Ihren externen Schlüsselspeicher-Proxy erreicht hat, können Sie mithilfe der Anfrage-ID im Protokolleintrag die entsprechende Anfrage in Ihren Proxy-Protokollen finden, sofern Ihr Proxy sie bereitstellt.

CloudWatch Metriken

AWS KMS zeichnet detaillierte CloudWatch Amazon-Metriken über den Betrieb und die Leistung Ihres externen Schlüsselspeichers auf, darunter Latenz, Drosselung, Proxyfehler, den Status des externen Schlüsselmanagers, die Anzahl der Tage bis zum Ablauf Ihres TLS-Zertifikats und das gemeldete Alter Ihrer Proxy-Authentifizierungsdaten. Sie können diese Metriken verwenden, um Datenmodelle für den Betrieb Ihres externen Schlüsselspeichers und CloudWatch Alarme zu entwickeln, die Sie vor drohenden Problemen warnen, bevor sie auftreten.

> **Important**
>
> AWS KMS empfiehlt, dass Sie CloudWatch Alarme erstellen, um die Messwerte des externen Schlüsselspeichers zu überwachen. Diese Alarme warnen Sie vor ersten Anzeichen von Problemen, bevor sie auftreten.

Überwachungsdiagramme

AWS KMS zeigt auf der Detailseite für jeden externen Schlüsselspeicher in der AWS KMS Konsole Diagramme der CloudWatch Messwerte für externe Schlüsselspeicher an. Sie können die Daten in den Diagrammen verwenden, um die Fehlerquelle zu lokalisieren, drohende Probleme zu erkennen, Ausgangswerte festzulegen und Ihre CloudWatch Alarmschwellenwerte zu verfeinern. Einzelheiten zur Interpretation der Überwachungsdiagramme und zur Verwendung der darin enthaltenen Daten finden Sie unter Überwachen Sie externe Schlüsselspeicher.

Anzeigen von externen Schlüsselspeichern und KMS-Schlüsseln

AWS KMS zeigt detaillierte Informationen zu Ihren externen Schlüsselspeichern und den KMS-Schlüsseln im externen Schlüsselspeicher in der AWS KMS Konsole sowie in der Antwort auf die AND-Operationen an. DescribeCustomKeyStoresDescribeKey Diese Anzeigen enthalten spezielle Felder für externe Schlüsselspeicher und KMS-Schlüssel mit Informationen, die Sie für die Problembehandlung verwenden können, z. B. den Verbindungsstatus des externen Schlüsselspeichers und die ID des externen Schlüssels, der dem KMS-Schlüssel zugeordnet ist. Details hierzu finden Sie unter Externe Schlüsselspeicher anzeigen.

XKS-Proxy-Testclient

AWS KMS stellt einen Open-Source-Testclient bereit, der überprüft, ob Ihr externer Schlüsselspeicher-Proxy der API-Spezifikation für den AWS KMS externen Schlüsselspeicher-Proxy entspricht. Sie können diesen Testclient verwenden, um Probleme mit Ihrem externen Schlüsselspeicher-Proxy zu identifizieren und zu beheben.

Konfigurationsfehler

Wenn Sie einen externen Schlüsselspeicher erstellen, geben Sie Eigenschaftswerte an, die die Konfiguration Ihres externen Schlüsselspeichers umfassen. Dazu zählen beispielsweise die Proxy-Authentifizierungsanmeldeinformation, der Proxy-URI-Endpunkt, der Proxy-URI-Pfad und der Name des VPC-Endpunktservice. Wenn ein Fehler in einem Eigenschaftswert AWS KMS erkannt wird, schlägt der Vorgang fehl und es wird ein Fehler zurückgegeben, der auf den fehlerhaften Wert hinweist.

Viele Konfigurationsprobleme lassen sich beheben, indem der falsche Wert korrigiert wird. Sie können einen ungültigen Proxy-URI-Pfad oder eine ungültige Proxy-Authentifizierungsanmeldeinformation korrigieren, ohne den externen Schlüsselspeicher zu trennen. Definitionen dieser Werte, einschließlich der Anforderungen hinsichtlich der Eindeutigkeit, finden Sie

unter Erfüllen der Voraussetzungen. Anweisungen zum Aktualisieren dieser Werte finden Sie unter Eigenschaften des externen Schlüsselspeichers bearbeiten.

Laden Sie beim Erstellen oder Aktualisieren Ihres externen Schlüsselspeichers eine Proxy-Konfigurationsdatei in die AWS KMS -Konsole hoch, um Fehler bei den Werten für den Proxy-URI-Pfad und die Proxy-Authentifizierungsanmeldeinformation zu vermeiden. Dies ist eine JSON-basierte Datei mit den Werten für den Proxy-URI-Pfad und die Proxy-Authentifizierungsanmeldeinformation, die von Ihrem externen Schlüsselspeicher-Proxy oder vom externen Schlüsselmanager bereitgestellt werden. Sie können keine Proxy-Konfigurationsdatei für AWS KMS API-Operationen verwenden, aber Sie können die Werte in der Datei verwenden, um Parameterwerte für Ihre API-Anfragen bereitzustellen, die den Werten in Ihrem Proxy entsprechen.

Allgemeine Konfigurationsfehler

Ausnahmen: `CustomKeyStoreInvalidStateException` (`CreateKey`), `KMSInvalidStateException` (kryptografische Operationen), `XksProxyInvalidConfigurationException` (Verwaltungsoperationen, außer `CreateKey`)

Verbindungsfehlercodes: `XKS_PROXY_INVALID_CONFIGURATION`, `XKS_PROXY_INVALID_TLS_CONFIGURATION`

AWS KMS Testet bei externen Schlüsselspeichern mit öffentlicher Endpunktkonnektivität die Eigenschaftswerte, wenn Sie den externen Schlüsselspeicher erstellen und aktualisieren. Für externe Schlüsselspeicher mit der Konnektivität eines VPC-Endpunktservice testet AWS KMS die Eigenschaftswerte, wenn Sie den externen Schlüsselspeicher verbinden und aktualisieren.

> **Note**
>
> Die `ConnectCustomKeyStore`-Operation ist asynchron und kann auch dann erfolgreich ausgeführt werden, wenn sich der externe Schlüsselspeicher nicht mit seinem externen Schlüsselspeicher-Proxy verbinden lässt. In diesem Fall gibt es keine Ausnahme, aber der Verbindungsstatus des externen Schlüsselspeichers ist fehlgeschlagen und ein Verbindungsfehlercode erklärt die Fehlermeldung. Weitere Informationen finden Sie unter Fehler bei der Verbindung mit dem externen Schlüsselspeicher.

Wenn ein Fehler in einem Eigenschaftswert AWS KMS erkannt wird, schlägt der Vorgang fehl `XksProxyInvalidConfigurationException` und es wird eine der folgenden Fehlermeldungen angezeigt.

Der externe Schlüsselspeicher-Proxy hat die Anfrage aufgrund eines ungültigen URI-Pfads abgelehnt. Überprüfen Sie den URI-Pfad für Ihren externen Schlüsselspeicher und aktualisieren Sie ihn gegebenenfalls.

- Der Proxy-URI-Pfad ist der Basispfad für AWS KMS Anfragen an den Proxy APIs. Wenn dieser Pfad falsch ist, schlagen alle Anfragen an den Proxy fehl. Verwenden Sie die AWS KMS -Konsole oder die `DescribeCustomKeyStores`-Operation, um den aktuellen Proxy-URI-Pfad für Ihren externen Schlüsselspeicher anzuzeigen. Wie Sie den richtigen Proxy-URI-Pfad finden, erfahren Sie in der Dokumentation für Ihren externen Schlüsselspeicher-Proxy. Informationen zum Korrigieren des Werts für den Proxy-URI-Pfad finden Sie unter Eigenschaften des externen Schlüsselspeichers bearbeiten.
- Der Proxy-URI-Pfad für Ihren externen Schlüsselspeicher-Proxy kann sich ändern, wenn Ihr externer Schlüsselspeicher-Proxy oder der externe Schlüsselmanager aktualisiert wird. Informationen zu diesen Änderungen finden Sie in der Dokumentation für Ihren externen Schlüsselspeicher-Proxy oder externen Schlüsselmanager.

`XKS_PROXY_INVALID_TLS_CONFIGURATION`
AWS KMS kann keine TLS-Verbindung zum externen Schlüsselspeicher-Proxy herstellen. Überprüfen Sie die TLS-Konfiguration, einschließlich des Zertifikats.

- Für alle externen Schlüsselspeicher-Proxys ist ein TLS-Zertifikat erforderlich. Das TLS-Zertifikat muss von einer öffentlichen Zertifizierungsstelle (CA) ausgestellt werden, die für externe Schlüsselspeicher unterstützt wird. Eine Liste der unterstützten CAs Zertifizierungsstellen finden Sie unter Vertrauenswürdige Zertifizierungsstellen in der AWS KMS External Key Store Proxy API-Spezifikation.
- Für die Konnektivität eines öffentlichen Endpunkts muss der Subject Common Name (CN) auf dem TLS-Zertifikat mit dem Domainnamen im Proxy-URI-Endpunkt für den externen Schlüsselspeicher-Proxy identisch sein. Ist der öffentliche Endpunkt beispielsweise https://myproxy.xks.example.com, muss der CN auf dem TLS-Zertifikat `myproxy.xks.example.com` oder `*.xks.example.com` lauten.
- Für die Konnektivität eines VPC-Endpunktservice muss der Subject Common Name (CN) auf dem TLS-Zertifikat mit dem privaten DNS-Namen für Ihren VPC-Endpunktservice übereinstimmen. Wenn der private DNS-Name beispielsweise myproxy-private.xks.example.com ist, muss der CN

auf dem TLS-Zertifikat `myproxy-private.xks.example.com` oder `*.xks.example.com` lauten.

- Das TLS-Zertifikat darf nicht abgelaufen sein. Verwenden Sie SSL-Tools wie OpenSSL, um das Ablaufdatum eines TLS-Zertifikats zu ermitteln. Verwenden Sie die XksProxyCertificateDaysToExpire CloudWatch Metrik, um das Ablaufdatum eines TLS-Zertifikats zu überwachen, das einem externen Schlüsselspeicher zugeordnet ist. Die Anzahl der Tage bis zum Ablaufdatum Ihrer TLS-Zertifizierung wird auch im Bereich Überwachung der AWS KMS Konsole angezeigt.
- Wenn Sie die Konnektivität eines öffentlichen Endpunkts verwenden, testen Sie Ihre SSL-Konfiguration mithilfe von SSL-Testtools. TLS-Verbindungsfehler können durch eine falsche Zertifikatverkettung verursacht werden.

Konfigurationsfehler bei der Konnektivität eines VPC-Endpunktservice

Ausnahmen: `XksProxyVpcEndpointServiceNotFoundException`, `XksProxyVpcEndpointServiceInvalidConfigurationException`

Zusätzlich zu allgemeinen Verbindungsproblemen können beim Erstellen, Verbinden oder Aktualisieren eines externen Schlüsselspeichers mit VPC-Endpunktdienst-Konnektivität die folgenden Probleme auftreten. AWS KMS testet die Eigenschaftswerte eines externen Schlüsselspeichers mit VPC-Endpunktdienst-Konnektivität, während der externe Schlüsselspeicher erstellt, verbunden und aktualisiert wird. Wenn Verwaltungsoperationen aufgrund von Konfigurationsfehlern fehlschlagen, werden die folgenden Ausnahmen generiert:

```
XksProxyVpcEndpointServiceNotFoundException
```

Dies kann folgende Ursachen haben:

- Ein falscher Name des VPC-Endpunktservice. Stellen Sie sicher, dass der Name des VPC-Endpunktservice für den externen Schlüsselspeicher korrekt ist und mit dem Wert des Proxy-URI-Endpunkts für den externen Schlüsselspeicher übereinstimmt. Um den Namen des VPC-Endpunktdienstes zu finden, verwenden Sie die Amazon VPC-Konsole oder den DescribeVpcEndpointServicesVorgang. Verwenden Sie die AWS KMS Konsole oder den DescribeCustomKeyStoresVorgang, um den VPC-Endpunktdienstnamen und den Proxy-URI-Endpunkt eines vorhandenen externen Schlüsselspeichers zu ermitteln. Details hierzu finden Sie unter Externe Schlüsselspeicher anzeigen.

- Der VPC-Endpunktdienst befindet sich möglicherweise in einem anderen AWS-Region als dem externen Schlüsselspeicher. Stellen Sie sicher, dass sich der VPC-Endpunktservice und der externe Schlüsselspeicher in derselben Region befinden. (Der externe Name des Regionsnamens, z. B.`us-east-1`, ist Teil des VPC-Endpunktdienstnamens, z. B. com.amazonaws.vpce.us-east-1.vpce-svc-example.) Eine Liste der Anforderungen für den VPC-Endpunktservice für einen externen Schlüsselspeicher finden Sie unter VPC-Endpunktservice. Ein VPC-Endpunktservice oder ein externer Schlüsselspeicher lässt sich nicht in eine andere Region verschieben. Sie können jedoch einen neuen externen Schlüsselspeicher in derselben Region wie der VPC-Endpunktservice erstellen. Details dazu finden Sie unter VPC-Endpunktdienstkonnektivität konfigurieren und Erstellen Sie einen externen Schlüsselspeicher.
- AWS KMS ist kein zulässiger Principal für den VPC-Endpunktdienst. Die Liste der zulässigen Prinzipale für den VPC-Endpunktservice muss den `cks.kms.<region>.amazonaws.com`-Wert enthalten, etwa `cks.kms.eu-west-3.amazonaws.com`. Anweisungen zum Hinzufügen dieses Werts finden Sie unter Verwalten von Berechtigungen im AWS PrivateLink -Handbuch.

XksProxyVpcEndpointServiceInvalidConfigurationException

Dieser Fehler tritt auf, wenn der VPC-Endpunktservice eine der folgenden Anforderungen nicht erfüllt:

- Die VPC muss über mindestens zwei private Subnetze verfügen, die in verschiedenen Availability Zones sind. Weitere Informationen zum Hinzufügen eines Subnetzes zu Ihrer VPC finden Sie unter Erstellen eines Subnetzes in der VPC im Amazon VPC-Benutzerhandbuch.
- Der Typ es VPC-Endpunktservice muss einen Network Load Balancer verwenden, keinen Gateway Load Balancer.
- Für den VPC-Endpunktservice darf keine Akzeptanz erforderlich sein (Akzeptanz erforderlich muss auf falsch gesetzt sein.). Wenn jede Verbindungsanforderung manuell akzeptiert werden muss, AWS KMS kann der VPC-Endpunktdienst nicht verwendet werden, um eine Verbindung zum externen Schlüsselspeicher-Proxy herzustellen. Einzelheiten hierzu finden Sie unter Annehmen oder Ablehnen von Verbindungsanforderungen im AWS PrivateLink -Handbuch.
- Der VPC-Endpunktservice muss einen privaten DNS-Namen haben, der eine Subdomain einer öffentlichen Domain ist. Lautet der private DNS-Name beispielsweise `https://myproxy-private.xks.example.com`, müssen die Domains `xks.example.com` und `example.com` über einen öffentlichen DNS-Server verfügen. Informationen zum Anzeigen oder Ändern des

privaten DNS-Namens für Ihren VPC-Endpunktservice finden Sie unter Verwalten von DNS-Namen für VPC-Endpunktservices im AWS PrivateLink -Handbuch.

- Der Domain-Verifizierungsstatus der Domain für Ihren privaten DNS-Namen muss `verified` lauten. Informationen zum Anzeigen und Aktualisieren des Verifizierungsstatus der Domain für Ihren privaten DNS-Namen finden Sie unter Schritt 5: Überprüfen Sie Ihre private DNS-Namensdomäne. Nachdem Sie den erforderlichen Textdatensatz hinzugefügt haben, kann es einige Minuten dauern, bis der aktualisierte Verifizierungsstatus angezeigt wird.

> **Note**
>
> Eine private DNS-Domain kann nur verifiziert werden, wenn es sich um die Subdomain einer öffentlichen Domain handelt. Andernfalls ändert sich der Verifizierungsstatus der privaten DNS-Domain nicht, auch wenn Sie den erforderlichen TXT-Datensatz hinzugefügt haben.

- Stellen Sie sicher, dass alle Firewalls zwischen AWS KMS und dem externen Schlüsselspeicher-Proxy den Datenverkehr zu und von Port 443 auf dem Proxy zulassen. AWS KMS kommuniziert auf Port 443 über. IPv4 Dieser Wert kann nicht konfiguriert werden.
- Der private DNS-Name des VPC-Endpunktservice muss mit dem Wert des Proxy-URI-Endpunkts für den externen Schlüsselspeicher übereinstimmen. Für einen externen Schlüsselspeicher mit der Konnektivität eines VPC-Endpunktservice muss der Proxy-URI-Endpunkt `https://` gefolgt vom privaten DNS-Namen des VPC-Endpunktservice sein. Informationen zum Anzeigen des Werts für den Proxy-URI-Endpunkt finden Sie unter Externe Schlüsselspeicher anzeigen. Informationen zum Ändern des Werts für den Proxy-URI-Endpunkt finden Sie unter Eigenschaften des externen Schlüsselspeichers bearbeiten.

Fehler bei der Verbindung mit dem externen Schlüsselspeicher

Das Verbinden eines externen Schlüsselspeichers mit seinem externen Schlüsselspeicher-Proxy dauert etwa fünf Minuten. Sofern sie nicht schnell fehlschlägt, gibt die `ConnectCustomKeyStore`-Operation eine HTTP-Antwort 200 und ein JSON-Objekt ohne Eigenschaften zurück. Diese erste Antwort gibt jedoch nicht an, dass die Verbindung erfolgreich war. Um festzustellen, ob der externe Schlüsselspeicher verbunden ist, sehen Sie sich den Verbindungsstatus an. Wenn die Verbindung fehlschlägt, ändert sich der Verbindungsstatus des externen Schlüsselspeichers in einen Verbindungsfehlercode, der die Ursache des Fehlers erklärt, `FAILED` und es wird ein Verbindungsfehlercode AWS KMS zurückgegeben.

> **Note**
>
> Wenn der Verbindungsstatus eines benutzerdefinierten Schlüsselspeichers `FAILED` ist, müssen Sie den benutzerdefinierten Schlüsselspeicher trennen, bevor Sie versuchen, ihn wieder zu verbinden. Ein benutzerdefinierter Schlüsselspeicher mit dem Verbindungsstatus `FAILED` kann nicht verbunden werden.

Anzeigen des Verbindungsstatus eines externen Schlüsselspeichers:

- Sehen Sie sich in der DescribeCustomKeyStoresAntwort den Wert des `ConnectionState` Elements an.
- In der AWS KMS Konsole wird der Verbindungsstatus in der Tabelle mit dem externen Schlüsselspeicher angezeigt. Außerdem wird auf der Detailseite für jeden externen Schlüsselspeicher der Verbindungsstatus im Abschnitt Allgemeine Konfiguration angezeigt.

Wenn der Verbindungsstatus `FAILED` lautet, lässt sich mithilfe des Verbindungsfehlercodes der Fehler erklären.

Anzeigen des Verbindungsfehlercodes:

- Sehen Sie sich in der DescribeCustomKeyStoresAntwort den Wert des `ConnectionErrorCode` Elements an. Die `DescribeCustomKeyStores`-Antwort enthält dieses Element nur, wenn der `ConnectionState` `FAILED` ist.
- Um den Verbindungsfehlercode in der AWS KMS Konsole auf der Detailseite für den externen Schlüsselspeicher anzuzeigen, zeigen Sie mit der Maus auf den Wert Fehlgeschlagen.

Verbindungsfehlercodes für externe Schlüsselspeicher

Die folgenden Verbindungsfehlercodes gelten für externe Schlüsselspeicher

`INTERNAL_ERROR`

AWS KMS konnte die Anfrage aufgrund eines internen Fehlers nicht abschließen. Wiederholen Sie die Anforderung. Trennen Sie bei `ConnectCustomKeyStore`-Anforderungen den benutzerdefinierten Schlüsselspeicher, bevor Sie die Verbindung wiederherstellen.

`INVALID_CREDENTIALS`

Einer oder beide `XksProxyAuthenticationCredential`-Werte sind auf dem angegebenen externen Schlüsselspeicher-Proxy ungültig.

`NETWORK_ERRORS`

Netzwerkfehler AWS KMS verhindern, dass der benutzerdefinierte Schlüsselspeicher mit seinem Backing-Schlüsselspeicher verbunden werden kann.

`XKS_PROXY_ACCESS_DENIED`

AWS KMS Anfragen wird der Zugriff auf den externen Schlüsselspeicher-Proxy verweigert. Wenn es für den externen Schlüsselspeicher-Proxy Autorisierungsregeln gibt, stellen Sie sicher, dass diese zulassen, dass AWS KMS in Ihrem Namen mit dem Proxy kommuniziert.

`XKS_PROXY_INVALID_CONFIGURATION`

Ein Konfigurationsfehler verhindert, dass der externe Schlüsselspeicher eine Verbindung zu seinem Proxy herstellt. Überprüfen Sie den Wert von `XksProxyUriPath`.

`XKS_PROXY_INVALID_RESPONSE`

AWS KMS kann die Antwort des externen Schlüsselspeicher-Proxys nicht interpretieren. Wenn Sie diesen Verbindungsfehlercode wiederholt sehen, benachrichtigen Sie den Proxy-Anbieter Ihres externen Schlüsselspeichers.

`XKS_PROXY_INVALID_TLS_CONFIGURATION`

AWS KMS kann keine Verbindung zum externen Schlüsselspeicher-Proxy herstellen, da die TLS-Konfiguration ungültig ist. Stellen Sie sicher, dass der externe Schlüsselspeicher-Proxy TLS 1.2 oder 1.3 unterstützt. Stellen Sie außerdem sicher, dass das TLS-Zertifikat nicht abgelaufen ist, dass es mit dem Hostnamen im `XksProxyUriEndpoint`-Wert übereinstimmt und dass es von einer vertrauenswürdigen Zertifizierungsstelle signiert wurde, die in der Liste der vertrauenswürdigen Zertifizierungsstellen enthalten ist.

`XKS_PROXY_NOT_REACHABLE`

AWS KMS kann nicht mit Ihrem externen Schlüsselspeicher-Proxy kommunizieren. Stellen Sie sicher, dass der `XksProxyUriEndpoint` und der `XksProxyUriPath` korrekt sind. Überprüfen Sie mithilfe der Tools für Ihren externen Schlüsselspeicher-Proxy, ob der Proxy aktiv und in seinem Netzwerk verfügbar ist. Stellen Sie außerdem sicher, dass die Instances Ihres externen Schlüsselmanagers ordnungsgemäß funktionieren. Verbindungsversuche schlagen mit

diesem Verbindungsfehlercode fehl, wenn der Proxy meldet, dass keine Instance von externen Schlüsselmanagern verfügbar ist.

`XKS_PROXY_TIMED_OUT`

AWS KMS kann eine Verbindung zum externen Schlüsselspeicher-Proxy herstellen, aber der Proxy reagiert nicht AWS KMS in der vorgesehenen Zeit. Wenn Sie diesen Verbindungsfehlercode wiederholt sehen, benachrichtigen Sie den Proxy-Anbieter Ihres externen Schlüsselspeichers.

`XKS_VPC_ENDPOINT_SERVICE_INVALID_CONFIGURATION`

Die Konfiguration des Amazon VPC-Endpunktdienstes entspricht nicht den Anforderungen für einen AWS KMS externen Schlüsselspeicher.

- Der VPC-Endpunktservice muss ein Endpunktservice für Schnittstellen-Endpunkte im AWS-Konto des Aufrufers sein.
- Er muss über einen Network Load Balancer (NLB) verfügen, der mit mindestens zwei Subnetzen verbunden ist, die jeweils in einer anderen Availability Zone liegen.
- Die `Allow principals` Liste muss den AWS KMS Service Principal für die Region enthalten`cks.kms.<region>.amazonaws.com`, z. B. `cks.kms.us-east-1.amazonaws.com`
- Er darf keine Annahme von Verbindungsanforderungen verlangen.
- Er muss einen privaten DNS-Namen haben. Der private DNS-Name für einen externen Schlüsselspeicher mit `VPC_ENDPOINT_SERVICE`-Konnektivität muss in seiner AWS-Region eindeutig sein.
- Der Verifizierungsstatus der Domain des privaten DNS-Namens muss `verified` lauten.
- Das TLS-Zertifikat gibt den privaten DNS-Hostnamen an, unter dem der Endpunkt erreichbar ist.

`XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND`

AWS KMS kann den VPC-Endpunktdienst, den er für die Kommunikation mit dem externen Schlüsselspeicher-Proxy verwendet, nicht finden. Stellen Sie sicher, dass der `XksProxyVpcEndpointServiceName` korrekt ist und der AWS KMS -Service-Prinzipal über Service-Verbraucher-Berechtigungen für den Amazon VPC-Endpunktservice verfügt.

Latenz- und Zeitüberschreitungsfehler

Ausnahmen: `CustomKeyStoreInvalidStateException` (`CreateKey`), `KMSInvalidStateException` (kryptografische Operationen), `XksProxyUriUnreachableException` (Verwaltungsoperationen)

Verbindungsfehlercodes: `XKS_PROXY_NOT_REACHABLE`, `XKS_PROXY_TIMED_OUT`

Wenn der Proxy innerhalb des Timeout-Intervalls von 250 Millisekunden nicht kontaktiert werden AWS KMS kann, wird eine Ausnahme zurückgegeben. `CreateCustomKeyStore`und `UpdateCustomKeyStore` zurück. `XksProxyUriUnreachableException` Kryptografische Operationen geben den Standard `KMSInvalidStateException` mit einer Fehlermeldung zurück, die das Problem beschreibt. Schlägt das Problem `ConnectCustomKeyStore` fehl, wird ein Verbindungsfehlercode AWS KMS zurückgegeben, der das Problem beschreibt.

Zeitüberschreitungsfehler können vorübergehende Probleme sein, die sich durch das Wiederholen der Anforderung beheben lassen. Wenn dieses Problem weiterhin auftritt, überprüfen Sie, ob Ihr externer Schlüsselspeicher-Proxy aktiv und mit dem Netzwerk verbunden ist und ob sein Proxy-URI-Endpunkt, der Proxy-URI-Pfad und der Name des VPC-Endpunktservice (sofern vorhanden) in Ihrem externen Schlüsselspeicher korrekt sind. Stellen Sie außerdem sicher, dass sich Ihr externer Schlüsselmanager in der Nähe des AWS-Region für Ihren externen Schlüsselspeicher befindet. Wenn Sie einen dieser Werte aktualisieren müssen, finden Sie unter Eigenschaften des externen Schlüsselspeichers bearbeiten weitere Informationen.

Um Latenzmuster zu verfolgen, verwenden Sie die `XksProxyLatency` CloudWatch Metrik und das Diagramm mit der durchschnittlichen Latenz (basierend auf dieser Metrik) im Bereich Überwachung der AWS KMS Konsole. Ihr externer Schlüsselspeicher-Proxy generiert möglicherweise ebenfalls Protokolle und Metriken, die Latenz und Zeitüberschreitungen erfassen.

`XksProxyUriUnreachableException`
AWS KMS kann nicht mit dem externen Schlüsselspeicher-Proxy kommunizieren. Dies könnte ein vorübergehendes Netzwerkproblem sein. Wenn dieser Fehler wiederholt auftritt, überprüfen Sie, ob Ihr externer Schlüsselspeicher-Proxy aktiv und mit dem Netzwerk verbunden ist und ob sein Endpunkt-URI in Ihrem externen Schlüsselspeicher korrekt sind.

- Der externe Schlüsselspeicher-Proxy hat innerhalb des Timeout-Intervalls von 250 Millisekunden nicht auf eine AWS KMS Proxy-API-Anfrage geantwortet. Dies kann auf ein vorübergehendes

Netzwerkproblem oder ein Betriebs- oder Leistungsproblem mit dem Proxy hinweisen. Wenn ein erneuter Versuch das Problem nicht löst, benachrichtigen Sie den Administrator für Ihren externen Schlüsselspeicher-Proxy.

Latenz- und Zeitüberschreitungsfehler äußern sich häufig als Verbindungsfehler. Wenn der ConnectCustomKeyStoreVorgang fehlschlägt, ändert sich der Verbindungsstatus des externen Schlüsselspeichers in einen Verbindungsfehlercode, der den Fehler `FAILED` erklärt, und es wird ein Verbindungsfehlercode AWS KMS zurückgegeben. Eine Liste der Verbindungsfehlercodes und Vorschläge zur Behebung der Fehler finden Sie unter Verbindungsfehlercodes für externe Schlüsselspeicher. Die Verbindungscodelisten für alle benutzerdefinierten Schlüsselspeicher und externe Schlüsselspeicher gelten für externe Schlüsselspeicher. Die folgenden Verbindungsfehler stehen im Zusammenhang mit der Latenz und Zeitüberschreitungen.

`XKS_PROXY_NOT_REACHABLE`

–oder–

`CustomKeyStoreInvalidStateException`, `KMSInvalidStateException`, `XksProxyUriUnreachableException`
AWS KMS kann nicht mit dem externen Schlüsselspeicher-Proxy kommunizieren. Überprüfen Sie, ob Ihr externer Schlüsselspeicher-Proxy aktiv und mit dem Netzwerk verbunden ist und ob sein URI-Pfad und Endpunkt-URI oder der Name des VPC-Service in Ihrem externen Schlüssel speicher korrekt sind.

Dieser Fehler kann aus folgenden Gründen auftreten:

- Der externe Schlüsselspeicher-Proxy ist nicht aktiv bzw. nicht mit dem Netzwerk verbunden.
- Bei den Werten für den Proxy-URI-Endpunkt, den Proxy-URI-Pfad oder den Namen des VPC-Endpunktservice (sofern zutreffend) in der Konfiguration des externen Schlüsselspeichers ist ein Fehler aufgetreten. Um die Konfiguration des externen Schlüsselspeichers anzuzeigen, verwenden Sie den DescribeCustomKeyStoresVorgang oder rufen Sie die Detailseite für den externen Schlüsselspeicher in der AWS KMS Konsole auf.
- Möglicherweise liegt ein Netzwerkkonfigurationsfehler, z. B. ein Portfehler, auf dem Netzwerkpfad zwischen dem externen Schlüsselspeicher-Proxy AWS KMS und dem externen Schlüsselspeicher-Proxy vor. AWS KMS kommuniziert mit dem externen Schlüsselspeicher-Proxy auf Port 443 über IPv4. Dieser Wert kann nicht konfiguriert werden.

- Wenn der externe Schlüsselspeicher-Proxy (in einer GetHealthStatusAntwort) meldet, dass alle externen Schlüsselmanager-Instanzen vorhanden sind`UNAVAILABLE`, schlägt der ConnectCustomKeyStoreVorgang mit einem `ConnectionErrorCode` von `XKS_PROXY_NOT_REACHABLE` fehl. Weitere Informationen hierzu finden Sie in der Dokumentation Ihres externen Schlüsselmanagers.
- Dieser Fehler kann auf eine große physische Entfernung zwischen dem externen Schlüsselmanager und AWS-Region dem externen Schlüsselspeicher zurückzuführen sein. Die Ping-Latenz (Network Round-Trip Time (RTT)) zwischen dem AWS-Region und dem externen Schlüsselmanager sollte nicht mehr als 35 Millisekunden betragen. Möglicherweise müssen Sie einen externen Schlüsselspeicher in einem Bereich einrichten AWS-Region , der sich näher am externen Schlüsselmanager befindet, oder Sie müssen den externen Schlüsselmanager in ein Rechenzentrum verschieben, das näher am. AWS-Region

`XKS_PROXY_TIMED_OUT`

–oder–

`CustomKeyStoreInvalidStateException` , `KMSInvalidStateException` , `XksProxyUriUnreachableException`
AWS KMS hat die Anforderung abgelehnt, da der externe Schlüsselspeicher-Proxy nicht rechtzeitig geantwortet hat. Wiederholen Sie die Anforderung. Wenn Sie diesen Fehler wiederholt sehen, melden Sie ihn dem Administrator für Ihren externen Schlüsselspeicher-Proxy.

Dieser Fehler kann aus folgenden Gründen auftreten:

- Dieser Fehler kann darauf zurückzuführen sein, dass der externe Schlüsselmanager und der externe Schlüsselspeicher-Proxy geografisch weit voneinander entfernt sind. Bringen Sie den externen Schlüsselspeicher-Proxy nach Möglichkeit näher an den externen Schlüsselmanager.
- Timeoutfehler können auftreten, wenn der Proxy nicht darauf ausgelegt ist, das Volumen und die Häufigkeit der Anfragen von AWS KMS zu verarbeiten. Wenn Ihre CloudWatch Messwerte auf ein anhaltendes Problem hinweisen, benachrichtigen Sie Ihren Proxyadministrator für den externen Schlüsselspeicher.
- Zeitüberschreitungsfehler können auftreten, wenn die Verbindung zwischen dem externen Schlüsselmanager und der Amazon VPC für den externen Schlüsselspeicher nicht ordnungsgemäß funktioniert. Wenn Sie verwenden AWS Direct Connect, stellen Sie sicher, dass Ihre VPC und der

externe Schlüsselmanager effektiv kommunizieren können. Hilfe zur Lösung von Problemen finden Sie AWS Direct Connect im AWS Direct Connect Benutzerhandbuch unter Problembehandlung.

`XKS_PROXY_TIMED_OUT`

–oder–

`CustomKeyStoreInvalidStateException`, `KMSInvalidStateException`, `XksProxyUriUnreachableException`
Der externe Schlüsselspeicher-Proxy hat nicht in der vorgesehenen Zeit auf die Anforderung geantwortet. Wiederholen Sie die Anforderung. Wenn Sie diesen Fehler wiederholt sehen, melden Sie ihn dem Administrator für Ihren externen Schlüsselspeicher-Proxy.

- Dieser Fehler kann darauf zurückzuführen sein, dass der externe Schlüsselmanager und der externe Schlüsselspeicher-Proxy geografisch weit voneinander entfernt sind. Bringen Sie den externen Schlüsselspeicher-Proxy nach Möglichkeit näher an den externen Schlüsselmanager.

Fehler mit der Anmeldeinformation für die Authentifizierung

Ausnahmen: `CustomKeyStoreInvalidStateException` (`CreateKey`), `KMSInvalidStateException` (kryptografische Operationen), `XksProxyIncorrectAuthenticationCredentialException` (Verwaltungsoperationen außer `CreateKey`)

Sie richten Authentifizierungsdaten für Ihren externen AWS KMS Schlüsselspeicher-Proxy ein und verwalten diese. Anschließend geben Sie die Werte AWS KMS der Anmeldeinformationen an, wenn Sie einen externen Schlüsselspeicher erstellen. Wenn Sie die Anmeldeinformation für die Authentifizierung ändern möchten, nehmen Sie diese Änderung auf Ihrem externen Schlüsselspeicher-Proxy vor. Aktualisieren Sie danach die Anmeldeinformation für Ihren externen Schlüsselspeicher. Wenn Ihr Proxy die Anmeldeinformation rotiert, müssen Sie sie für Ihren externen Schlüsselspeicher aktualisieren.

Wenn der externe Schlüsselspeicher-Proxy eine mit der Proxy-Authentifizierungsanmeldeinformation für Ihren externen Schlüsselspeicher signierte Anforderung nicht authentifiziert, hängt es von der Anforderung ab, was geschieht:

- `CreateCustomKeyStore` und `UpdateCustomKeyStore` schlagen mit einem `XksProxyIncorrectAuthenticationCredentialException` fehl.
- `ConnectCustomKeyStore` wird erfolgreich ausgeführt, aber die Verbindung schlägt fehl. Der Verbindungsstatus ist `FAILED` und der Verbindungsfehlercode lautet `INVALID_CREDENTIALS`. Details hierzu finden Sie unter Fehler bei der Verbindung mit dem externen Schlüsselspeicher.
- `KMSInvalidStateException`Bei kryptografischen Vorgängen werden alle externen Konfigurationsfehler und Verbindungsstatusfehler in einem externen Schlüsselspeicher zurückgegeben. Die zugehörige Fehlermeldung beschreibt das Problem.

Der externe Schlüsselspeicher-Proxy hat die Anforderung abgelehnt, da er AWS KMS nicht authentifizieren konnte. Überprüfen Sie die Anmeldeinformationen für Ihren externen Schlüssel speicher und aktualisieren Sie sie gegebenenfalls.

Dieser Fehler kann aus folgenden Gründen auftreten:

- Die Zugriffsschlüssel-ID oder der geheime Zugriffsschlüssel für den externen Schlüsselspeicher stimmt nicht mit den Werten überein, die auf dem externen Schlüsselspeicher-Proxy festgelegt wurden.

  Um diesen Fehler zu beheben, aktualisieren Sie die Proxy-Authentifizierungsanmeldeinformation für Ihren externen Schlüsselspeicher. Sie können diese Änderung vornehmen, ohne Ihren externen Schlüsselspeicher zu trennen.
- Ein Reverse-Proxy zwischen AWS KMS und dem externen Schlüsselspeicher-Proxy könnte HTTP-Header so manipulieren, dass die SigV4-Signaturen ungültig werden. Um diesen Fehler zu beheben, benachrichtigen Sie den Administrator für Ihren Proxy.

Fehler mit dem Schlüsselstatus

Ausnahmen: `KMSInvalidStateException`

`KMSInvalidStateException` wird für zwei verschiedene Zwecke für KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern verwendet.

- Wenn ein Verwaltungsvorgang (z. B. `CancelKeyDeletion`) fehlschlägt und diese Ausnahme zurückgibt, bedeutet dies, dass der Schlüsselstatus des KMS-Schlüssels nicht mit der Operation kompatibel ist.

- Wenn eine kryptografische Operation für einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher mit `KMSInvalidStateException` fehlschlägt, kann dies auf ein Problem mit dem Schlüsselstatus des KMS-Schlüssels hindeuten. `KMSInvalidStateException`Bei allen externen Konfigurationsfehlern und Verbindungsstatusfehlern in einem externen Schlüsselspeicher wird jedoch ein AWS KMS kryptografischer Vorgang zurückgegeben. Identifizieren Sie das Problem anhand der Fehlermeldung, die zusammen mit der Ausnahme angezeigt wird.

Informationen zum erforderlichen Schlüsselstatus für AWS KMS API-Operationen finden Sie unterWichtige Zustände von AWS KMS Schlüsseln. Um den Schlüsselstatus eines KMS-Schlüssels zu ermitteln, zeigen Sie auf der Seite Customer managed keys (Kundenverwaltete Schlüssel) das Feld Status des KMS-Schlüssels an. Oder verwenden Sie die DescribeKeyOperation und sehen Sie sich das `KeyState` Element in der Antwort an. Details hierzu finden Sie unter Schlüssel identifizieren und anzeigen.

> **Note**
>
> Der Schlüsselstatus eines KMS-Schlüssels in einem externen Schlüsselspeicher sagt nichts über den Status des zugehörigen externen Schlüssels aus. Informationen zum Status des externen Schlüssels finden Sie mithilfe Ihres externen Schlüsselmanagers und der Tools für den externen Schlüsselspeicher-Proxy.
> Die `CustomKeyStoreInvalidStateException` bezieht sich auf den Verbindungsstatus des externen Schlüsselspeichers, nicht auf den Schlüsselstatus eines KMS-Schlüssels.

Eine kryptografische Operation für einen KMS-Schlüssel in einem benutzerdefinierten Speicher schlägt möglicherweise fehl, weil der Schlüsselstatus des KMS-Schlüssels `Unavailable` oder `PendingDeletion` lautet. (Deaktivierte Schlüssel geben `DisabledException` zurück.)

- Ein KMS-Schlüssel hat nur dann einen `Disabled` Schlüsselstatus, wenn Sie den KMS-Schlüssel in der AWS KMS Konsole oder mithilfe des DisableKeyVorgangs absichtlich deaktivieren. Wenn ein KMS-Schlüssel deaktiviert ist, können Sie ihn anzeigen und verwalten, ihn jedoch nicht für kryptographische Operationen verwenden. Um dieses Problem zu beheben, aktivieren Sie den Schlüssel. Details hierzu finden Sie unter Tasten aktivieren und deaktivieren.
- Ein KMS-Schlüssel hat den Schlüsselstatus `Unavailable`, wenn der externe Schlüsselspeicher von seinem externen Schlüsselspeicher-Proxy getrennt ist. Um die Nichtverfügbarkeit eines KMS-Schlüssels zu beheben, verbinden Sie den externen Schlüsselspeicher wieder. Wenn der externe Schlüsselspeicher wieder verbunden ist, wechselt der Schlüsselstatus der KMS-Schlüssel im

externen Schlüsselspeicher automatisch wieder zum vorherigen Status, etwa zu `Enabled` oder `Disabled`.

Ein KMS-Schlüssel hat den Schlüsselstatus `PendingDeletion`, wenn seine Löschung geplant ist und er darauf wartet. Ein Schlüsselstatusfehler bei einem KMS-Schlüssel, dessen Löschung aussteht, bedeutet, dass der Schlüssel nicht gelöscht werden sollte – entweder, weil er für die Verschlüsselung verwendet wird oder weil er für die Entschlüsselung erforderlich ist. Um den KMS-Schlüssel erneut zu aktivieren, brechen Sie den geplanten Löschvorgang ab und aktivieren Sie dann den Schlüssel. Details hierzu finden Sie unter Löschen von Schlüsseln planen.

Entschlüsselungsfehler

Ausnahmen: `KMSInvalidStateException`

Wenn ein Entschlüsselungsvorgang mit einem KMS-Schlüssel in einem externen Schlüsselspeicher fehlschlägt, wird der Standard AWS KMS `KMSInvalidStateException` zurückgegeben, den kryptografische Operationen für alle externen Konfigurationsfehler und Verbindungsstatusfehler in einem externen Schlüsselspeicher verwenden. Die Fehlermeldung, die das Problem angibt.

Zum Entschlüsseln eines Geheimtexts, der mit doppelter Verschlüsselung verschlüsselt wurde, verwendet der externe Schlüsselmanager zunächst den externen Schlüssel, um die äußere Geheimtextschicht zu entschlüsseln. AWS KMS Verwendet dann das AWS KMS Schlüsselmaterial im KMS-Schlüssel, um die innere Chiffretextschicht zu entschlüsseln. Ein ungültiger oder beschädigter Geheimtext kann vom externen Schlüsselmanager oder von AWS KMS abgelehnt werden.

Die folgenden Fehlermeldungen werden zusammen mit der `KMSInvalidStateException` angezeigt, wenn die Entschlüsselung fehlschlägt. Sie deuten auf ein Problem mit dem Geheimtext oder dem optionalen Verschlüsselungskontext in der Anfrage hin.

Der externe Schlüsselspeicher-Proxy hat die Anforderung, da der angegebene Geheimtext oder zusätzliche authentifizierte Daten beschädigt sind, fehlen oder anderweitig ungültig sind.

- Wenn der externe Schlüsselspeicher-Proxy oder der externe Schlüsselmanager meldet, dass ein Chiffretext oder sein Verschlüsselungskontext ungültig ist, deutet dies in der Regel auf ein Problem mit dem Chiffretext oder dem Verschlüsselungskontext in der Anforderung hin, an die gesendet

wurde. `Decrypt` AWS KMS Bei `Decrypt` Vorgängen AWS KMS sendet der Proxy denselben Chiffretext und denselben Verschlüsselungskontext, den er in der Anfrage empfängt. `Decrypt`

Dieser Fehler kann durch ein Netzwerkproblem während der Übertragung verursacht werden, etwa durch ein „umgekipptes“ Bit. Wiederholen Sie die `Decrypt`-Anforderung. Wenn das Problem weiterhin besteht, stellen Sie sicher, dass der Geheimtext nicht geändert oder beschädigt wurde. Stellen Sie außerdem sicher, dass der Verschlüsselungskontext in der `Decrypt` Anfrage mit dem Verschlüsselungskontext in der Anfrage AWS KMS übereinstimmt, mit der die Daten verschlüsselt wurden.

Der Geheimtext, den der externe Schlüsselspeicher-Proxy zur Entschlüsselung übermittelt hat, oder der Verschlüsselungskontext ist beschädigt, fehlt oder ist anderweitig ungültig.

- Wenn der vom Proxy empfangene AWS KMS Chiffretext zurückgewiesen wird, bedeutet dies, dass der externe Schlüsselmanager oder der Proxy einen ungültigen oder beschädigten Chiffretext zurückgegeben hat. AWS KMS

  Dieser Fehler kann durch ein Netzwerkproblem während der Übertragung verursacht werden, etwa durch ein „umgekipptes“ Bit. Wiederholen Sie die `Decrypt`-Anforderung. Wenn das Problem weiterhin besteht, stellen Sie sicher, dass der externe Schlüsselmanager ordnungsgemäß funktioniert und dass der externe Schlüsselspeicher-Proxy den Chiffretext, den er vom externen Schlüsselmanager empfängt, nicht ändert, bevor er ihn zurückgibt. AWS KMS

Fehler mit externen Schlüsseln

Ein externer Schlüssel ist ein kryptografischer Schlüssel im externen Schlüsselmanager, der als externes Schlüsselmaterial für einen KMS-Schlüssel dient. AWS KMS kann nicht direkt auf den externen Schlüssel zugreifen. Es muss den externen Schlüsselmanager über den externen Schlüsselspeicher-Proxy bitten, den externen Schlüssel zum Verschlüsseln von Daten oder zum Entschlüsseln eines Geheimtextes zu verwenden.

Sie geben die ID des externen Schlüssels in seinem externen Schlüsselmanager an, wenn Sie einen KMS-Schlüssel in Ihrem externen Schlüsselspeicher erstellen. Sie können die ID des externen Schlüssels nicht mehr ändern, nachdem der KMS-Schlüssel erstellt wurde. Um Probleme mit dem KMS-Schlüssel zu vermeiden, fordert die `CreateKey`-Operation den externen Schlüsselspeicher-Proxy auf, die ID und Konfiguration des externen Schlüssels zu verifizieren. Wenn der externe

Schlüssel nicht den Anforderungen für die Verwendung mit einem KMS-Schlüssel entspricht, schlägt die `CreateKey`-Operation mit einer Ausnahme und einer Fehlermeldung fehl, die Aufschluss über das Problem gibt.

Aber auch nach der Erstellung des KMS-Schlüssels können Probleme auftreten. Wenn eine kryptografische Operation aufgrund eines Problems mit dem externen Schlüssel fehlschlägt, schlägt die Operation fehl und gibt eine `KMSInvalidStateException` mit einer Fehlermeldung zurück, die das Problem angibt.

CreateKey Fehler für den externen Schlüssel

Ausnahmen: `XksKeyAlreadyInUseException`, `XksKeyNotFoundException`, `XksKeyInvalidConfigurationException`

Der CreateKeyVorgang versucht, die ID und die Eigenschaften des externen Schlüssels zu überprüfen, den Sie im Parameter Externe Schlüssel-ID (Konsole) oder `XksKeyId` (API) angeben. Dies dient dazu, Fehler frühzeitig zu erkennen, bevor Sie versuchen, den externen Schlüssel zusammen mit dem KMS-Schlüssel zu verwenden.

Externer Schlüssel wird verwendet

Jeder KMS-Schlüssel in einem externen Schlüsselspeicher muss einen anderen externen Schlüssel verwenden. Wenn `CreateKey` erkannt wird, dass die externe Schlüssel-ID (XksKeyId) für einen KMS-Schlüssel im externen Schlüsselspeicher nicht eindeutig ist, schlägt der Vorgang mit einer fehl`XksKeyAlreadyInUseException`.

Wenn Sie mehrere IDs für denselben externen Schlüssel verwenden, `CreateKey` wird das Duplikat nicht erkannt. KMS-Schlüssel mit demselben externen Schlüssel sind jedoch nicht interoperabel, da sie unterschiedliche AWS KMS Schlüsselmaterialien und Metadaten haben.

Externer Schlüssel nicht gefunden

Wenn der Proxy für den externen Schlüsselspeicher meldet, dass er den externen Schlüssel mithilfe der externen Schlüssel-ID (XksKeyId) für den KMS-Schlüssel nicht finden kann, schlägt der `CreateKey` Vorgang fehl und es wird die folgende Fehlermeldung angezeigt. `XksKeyNotFoundException`

> Der externe Schlüsselspeicher-Proxy hat die Anforderung abgelehnt, da er den externen Schlüssel nicht finden konnte.

Dieser Fehler kann aus folgenden Gründen auftreten:

- Die ID des externen Schlüssels (`XksKeyId`) für den KMS-Schlüssel ist möglicherweise ungültig. Die ID Ihres externen Schlüssels, den der Proxy zur Identifizierung des externen Schlüssels verwendet, finden Sie in der Dokumentation zu Ihrem externen Schlüsselspeicher-Proxy oder zu Ihrem externen Schlüsselmanager.
- Möglicherweise wurde der externe Schlüssel aus Ihrem externen Schlüsselmanager gelöscht. Verwenden Sie zur Untersuchung dieses Problems die Tools Ihres externen Schlüsselmanagers. Wenn der externe Schlüssel dauerhaft gelöscht wird, verwenden Sie einen anderen externen Schlüssel zusammen mit dem KMS-Schlüssel. Eine Liste der Anforderungen für den externen Schlüssel finden Sie unter Anforderungen an einen KMS-Schlüssel in einem externen Schlüsselspeicher.

Anforderungen an externe Schlüssel nicht erfüllt

Wenn der externe Schlüsselspeicher-Proxy meldet, dass der externe Schlüssel die Anforderungen für die Verwendung mit einem KMS-Schlüssel nicht erfüllt, schlägt die `CreateKey`-Operation fehl und gibteine `XksKeyInvalidConfigurationException` mit einer der folgenden Fehlermeldungen zurück.

Die Schlüsselspezifikation des externen Schlüssels muss AES_256 sein. Die Schlüsselspezifika tion des angegebenen externen Schlüssels lautet*<key-spec>* .

- Der externe Schlüssel muss ein symmetrischer 256-Bit-Verschlüsselungsschlüssel mit der Schlüsselspezifikation AES_256 sein. Wenn es sich bei dem angegebenen externen Schlüssel um einen anderen Typ handelt, geben Sie die ID eines externen Schlüssels an, der diese Anforderung erfüllt.

Der Status des externen Schlüssels muss ENABLED (AKTIVIERT) sein. Der Status des angegebenen externen Schlüssels ist*<status>*.

- Der externe Schlüssel muss im externen Schlüsselmanager aktiviert sein. Wenn der angegebene externe Schlüssel nicht aktiviert ist, verwenden Sie die Tools Ihres externen Schlüsselmanagers, um ihn zu aktivieren, oder geben Sie einen aktivierten externen Schlüssel an.

Die Schlüsselnutzung des externen Schlüssels muss ENCRYPT (VERSCHLÜSSELN) und DECRYPT (ENTSCHLÜSSELN) beinhalten. Die Schlüsselverwendung des angegebenen externen Schlüssels ist < *key-usage* >.

- Der externe Schlüssel muss für die Verschlüsselung und Entschlüsselung im externen Schlüsselmanager konfiguriert sein. Wenn der angegebene externe Schlüssel diese Operationen nicht beinhaltet, verwenden Sie die Tools Ihres externen Schlüsselmanagers, um die Operationen zu ändern, oder geben Sie einen anderen externen Schlüssel an.

Kryptografische Operationsfehler für den externen Schlüssel

Ausnahmen: `KMSInvalidStateException`

Wenn der externe Schlüsselspeicher-Proxy den mit dem KMS-Schlüssel verknüpften externen Schlüssel nicht finden kann oder der externe Schlüssel die Anforderungen für die Verwendung mit einem KMS-Schlüssel nicht erfüllt, schlägt die kryptografische Operation fehl.

Probleme mit externen Schlüsseln, die bei einer kryptografischen Operation erkannt werden, sind schwieriger zu beheben als Probleme mit externen Schlüsseln, die vor der Erstellung des KMS-Schlüssels erkannt wurden. Sie können die ID des externen Schlüssels nicht mehr ändern, nachdem der KMS-Schlüssel erstellt wurde. Wenn mit dem KMS-Schlüssel noch keine Daten verschlüsselt wurden, können Sie ihn löschen und einen neuen KMS-Schlüssel mit einer anderen ID für den externen Schlüssel erstellen. Mit dem KMS-Schlüssel generierter Chiffretext kann jedoch nicht mit einem anderen KMS-Schlüssel entschlüsselt werden, auch nicht mit einem mit demselben externen Schlüssel, da Schlüssel unterschiedliche Schlüsselmetadaten und anderes Schlüsselmaterial haben. AWS KMS Verwenden Sie stattdessen nach Möglichkeit die Tools Ihres externen Schlüsselmanagers, um das Problem mit dem externen Schlüssel zu beheben.

Wenn der externe Schlüsselspeicher-Proxy ein Problem mit dem externen Schlüssel meldet, geben kryptografische Operationen eine `KMSInvalidStateException` mit einer Fehlermeldung zurück, die das Problem identifiziert.

Externer Schlüssel nicht gefunden

Wenn der externe Schlüsselspeicher-Proxy meldet, dass er den externen Schlüssel mithilfe der externen Schlüssel-ID (XksKeyId) für den KMS-Schlüssel nicht finden kann, geben kryptografische Operationen a `KMSInvalidStateException` mit der folgenden Fehlermeldung zurück.

Der externe Schlüsselspeicher-Proxy hat die Anforderung abgelehnt, da er den externen Schlüssel nicht finden konnte.

Dieser Fehler kann aus folgenden Gründen auftreten:

- Die ID des externen Schlüssels (`XksKeyId`) für den KMS-Schlüssel ist nicht mehr gültig.

  Die ID des externen Schlüssels, der Ihrem KMS-Schlüssel zugeordnet ist, finden Sie in den Details des KMS-Schlüssels. Die ID, die Ihr externer Schlüssel-Proxy zur Identifizierung des externen Schlüssels verwendet, finden Sie in der Dokumentation zu Ihrem externen Schlüsselspeicher-Proxy oder zu Ihrem externen Schlüsselmanager.

  AWS KMS überprüft die externe Schlüssel-ID, wenn ein KMS-Schlüssel in einem externen Schlüsselspeicher erstellt wird. Die ID kann jedoch ungültig werden, insbesondere wenn der Wert der ID des externen Schlüssels ein Alias oder ein veränderbarer Name ist. Sie können die einem vorhandenen KMS-Schlüssel zugeordnete ID des externen Schlüssels nicht ändern. Um einen Geheimtext zu entschlüsseln, der unter dem KMS-Schlüssel verschlüsselt wurde, müssen Sie den externen Schlüssel erneut der vorhandenen ID des externen Schlüssels zuordnen.

  Wenn Sie den KMS-Schlüssel noch nicht zum Verschlüsseln von Daten verwendet haben, können Sie einen neuen KMS-Schlüssel mit einer gültigen ID des externen Schlüssels erstellen. Haben Sie jedoch mit dem KMS-Schlüssel Geheimtext generiert, können Sie diesen Geheimtext mit keinem anderen KMS-Schlüssel entschlüsseln, selbst wenn derselbe externe Schlüssel verwendet wird.
- Möglicherweise wurde der externe Schlüssel aus Ihrem externen Schlüsselmanager gelöscht. Verwenden Sie zur Untersuchung dieses Problems die Tools Ihres externen Schlüsselmanagers. Versuchen Sie nach Möglichkeit, das Schlüsselmaterial aus einer Kopie oder einem Backup Ihres externen Schlüsselmanagers wiederherzustellen. Wenn der externe Schlüssel dauerhaft gelöscht wird, kann der unter dem zugehörigen KMS-Schlüssel verschlüsselte Geheimtext nicht wiederhergestellt werden.

Fehler bei der Konfiguration externer Schlüssel

Wenn der externe Schlüsselspeicher-Proxy meldet, dass der externe Schlüssel die Anforderungen für die Verwendung mit einem KMS-Schlüssel nicht erfüllt, gibt die kryptografische Operation eine `KMSInvalidStateException` mit einer der folgenden Fehlermeldungen zurück.

Der externe Schlüsselspeicher-Proxy lehnte die Anforderung ab, da der externe Schlüssel die angeforderte Operation nicht unterstützt.

- Der externe Schlüssel muss sowohl die Verschlüsselung als auch die Entschlüsselung unterstützen. Wenn die Schlüsselnutzung keine Verschlüsselung und Entschlüsselung beinhaltet, ändern Sie die Schlüsselnutzung mithilfe der Tools Ihres externen Schlüsselmanagers.

Der externe Schlüsselspeicher-Proxy hat die Anforderung abgelehnt, da der externe Schlüssel im externen Schlüsselmanager nicht aktiviert ist.

- Der externe Schlüssel muss im externen Schlüsselmanager aktiviert und für die Verwendung verfügbar sein. Wenn der Status des externen Schlüssels nicht `Enabled` lautet, aktivieren Sie ihn mithilfe der Tools Ihres externen Schlüsselmanagers.

Proxy-Probleme

Ausnahmen:

`CustomKeyStoreInvalidStateException` (`CreateKey`), `KMSInvalidStateException` (kryptografische Operationen), `UnsupportedOperationException`, `XksProxyUriUnreachableException`, `XksProxyInvalidResponseException` (Verwaltungsoperationen außer `CreateKey`)

Der externe Schlüsselspeicher-Proxy vermittelt die gesamte Kommunikation zwischen AWS KMS und dem externen Schlüsselmanager. Er übersetzt generische AWS KMS Anfragen in ein Format, das Ihr externer Schlüsselmanager verstehen kann. Wenn der externe Schlüsselspeicher-Proxy nicht der API-Spezifikation für den AWS KMS externen Schlüsselspeicher-Proxy entspricht oder wenn er nicht ordnungsgemäß funktioniert oder nicht mit ihm kommunizieren kann AWS KMS, können Sie in Ihrem externen Schlüsselspeicher keine KMS-Schlüssel erstellen oder verwenden.

In vielen Fehlermeldungen wird der externe Schlüsselspeicher-Proxy erwähnt, weil er eine wichtige Rolle in der Architektur des externen Schlüsselspeichers spielt. Diese Probleme können ihren Ursprung aber im externen Schlüsselmanager oder im externen Schlüssel haben.

Die Probleme in diesem Abschnitt beziehen sich auf Probleme beim Design oder Betrieb des externen Schlüsselspeicher-Proxys. Zum Beheben dieser Probleme ist möglicherweise eine Änderung an der Proxysoftware erforderlich. Wenden Sie sich an den Administrator für Ihren Proxy. Um Sie bei der Diagnose von Proxy-Problemen zu unterstützen, stellt AWS KMS einen Open-Source-Testclient bereit (XKS-Proxy-Textclient), der überprüft, ob Ihr externer Schlüsselspeicher-Proxy der API-Spezifikation von AWS KMS für externe Schlüsselspeicher-Proxys entspricht.

`CustomKeyStoreInvalidStateException`, `KMSInvalidStateException` oder `XksProxyUriUnreachableException`
Der externe Schlüsselspeicher-Proxy hat einen fehlerhaften Status. Wenn Sie diese Meldung wiederholt sehen, benachrichtigen Sie den Administrator für Ihren externen Schlüsselspeicher-Proxy.

- Dieser Fehler kann auf ein Betriebsproblem oder einen Softwarefehler im externen Schlüsselspeicher-Proxy hindeuten. Sie können CloudTrail Protokolleinträge für den AWS KMS API-Vorgang finden, der die einzelnen Fehler generiert hat. Dieser Fehler kann möglicherweise durch das erneute Ausführen der Operation behoben werden. Sollte er jedoch weiterhin bestehen, benachrichtigen Sie den Administrator für Ihren externen Schlüsselspeicher-Proxy.
- Wenn der externe Schlüsselspeicher-Proxy (in einer GetHealthStatusAntwort) meldet, dass alle externen Schlüsselmanager-Instanzen vorhanden sind`UNAVAILABLE`, schlagen Versuche, einen externen Schlüsselspeicher zu erstellen oder zu aktualisieren, mit dieser Ausnahme fehl. Wenn dieser Fehler weiterhin besteht, lesen Sie in der Dokumentation Ihres externen Schlüsselmanagers nach.

`CustomKeyStoreInvalidStateException`, `KMSInvalidStateException` oder `XksProxyInvalidResponseException`
AWS KMS kann die Antwort des externen Schlüsselspeicher-Proxys nicht interpretieren. Wenn Sie diesen Fehler wiederholt sehen, ziehen Sie den Administrator für Ihren externen Schlüssel speicher-Proxy zurate.

- AWS KMS Operationen erzeugen diese Ausnahme, wenn der Proxy eine undefinierte Antwort zurückgibt, die AWS KMS nicht analysiert oder interpretiert werden kann. Dieser Fehler kann gelegentlich aufgrund vorübergehender externer Probleme oder sporadischer Netzwerkfehler auftreten. Wenn das Problem jedoch weiterhin besteht, kann dies darauf hindeuten, dass der

externe Schlüsselspeicher-Proxy nicht der API-Spezifikation von AWS KMS für den externen Schlüsselspeicher-Proxy entspricht. Informieren Sie den Administrator oder Anbieter Ihres externen Schlüsselspeichers.

`CustomKeyStoreInvalidStateException`, `KMSInvalidStateException` oder `UnsupportedOperationException`

Der externe Schlüsselspeicher-Proxy lehnte die Anforderung ab, da er die angeforderte kryptografische Operation nicht unterstützt.

- Der externe Schlüsselspeicher-Proxy sollte alle Proxys unterstützen, die in der AWS KMS External Key Store Proxy API-Spezifikation APIs definiert sind. Dieser Fehler deutet darauf hin, dass der Proxy die Operation, die mit der Anforderung zusammenhängt, nicht unterstützt. Informieren Sie den Administrator oder Anbieter Ihres externen Schlüsselspeichers.

Probleme mit der Proxy-Autorisierung

Ausnahmen: `CustomKeyStoreInvalidStateException`, `KMSInvalidStateException`

Einige externe Schlüsselspeicher-Proxys implementieren Autorisierungsanforderungen für die Verwendung ihrer externen Schlüssel. Ein externer Schlüsselspeicher-Proxy ist berechtigt, aber nicht verpflichtet, ein Autorisierungsschema zu entwerfen und zu implementieren, das es bestimmten Benutzern erlaubt, bestimmte Operationen unter bestimmten Bedingungen anzufordern. Beispielsweise könnte ein Proxy zulassen, dass ein:e Benutzer:in die Verschlüsselung mit einem bestimmten externen Schlüssel durchführt, aber nicht die Entschlüsselung mit diesem Schlüssel. Weitere Informationen finden Sie unter Proxy-Autorisierung für externen Schlüsselspeicher (optional).

Die Proxyautorisierung basiert auf Metadaten, die in den Anfragen an den Proxy AWS KMS enthalten sind. Die Felder `awsSourceVpc` und `awsSourceVpce` sind nur dann in den Metadaten enthalten, wenn die Anforderung von einem VPC-Endpunkt stammt – und auch nur, wenn der Aufrufer im selben Konto wie der KMS-Schlüssel ist.

```
"requestMetadata": {
    "awsPrincipalArn": string,
    "awsSourceVpc": string, // optional
    "awsSourceVpce": string, // optional
    "kmsKeyArn": string,
```

```
    "kmsOperation": string,
    "kmsRequestId": string,
    "kmsViaService": string // optional
}
```

Wenn der Proxy eine Anfrage aufgrund eines Autorisierungsfehlers ablehnt, schlägt der entsprechende AWS KMS Vorgang fehl. `CreateKey`kehrt zurück`CustomKeyStoreInvalidStateException`. AWS KMS kryptografische Operationen kehren zurück`KMSInvalidStateException`. Beide verwenden die folgende Fehlermeldung:

> Der externe Schlüsselspeicher-Proxy hat den Zugriff auf die Operation verweigert. Stellen Sie sicher, dass sowohl der:die Benutzer:in als auch der externe Schlüssel für diese Operation autorisiert ist, und führen Sie die Anforderung erneut aus.

- Um den Fehler zu beheben, ermitteln Sie mithilfe Ihres externen Schlüsselmanagers oder mithilfe der Tools für Ihren externen Schlüsselspeicher-Proxy, warum die Autorisierung fehlgeschlagen ist. Aktualisieren Sie anschließend das Verfahren, das zur unbefugten Anforderung geführt hat, oder verwenden Sie die Tools für Ihren externen Schlüsselspeicher-Proxy, um die Autorisierungsrichtlinie zu aktualisieren. Dieser Fehler lässt sich in AWS KMS nicht beheben.

Sicherheit von AWS Key Management Service

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einem Rechenzentrum und einer Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der übergreifenden Verantwortlichkeit beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für AWS Key Management Service (AWS KMS) gelten, finden Sie unter AWS Services im Bereich nach Compliance-Programm AWS .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Zusätzlich zu Ihrer Konfiguration und Nutzung von sind Sie für andere Faktoren verantwortlich AWS KMS keys, darunter die Sensibilität Ihrer Daten, die Anforderungen Ihres Unternehmens und die geltenden Gesetze und Vorschriften AWS KMS

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung anwenden können AWS Key Management Service. Es zeigt Ihnen, wie Sie die Konfiguration vornehmen AWS KMS , um Ihre Sicherheits- und Compliance-Ziele zu erreichen.

Themen

- Datenschutz in AWS Key Management Service
- Identitäts- und Zugriffsmanagement für AWS Key Management Service
- Einloggen und Überwachen AWS Key Management Service
- Konformitätsvalidierung für AWS Key Management Service
- Resilienz in AWS Key Management Service
- Sicherheit der Infrastruktur in AWS Key Management Service

Datenschutz in AWS Key Management Service

AWS Key Management Service speichert und schützt Ihre Verschlüsselungsschlüssel, um sie hochverfügbar zu machen und Ihnen gleichzeitig eine starke und flexible Zugriffskontrolle zu bieten.

Themen

- Schutz von Schlüsselmaterial
- Datenverschlüsselung
- Richtlinie für den Datenverkehr zwischen Netzwerken

Schutz von Schlüsselmaterial

AWS KMS Generiert und schützt standardmäßig das kryptografische Schlüsselmaterial für KMS-Schlüssel. Darüber hinaus AWS KMS bietet es Optionen für Schlüsselmaterial, das außerhalb von AWS KMS erstellt und geschützt wird.

Schutz von Schlüsselmaterial, das in erzeugt wurde AWS KMS

Wenn Sie einen KMS-Schlüssel erstellen, wird standardmäßig das kryptografische Material für den KMS-Schlüssel AWS KMS generiert und geschützt.

Um das Schlüsselmaterial für KMS-Schlüssel zu schützen, AWS KMS nutzt es eine verteilte Flotte von FIPS 140-3 Security Level 3-validierten Hardware-Sicherheitsmodulen (). HSMs Jedes AWS KMS HSM ist eine dedizierte, eigenständige Hardware-Appliance, die spezielle kryptografische Funktionen bereitstellt, um die Sicherheits- und Skalierbarkeitsanforderungen von zu erfüllen. AWS KMS(Diejenigen HSMs , die in China Regionen AWS KMS verwendet werden, sind von OSCCA zertifiziert und entsprechen allen einschlägigen chinesischen Vorschriften, sind jedoch nicht im Rahmen des FIPS 140-3 Cryptographic Module Validation Program validiert.)

Das Schlüsselmaterial für einen KMS-Schlüssel wird standardmäßig verschlüsselt, wenn er im HSM generiert wird. Das Schlüsselmaterial wird nur innerhalb des flüchtigen HSM-Speichers und nur für die wenigen Millisekunden entschlüsselt, die für die Verwendung in einer kryptografischen Operation benötigt werden. Wenn das Schlüsselmaterial nicht aktiv verwendet wird, wird es innerhalb des HSM verschlüsselt und in einen äußerst dauerhaften (99,999999999%) persistenten Speicher mit niedriger Latenz übertragen, wo es getrennt und isoliert von der HSMs Das Klartext-Schlüsselmaterial verlässt dieHSM-Sicherheitsgrenze nie; es wird nie auf die Festplatte geschrieben oder in einem Speichermedium gespeichert. (Die einzige Ausnahme ist der öffentliche Schlüssel eines asymmetrischen Schlüsselpaares, der nicht geheim ist).

AWS behauptet als grundlegendes Sicherheitsprinzip, dass keinerlei menschliche Interaktion mit kryptografischem Schlüsselmaterial im Klartext stattfindet. AWS-Service Es gibt keinen Mechanismus, mit dem irgendjemand, einschließlich AWS-Service Betreiber, Schlüsselmaterial im Klartext einsehen, darauf zugreifen oder es exportieren könnte. Dieses Prinzip gilt auch bei katastrophalen Ausfällen und Notfallwiederherstellungsereignissen. Das Kundenschlüsselmaterial in Klartext AWS KMS wird für kryptografische Operationen innerhalb von AWS KMS FIPS 140-3 verwendet und HSMs nur als Antwort auf autorisierte Anfragen des Kunden oder seines Beauftragten an den Service validiert.

Bei vom Kunden verwalteten Schlüsseln ist derjenige, der den Schlüssel erstellt AWS-Konto , der alleinige und nicht übertragbare Eigentümer des Schlüssels. Das Eigentümerkonto hat die vollständige und ausschließliche Kontrolle über die Autorisierungsrichtlinien, die den Zugriff auf den Schlüssel regeln. Denn Von AWS verwaltete Schlüssel er AWS-Konto hat die vollständige Kontrolle über die IAM-Richtlinien, die Anfragen an die autorisieren. AWS-Service

Schutz von außerhalb von AWS KMS generiertem Schlüsselmaterial

AWS KMS bietet Alternativen zu wichtigen Materialien, die in generiert wurden. AWS KMS

Mit benutzerdefinierten Schlüsselspeichern, einer optionalen AWS KMS Funktion, können Sie KMS-Schlüssel erstellen, die auf Schlüsselmaterial basieren, das außerhalb von generiert und verwendet wird AWS KMS. KMS-Schlüssel in AWS CloudHSM Schlüsselspeichern werden durch Schlüssel in AWS CloudHSM Hardware-Sicherheitsmodulen gesichert, die Sie kontrollieren. Diese HSMs sind nach FIPS 140-2 Security Level 3 oder 140-3 Security Level 3 zertifiziert. KMS-Schlüssel in externen Schlüsselspeichern werden durch Schlüssel in einem externen Schlüsselmanager gesichert, den Sie außerhalb steuern und verwalten AWS, z. B. ein physisches HSM in Ihrem privaten Rechenzentrum.

Ein weiteres optionales Feature ermöglicht das Importieren des Schlüsselmaterials für einen KMS-Schlüssel. Um importiertes Schlüsselmaterial während der Übertragung zu schützen AWS KMS, verschlüsseln Sie das Schlüsselmaterial mit einem öffentlichen Schlüssel aus einem in einem AWS KMS HSM generierten RSA-Schlüsselpaar. Das importierte Schlüsselmaterial wird in einem HSM entschlüsselt und im AWS KMS HSM unter einem symmetrischen Schlüssel erneut verschlüsselt. Wie jedes AWS KMS Schlüsselmaterial verlässt importiertes Schlüsselmaterial im Klartext niemals unverschlüsselt das System. HSMs Der Kunde, der das Schlüsselmaterial zur Verfügung gestellt hat, ist jedoch verantwortlich für die sichere Verwendung, Haltbarkeit und Wartung des Schlüsselmaterials außerhalb von AWS KMS.

Datenverschlüsselung

Die AWS KMS darin enthaltenen Daten AWS KMS keys und das Verschlüsselungsschlüsselmaterial, das sie repräsentieren. Dieses Schlüsselmaterial ist im Klartext nur in AWS KMS Hardware-Sicherheitsmodulen (HSMs) und nur dann vorhanden, wenn es verwendet wird. Andernfalls wird das Schlüsselmaterial verschlüsselt und in dauerhaftem persistenten Speicher aufbewahrt.

Das Schlüsselmaterial, das für KMS-Schlüssel AWS KMS generiert wird, überschreitet niemals die Grenze des AWS KMS HSMs unverschlüsselten Bereichs. Es wird bei keinen AWS KMS API-Vorgängen exportiert oder übertragen. Eine Ausnahme bilden Schlüssel mit mehreren Regionen, bei denen ein regionsübergreifender Replikationsmechanismus AWS KMS verwendet wird, um das Schlüsselmaterial für einen Schlüssel mit mehreren Regionen von einem HSM in einem in ein HSM in einem anderen AWS-Region zu kopieren. AWS-Region Einzelheiten finden Sie unter Replikationsprozess für Schlüssel mit mehreren Regionen im Abschnitt Kryptografische Details. AWS Key Management Service

Themen

- Verschlüsselung im Ruhezustand
- Verschlüsselung während der Übertragung

Verschlüsselung im Ruhezustand

AWS KMS generiert Schlüsselmaterial für AWS KMS keys FIPS 140-3-Sicherheitsmodule, die der Sicherheitsstufe 3 entsprechen (). HSMs Die einzige Ausnahme bilden China Regionen, in denen HSMs die zur Generierung von KMS-Schlüsseln AWS KMS verwendeten Dateien allen einschlägigen chinesischen Vorschriften entsprechen, aber nicht im Rahmen des FIPS 140-3-Validierungsprogramms für kryptografische Module validiert sind. Bei Nichtgebrauch wird Schlüsselmaterial durch einen HSM-Schlüssel verschlüsselt und in dauerhaftem persistenten Speicher geschrieben. Das Schlüsselmaterial für KMS-Schlüssel und die Verschlüsselungsschlüssel, die das Schlüsselmaterial schützen, verlassen diese niemals in Klartextform. HSMs

Die Verschlüsselung und Verwaltung von Schlüsselmaterial für KMS-Schlüssel erfolgt vollständig durch AWS KMS.

Weitere Informationen finden Sie unter Arbeiten mit AWS KMS keys unter AWS Key Management Service Kryptografische Details.

Verschlüsselung während der Übertragung

Schlüsselmaterial, das für KMS-Schlüssel AWS KMS generiert wird, wird bei AWS KMS API-Vorgängen niemals exportiert oder übertragen. AWS KMS verwendet Schlüsselkennungen, um die KMS-Schlüssel bei API-Vorgängen darzustellen. In ähnlicher Weise ist Schlüsselmaterial für KMS-Schlüssel in AWS KMS benutzerdefinierten Schlüsselspeichern nicht exportierbar und wird auch nicht im Rahmen von AWS KMS API-Vorgängen AWS CloudHSM übertragen.

Einige AWS KMS API-Operationen geben jedoch Datenschlüssel zurück. Kunden können außerdem API-Operationen zum Importieren von Schlüsselmaterial für ausgewählte KMS-Schlüssel verwenden.

Alle AWS KMS API-Aufrufe müssen mit Transport Layer Security (TLS) signiert und übertragen werden. AWS KMS erfordert TLS 1.2 und empfiehlt TLS 1.3 in allen Regionen. AWS KMS unterstützt auch hybrides Post-Quantum-TLS für AWS KMS Service-Endpunkte in allen Regionen außer den Regionen China. AWS KMS unterstützt kein hybrides Post-Quantum-TLS für FIPS-Endpunkte in. AWS GovCloud (US) Aufrufe an AWS KMS erfordern auch eine moderne Verschlüsselungssammlung, die Perfect Forward Secrecy unterstützt, was bedeutet, dass die Kompromittierung eines Geheimnisses, wie z. B. eines privaten Schlüssels, nicht auch den Sitzungsschlüssel gefährdet.

Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Um AWS KMS Standardendpunkte oder AWS KMS FIPS-Endpunkte verwenden zu können, müssen Clients TLS 1.2 oder höher unterstützen. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3. Eine Liste der AWS KMS FIPS-Endpunkte finden Sie unter AWS Key Management Service Endpunkte und Kontingente im. Allgemeine AWS-Referenz

HSMs Die Kommunikation zwischen AWS KMS Service-Hosts und wird mithilfe von Elliptic Curve Cryptography (ECC) und Advanced Encryption Standard (AES) in einem authentifizierten Verschlüsselungsschema geschützt. Weitere Informationen finden Sie unter Interne Kommunikationssicherheit unter Kryptografische Details. AWS Key Management Service

Richtlinie für den Datenverkehr zwischen Netzwerken

AWS KMS unterstützt eine AWS Management Console und eine Reihe von API-Vorgängen, mit denen Sie sie erstellen und verwalten AWS KMS keys und in kryptografischen Vorgängen verwenden können.

AWS KMS unterstützt zwei Netzwerkverbindungsoptionen von Ihrem privaten Netzwerk bis AWS.

- Eine IPSec VPN-Verbindung über das Internet
- AWS Direct Connect, das Ihr internes Netzwerk über ein Standard-Ethernet-Glasfaserkabel mit einem AWS Direct Connect Standort verbindet.

Alle AWS KMS API-Aufrufe müssen signiert und mithilfe von Transport Layer Security (TLS) übertragen werden. Die Anrufe erfordern auch eine moderne Verschlüsselungssammlung, die Perfect Forward Secrecy unterstützt. Der Datenverkehr zu den Hardware-Sicherheitsmodulen (HSMs), die Schlüsselmaterial für KMS-Schlüssel speichern, ist nur von bekannten AWS KMS API-Hosts über das AWS interne Netzwerk zulässig.

Um eine direkte Verbindung zu AWS KMS Ihrer Virtual Private Cloud (VPC) herzustellen, ohne Datenverkehr über das öffentliche Internet zu senden, verwenden Sie VPC-Endpunkte, betrieben von. AWS PrivateLink Weitere Informationen finden Sie unter Stellen Sie eine Connect AWS KMS über einen VPC-Endpunkt her.

AWS KMS unterstützt auch eine hybride Option für den Schlüsselaustausch nach dem Quantenzugriff für das Netzwerkverschlüsselungsprotokoll Transport Layer Security (TLS). Sie können diese Option mit TLS verwenden, wenn Sie eine Verbindung zu AWS KMS API-Endpunkten herstellen.

Identitäts- und Zugriffsmanagement für AWS Key Management Service

AWS Identity and Access Management (IAM) hilft Ihnen, den Zugriff auf AWS Ressourcen sicher zu kontrollieren. Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu nutzen AWS KMS . Weitere Informationen finden Sie unter Verwenden von IAM-Richtlinien mit AWS KMS.

Schlüsselrichtlinien sind der wichtigste Mechanismus zur Steuerung des Zugriffs auf KMS-Schlüssel in AWS KMS. Jeder KMS-Schlüssel muss über eine Schlüsselrichtlinie verfügen. Sie können auch IAM-Richtlinien und Erteilungen, zusammen mit Schlüsselrichtlinien, zum Steuern des Zugriffs auf Ihre KMS-Schlüssel verwenden. Weitere Informationen finden Sie unter KMS-Schlüsselzugriff und Berechtigungen.

Wenn Sie eine Amazon Virtual Private Cloud (Amazon VPC) verwenden, können Sie eine VPC-Schnittstelle für AWS KMS Powered by erstellen. AWS PrivateLink Sie können auch VPC-Endpunktrichtlinien verwenden, um zu bestimmen, welche Principals auf Ihren AWS KMS Endpunkt

zugreifen können, welche API-Aufrufe sie tätigen können und auf welchen KMS-Schlüssel sie zugreifen können.

Themen

- AWS verwaltete Richtlinien für AWS Key Management Service
- Verwenden von serviceverknüpften Rollen für AWS KMS

AWS verwaltete Richtlinien für AWS Key Management Service

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. AWS Verwaltete Richtlinien dienen dazu, Berechtigungen für viele gängige Anwendungsfälle bereitzustellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinie: AWSKey ManagementServicePowerUser

Sie können die `AWSKeyManagementServicePowerUser`-Richtlinie an Ihre IAM-Identitäten anfügen.

Sie können eine von `AWSKeyManagementServicePowerUser` verwaltete Richtlinie verwenden, um IAM-Prinzipalen in Ihrem Konto die Berechtigungen eines Hauptbenutzers zu gewähren. Hauptbenutzer können KMS-Schlüssel erstellen, die von ihnen erstellten KMS-Schlüssel verwenden und verwalten sowie alle KMS-Schlüssel und IAM-Identitäten anzeigen. Benutzer mit der von `AWSKeyManagementServicePowerUser` verwalteten Richtlinie können auch Berechtigungen aus anderen Quellen erhalten, darunter Schlüsselrichtlinien, andere IAM-Richtlinien und Erteilungen.

`AWSKeyManagementServicePowerUser`ist eine AWS verwaltete IAM-Richtlinie. Weitere Informationen zu AWS verwalteten Richtlinien finden Sie unter AWS Verwaltete Richtlinien im IAM-Benutzerhandbuch.

> **Note**
>
> Berechtigungen in dieser Richtlinie, die für einen KMS-Schlüssel spezifisch sind, wie z. B. `kms:TagResource` und `kms:GetKeyRotationStatus`, sind nur dann wirksam, wenn die Schlüsselrichtlinie für diesen KMS-Schlüssel dem AWS-Konto ausdrücklich erlaubt, IAM-Richtlinie zur Steuerung des Zugriffs auf den Schlüssel zu verwenden. Um festzustellen, ob sich eine Berechtigung auf einen KMS-Schlüssel bezieht, siehe AWS KMS Berechtigungen und suchen Sie in der Spalte Ressourcen nach dem Wert KMS-Schlüssel.
> Mit dieser Richtlinie erhält ein Hauptbenutzer Berechtigungen für jeden KMS-Schlüssel mit einer Schlüsselrichtlinie, die den Vorgang zulässt. Für kontoübergreifende Berechtigungen wie `kms:DescribeKey` und `kms:ListGrants` kann dies KMS-Schlüssel in nicht vertrauenswürdigen AWS-Konten einschließen. Details dazu finden Sie unter Bewährte Methoden für IAM-Richtlinien und Benutzern in anderen Konten die Verwendung des KMS-Schlüssels erlauben. Um festzustellen, ob eine Berechtigung für KMS-Schlüssel in anderen Konten gültig ist, siehe AWS KMS Berechtigungen und suchen Sie in der Spalte Kontoübergreifende Verwendung nach dem Wert Ja.
> Damit Principals die AWS KMS Konsole fehlerfrei aufrufen können, benötigt der Principal das Tag: GetResources permission, das nicht in der `AWSKeyManagementServicePowerUser` Richtlinie enthalten ist. Sie können diese Berechtigung in einer separaten IAM-Richtlinie erlauben.

Die von AWSKeyManagementServicePowerUser verwaltete IAM-Richtlinie muss die folgende Berechtigungen umfassen.

- Erlaubt es Prinzipalen, KMS-Schlüssel zu erstellen. Da dieser Prozess das Festlegen der Schlüsselrichtlinie beinhaltet, können Hauptbenutzer sich selbst und anderen die Berechtigung zur Verwendung und Verwaltung der von ihnen erstellten KMS-Schlüssel erteilen.
- Erlaubt es Benutzern, Aliase und Tags auf allen KMS-Schlüsseln zu erstellen und zu löschen. Wenn Sie ein Tag oder einen Alias ändern, wird die Berechtigung zur Verwendung und Verwaltung des KMS-Schlüssels erteilt oder verweigert. Details hierzu finden Sie unter ABAC für AWS KMS.

- Erlaubt es Benutzern, detaillierte Informationen zu allen KMS-Schlüsseln zu erhalten, einschließlich ihres Schlüssel-ARN, der kryptografischen Konfiguration, der Schlüsselrichtlinie, Aliase, Tags und Drehungsstatus.
- Erlaubt es Benutzern, IAM-Benutzer, -Gruppen und -Rollen aufzulisten.
- Diese Richtlinie erlaubt es Prinzipalen nicht, KMS-Schlüssel zu verwenden oder zu verwalten, die sie nicht erstellt haben. Sie können jedoch Aliase und Tags auf allen KMS-Schlüsseln ändern, was ihnen die Berechtigung zur Verwendung oder Verwaltung eines KMS-Schlüssels erlauben oder verweigern kann.

Informationen zu den Berechtigungen für diese Richtlinie finden Sie AWSKeyManagementServicePowerUserin der Referenz zu AWS verwalteten Richtlinien.

AWS verwaltete Richtlinie: AWSService RoleForKeyManagementServiceCustomKeyStores

Sie können `AWSServiceRoleForKeyManagementServiceCustomKeyStores` nicht an Ihre IAM-Entitäten anhängen. Diese Richtlinie ist mit einer dienstbezogenen Rolle verknüpft, mit der Sie die AWS KMS Erlaubnis erhalten, die mit Ihrem AWS CloudHSM Schlüsselspeicher verknüpften AWS CloudHSM Cluster anzuzeigen und das Netzwerk so einzurichten, dass eine Verbindung zwischen Ihrem benutzerdefinierten Schlüsselspeicher und seinem AWS CloudHSM Cluster unterstützt wird. Weitere Informationen finden Sie unter Autorisierung AWS KMS zur Verwaltung AWS CloudHSM und Amazon-Ressourcen EC2 .

AWS verwaltete Richtlinie: AWSService RoleForKeyManagementServiceMultiRegionKeys

Sie können `AWSServiceRoleForKeyManagementServiceMultiRegionKeys` nicht an Ihre IAM-Entitäten anhängen. Diese Richtlinie ist einer dienstbezogenen Rolle zugeordnet, die die AWS KMS Erlaubnis erteilt, alle Änderungen am Schlüsselmaterial eines Primärschlüssels mit mehreren Regionen mit seinen Replikatschlüsseln zu synchronisieren. Weitere Informationen finden Sie unter Autorisierung zur Synchronisation von AWS KMS Schlüsseln aus mehreren Regionen.

AWS KMS Aktualisierungen der verwalteten Richtlinien AWS

Hier finden Sie Informationen zu Aktualisierungen AWS verwalteter Richtlinien AWS KMS seit Beginn der Nachverfolgung dieser Änderungen durch diesen Dienst. Um automatische Warnungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der AWS KMS Dokumentverlauf-Seite.

Änderung	Beschreibung	Datum
AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy – Aktualisierung auf eine bestehende Richtlinie	AWS KMS In der Richtlinienversion v2 wurde der verwalteten Richtlinie ein Feld mit der Anweisungs-ID (`Sid`) hinzugefügt.	21. November 2024
AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy – Aktualisierung auf eine bestehende Richtlinie	AWS KMS hat die `ec2:DescribeNetworkInterfaces` Berechtigungen `ec2:DescribeVpcs` `ec2:DescribeNetworkAcls` , und hinzugefügt, um Änderungen in der VPC zu überwachen, die Ihren AWS CloudHSM Cluster enthält, sodass bei Ausfällen klare Fehlermeldungen ausgegeben werden AWS KMS können.	10. November 2023
AWS KMS hat begonnen, Änderungen zu verfolgen	AWS KMS hat begonnen, Änderungen für die AWS verwalteten Richtlinien zu verfolgen.	10. November 2023

Verwenden von serviceverknüpften Rollen für AWS KMS

AWS Key Management Service verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist ein einzigartiger Typ von IAM-Rolle, mit der direkt verknüpft ist. AWS KMS Servicebezogene Rollen werden durch alle Berechtigungen definiert AWS KMS und beinhalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

Eine dienstbezogene Rolle AWS KMS erleichtert die Einrichtung, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. AWS KMS definiert die Berechtigungen

ihrer dienstbezogenen Rollen und AWS KMS kann, sofern nicht anders definiert, nur ihre Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können eine serviceverknüpfte Rolle erst löschen, nachdem die zugehörigen Ressourcen gelöscht wurden. Dadurch werden Ihre AWS KMS Ressourcen geschützt, da Sie nicht versehentlich die Zugriffsberechtigung für die Ressourcen entziehen können.

Informationen zu anderen Services, die serviceverknüpften Rollen unterstützen, finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie nach den Services, für die Ja in der Spalte Serviceverknüpfte Rolle angegeben ist. Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Einzelheiten zu den Aktualisierungen der in diesem Thema erörterten dienstbezogenen Rollen finden Sie unter. AWS KMS Aktualisierungen der verwalteten Richtlinien AWS

Themen

- Autorisierung AWS KMS zur Verwaltung AWS CloudHSM und Amazon-Ressourcen EC2
- Autorisierung zur Synchronisation von AWS KMS Schlüsseln aus mehreren Regionen

Autorisierung AWS KMS zur Verwaltung AWS CloudHSM und Amazon-Ressourcen EC2

Um Ihre AWS CloudHSM wichtigsten Speicher zu unterstützen, ist eine Genehmigung AWS KMS erforderlich, um Informationen über Ihre AWS CloudHSM Cluster abzurufen. Außerdem benötigt es die Erlaubnis, die Netzwerkinfrastruktur zu erstellen, die Ihren AWS CloudHSM Schlüsselspeicher mit seinem AWS CloudHSM Cluster verbindet. Um diese Berechtigungen zu erhalten, AWS KMS erstellt die AWSServiceRoleForKeyManagementServiceCustomKeyStoresdienstbezogene Rolle in Ihrem AWS-Konto. Benutzer, die AWS CloudHSM Schlüsselspeicher erstellen, benötigen die `iam:CreateServiceLinkedRole` Erlaubnis, dienstbezogene Rollen zu erstellen.

Einzelheiten zu Aktualisierungen der AWSKeyManagementServiceCustomKeyStoresServiceRolePolicyverwalteten Richtlinie finden Sie unterAWS KMS Aktualisierungen der verwalteten Richtlinien AWS.

Themen

- Informationen zur AWS KMS serviceverknüpften Rolle
- Erstellen der serviceverknüpften Rolle

- Bearbeiten der Beschreibung der serviceverknüpften Rolle
- Löschen der serviceverknüpften Rolle

Informationen zur AWS KMS serviceverknüpften Rolle

Eine dienstverknüpfte Rolle ist eine IAM-Rolle, die einem AWS Dienst die Erlaubnis erteilt, andere AWS Dienste in Ihrem Namen aufzurufen. Sie wurde entwickelt, um Ihnen die Nutzung der Funktionen mehrerer integrierter AWS Dienste zu erleichtern, ohne komplexe IAM-Richtlinien erstellen und verwalten zu müssen. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen für AWS KMS.

AWS KMS Erstellt für AWS CloudHSM Schlüsselspeicher die AWSServiceRoleForKeyManagementServiceCustomKeyStoresdienstverknüpfte Rolle mit der AWSKeyManagementServiceCustomKeyStoresServiceRolePolicyverwalteten Richtlinie. Diese Richtlinie gewährt der Rolle die folgenden Berechtigungen:

- cloudHSM:Describe* — erkennt Änderungen im AWS CloudHSM Cluster, der an Ihren benutzerdefinierten Schlüsselspeicher angehängt ist.
- ec2: CreateSecurityGroup — wird verwendet, wenn Sie einen AWS CloudHSM Schlüsselspeicher verbinden, um die Sicherheitsgruppe zu erstellen, die den Netzwerkdatenfluss zwischen und Ihrem Cluster ermöglicht. AWS KMS AWS CloudHSM
- ec2: AuthorizeSecurityGroupIngress — wird verwendet, wenn Sie einen AWS CloudHSM Schlüsselspeicher verbinden, um den Netzwerkzugriff von der VPC aus AWS KMS zu ermöglichen, die Ihren AWS CloudHSM Cluster enthält.
- ec2: CreateNetworkInterface — wird verwendet, wenn Sie einen AWS CloudHSM Schlüsselspeicher verbinden, um die Netzwerkschnittstelle zu erstellen, die für die Kommunikation zwischen AWS KMS und dem Cluster verwendet wird. AWS CloudHSM
- ec2: RevokeSecurityGroupEgress — wird verwendet, wenn Sie eine Verbindung zu einem AWS CloudHSM Schlüsselspeicher herstellen, um alle ausgehenden Regeln aus der Sicherheitsgruppe zu entfernen, die erstellt wurde. AWS KMS
- ec2: DeleteSecurityGroup — wird verwendet, wenn Sie die Verbindung zu einem AWS CloudHSM Schlüsselspeicher trennen, um Sicherheitsgruppen zu löschen, die beim Verbinden mit dem AWS CloudHSM Schlüsselspeicher erstellt wurden.
- ec2: DescribeSecurityGroups — wird verwendet, um Änderungen in der Sicherheitsgruppe zu überwachen, die in der VPC AWS KMS erstellt wurde, die Ihren AWS CloudHSM Cluster enthält, sodass bei Ausfällen klare Fehlermeldungen ausgegeben werden AWS KMS können.

- ec2: DescribeVpcs — wird verwendet, um Änderungen in der VPC zu überwachen, die Ihren AWS CloudHSM Cluster enthält, sodass bei Ausfällen klare Fehlermeldungen ausgegeben werden AWS KMS können.
- ec2: DescribeNetworkAcls — wird verwendet, um Änderungen im Netzwerk ACLs für die VPC zu überwachen, die Ihren AWS CloudHSM Cluster enthält, sodass bei Ausfällen klare Fehlermeldungen ausgegeben werden AWS KMS können.
- ec2: DescribeNetworkInterfaces — wird verwendet, um Änderungen an den Netzwerkschnittstellen zu überwachen, die in der VPC AWS KMS erstellt wurden, die Ihren AWS CloudHSM Cluster enthält, sodass bei Ausfällen klare Fehlermeldungen ausgegeben werden AWS KMS können.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudhsm:Describe*",
        "ec2:CreateNetworkInterface",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateSecurityGroup",
        "ec2:DescribeSecurityGroups",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:DeleteSecurityGroup",
        "ec2:DescribeVpcs",
        "ec2:DescribeNetworkAcls",
        "ec2:DescribeNetworkInterfaces"
      ],
      "Resource": "*"
    }
  ]
}
```

Da die AWSServiceRoleForKeyManagementServiceCustomKeyStoresdienstgebundene Rolle nur vertrauenswürdig ist`cks.kms.amazonaws.com`, AWS KMS kann sie nur diese dienstgebundene Rolle übernehmen. Diese Rolle ist auf die Vorgänge beschränkt, die zum Anzeigen Ihrer AWS CloudHSM Cluster und zum Verbinden eines AWS CloudHSM Schlüsselspeichers mit dem

zugehörigen Cluster AWS KMS erforderlich sind. AWS CloudHSM Sie gewährt AWS KMS keine zusätzlichen Berechtigungen. Sie ist beispielsweise AWS KMS nicht berechtigt, Ihre AWS CloudHSM Cluster oder Backups zu erstellen HSMs, zu verwalten oder zu löschen.

Regionen

Wie die Funktion „ AWS CloudHSM Schlüsselspeicher“ wird die AWSServiceRoleForKeyManagementServiceCustomKeyStoresRolle AWS-Regionen überall unterstützt AWS KMS und AWS CloudHSM ist verfügbar. Eine Liste der Funktionen AWS-Regionen , die von den einzelnen Diensten unterstützt werden, finden Sie unter AWS Key Management Service Endpunkte und Kontingente und AWS CloudHSM Endpunkte und Kontingente in der. Allgemeine Amazon Web Services-Referenz

Weitere Informationen darüber, wie AWS Dienste dienstverknüpfte Rollen verwenden, finden Sie unter Verwenden von dienstverknüpften Rollen im IAM-Benutzerhandbuch.

Erstellen der serviceverknüpften Rolle

AWS KMS erstellt automatisch die AWSServiceRoleForKeyManagementServiceCustomKeyStoresserviceverknüpfte Rolle in Ihrem AWS-Konto , wenn Sie einen AWS CloudHSM Schlüsselspeicher erstellen, sofern die Rolle noch nicht vorhanden ist. Sie können diese serviceverknüpfte Rolle nicht direkt erstellen oder direkt neu erstellen.

Bearbeiten der Beschreibung der serviceverknüpften Rolle

Sie können den Namen der Rolle oder die Richtlinienanweisungen in der serviceverknüpften Rolle AWSServiceRoleForKeyManagementServiceCustomKeyStores nicht bearbeiten. Sie können jedoch die Beschreibung der Rolle bearbeiten. Anweisungen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen der serviceverknüpften Rolle

AWS KMS löscht die AWSServiceRoleForKeyManagementServiceCustomKeyStoresdienstbezogene Rolle nicht aus Ihrem, AWS-Konto auch wenn Sie alle Ihre AWS CloudHSM Schlüsselspeicher gelöscht haben. Derzeit gibt es zwar kein Verfahren zum Löschen der AWSServiceRoleForKeyManagementServiceCustomKeyStoresdienstbezogenen Rolle, AWS KMS nimmt diese Rolle jedoch nicht an und verwendet auch nicht ihre Berechtigungen, sofern Sie nicht über aktive AWS CloudHSM Schlüsselspeicher verfügen.

Autorisierung zur Synchronisation von AWS KMS Schlüsseln aus mehreren Regionen

Um Schlüssel mit mehreren Regionen zu unterstützen, ist die Erlaubnis AWS KMS erforderlich, die gemeinsamen Eigenschaften eines Primärschlüssels mit mehreren Regionen mit seinen Replikatschlüsseln zu synchronisieren. Um diese Berechtigungen zu erhalten, AWS KMS erstellt die AWSServiceRoleForKeyManagementServiceMultiRegionKeysdienstbezogene Rolle in Ihrem. AWS-Konto Benutzer, die Schlüssel für mehrere Regionen erstellen, müssen über die `iam:CreateServiceLinkedRole` entsprechende Berechtigung verfügen, um dienstbezogene Rollen zu erstellen.

Sie können das SynchronizeMultiRegionKey CloudTrail Ereignis, das die AWS KMS Synchronisierung gemeinsam genutzter Eigenschaften aufzeichnet, in Ihren Protokollen einsehen. AWS CloudTrail

Einzelheiten zu Aktualisierungen der AWSKeyManagementServiceMultiRegionKeysServiceRolePolicyverwalteten Richtlinie finden Sie unterAWS KMS Aktualisierungen der verwalteten Richtlinien AWS.

Themen

- Über die serviceverknüpfte Rolle für multiregionale Schlüssel
- Erstellen der serviceverknüpften Rolle
- Bearbeiten der Beschreibung der serviceverknüpften Rolle
- Löschen der serviceverknüpften Rolle

Über die serviceverknüpfte Rolle für multiregionale Schlüssel

Eine dienstbezogene Rolle ist eine IAM-Rolle, die einem AWS Dienst die Erlaubnis erteilt, andere AWS Dienste in Ihrem Namen aufzurufen. Sie wurde entwickelt, um Ihnen die Nutzung der Funktionen mehrerer integrierter AWS Dienste zu erleichtern, ohne komplexe IAM-Richtlinien erstellen und verwalten zu müssen.

Bei Schlüsseln mit mehreren Regionen AWS KMS wird die AWSServiceRoleForKeyManagementServiceMultiRegionKeysdienstbezogene Rolle mit der AWSKeyManagementServiceMultiRegionKeysServiceRolePolicyverwalteten Richtlinie erstellt. Diese Richtlinie gibt der Rolle die `kms:SynchronizeMultiRegionKey`-Berechtigung, die es ermöglicht, die freigegebenen Eigenschaften von multiregionalen Schlüsseln zu synchronisieren.

Da die AWSServiceRoleForKeyManagementServiceMultiRegionKeysdienstgebundene Rolle nur vertrauenswürdig ist`mrk.kms.amazonaws.com`, AWS KMS kann sie nur diese dienstbezogene Rolle übernehmen. Diese Rolle ist auf die Vorgänge beschränkt, bei denen gemeinsam genutzte Eigenschaften mehrerer Regionen synchronisiert AWS KMS werden müssen. Sie gewährt AWS KMS keine zusätzlichen Berechtigungen. AWS KMS Hat beispielsweise keine Berechtigung, KMS-Schlüssel zu erstellen, zu replizieren oder zu löschen.

Weitere Informationen darüber, wie AWS Dienste dienstverknüpfte Rollen verwenden, finden Sie unter Verwenden von dienstverknüpften Rollen im IAM-Benutzerhandbuch.

JSON

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Sid" : "KMSSynchronizeMultiRegionKey",
            "Effect" : "Allow",
            "Action" : [
                "kms:SynchronizeMultiRegionKey"
            ],
            "Resource" : "*"
        }
    ]
}
```

Erstellen der serviceverknüpften Rolle

AWS KMS erstellt automatisch die AWSServiceRoleForKeyManagementServiceMultiRegionKeysserviceverknüpfte Rolle in Ihrem, AWS-Konto wenn Sie einen Schlüssel für mehrere Regionen erstellen, sofern die Rolle noch nicht vorhanden ist. Sie können diese serviceverknüpfte Rolle nicht direkt erstellen oder direkt neu erstellen.

Bearbeiten der Beschreibung der serviceverknüpften Rolle

Sie können den Rollennamen oder die Richtlinienerklärungen in der AWSServiceRoleForKeyManagementServiceMultiRegionKeysdienstbezogenen Rolle nicht

bearbeiten, aber Sie können die Rollenbeschreibung bearbeiten. Anweisungen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen der serviceverknüpften Rolle

AWS KMS löscht die AWSServiceRoleForKeyManagementServiceMultiRegionKeysdienstverknüpfte Rolle nicht aus Ihrer AWS-Konto und Sie können sie nicht löschen. Übernimmt die AWSServiceRoleForKeyManagementServiceMultiRegionKeysRolle jedoch AWS KMS nicht und verwendet keine ihrer Berechtigungen, es sei denn, Sie haben Schlüssel für mehrere Regionen in Ihrer Region AWS-Konto und Ihrer Region.

Einloggen und Überwachen AWS Key Management Service

Überwachung ist wichtig, um die Verfügbarkeit, den Status und die Nutzung Ihrer AWS KMS keys in AWS KMS zu verstehen. Die Überwachung trägt dazu bei, die Sicherheit, Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer AWS Lösungen aufrechtzuerhalten. AWS bietet mehrere Tools zur Überwachung Ihrer KMS-Schlüssel.

AWS CloudTrail Logs

Jeder Aufruf einer AWS KMS API-Operation wird als Ereignis in einem AWS CloudTrail Protokoll erfasst. In diesen Protokollen werden alle API-Aufrufe von der AWS KMS Konsole sowie Aufrufe von AWS KMS und anderen AWS Diensten aufgezeichnet. Kontoübergreifende API-Aufrufe, z. B. ein Aufruf zur Verwendung eines KMS-Schlüssels in einem anderen AWS-Konto, werden in den CloudTrail Protokollen beider Konten aufgezeichnet.

Bei der Fehlerbehebung oder Überwachung können Sie mithilfe des Protokolls den Lebenszyklus eines KMS-Schlüssels rekonstruieren. Sie können auch die Verwaltung und Verwendung des KMS-Schlüssels in kryptografischen Operationen anzeigen. Weitere Informationen finden Sie unter the section called “Protokollierung mit AWS CloudTrail”.

CloudWatch Amazon-Protokolle

Überwachen, speichern und greifen Sie auf Ihre Protokolldateien aus AWS CloudTrail und anderen Quellen zu. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.

For CloudWatch speichert nützliche Informationen AWS KMS, die Ihnen helfen, Probleme mit Ihren KMS-Schlüsseln und den Ressourcen, die sie schützen, zu vermeiden. Weitere Informationen finden Sie unter the section called “Überwachen Sie die Schlüssel mit CloudWatch”.

Amazon EventBridge

AWS KMS generiert EventBridge Ereignisse, wenn Ihr KMS-Schlüssel rotiert oder gelöscht wird oder wenn das importierte Schlüsselmaterial in Ihrem KMS-Schlüssel abläuft. Suchen Sie nach AWS KMS Ereignissen (API-Operationen) und leiten Sie sie an eine oder mehrere Zielfunktionen oder Streams weiter, um Statusinformationen zu erfassen. Weitere Informationen finden Sie unter the section called “Tasten mit Amazon überwachen EventBridge” und im EventBridge Amazon-Benutzerhandbuch.

CloudWatch Amazon-Metriken

Sie können Ihre KMS-Schlüssel mithilfe von CloudWatch Metriken überwachen, bei denen Rohdaten gesammelt und AWS KMS zu Leistungskennzahlen verarbeitet werden. Die Daten werden in zweiwöchigen Intervallen aufgezeichnet, sodass Sie Trends aktueller und historischer Informationen anzeigen können. Auf diese Weise können Sie verstehen, wie Ihre KMS-Schlüssel verwendet werden und wie sich ihre Verwendung im Laufe der Zeit ändert. Informationen zur Verwendung von CloudWatch Metriken zur Überwachung von KMS-Schlüsseln finden Sie unterAWS KMS Metriken und Dimensionen.

CloudWatch Amazon-Alarme

Überwachen Sie eine Metrik über einen bestimmten, von Ihnen definierten Zeitraum. Der Alarm führt eine oder mehrere Aktionen durch, basierend auf dem Wert der Metrik im Vergleich zu einem bestimmten Schwellenwert in einer Reihe von Zeiträumen. Sie können beispielsweise einen CloudWatch Alarm erstellen, der ausgelöst wird, wenn jemand versucht, einen KMS-Schlüssel zu verwenden, dessen Löschung im Rahmen eines kryptografischen Vorgangs geplant ist. Dies zeigt an, dass der KMS-Schlüssel noch verwendet wird und wahrscheinlich nicht gelöscht werden sollte. Weitere Informationen finden Sie unter the section called “Alarm erstellen”.

AWS Security Hub

Mithilfe dieser Methode können Sie Ihre AWS KMS Nutzung im Hinblick auf Sicherheit, Industriestandards und bewährte Verfahren AWS Security Hubüberwachen. Security Hub verwendet Sicherheitskontrollen für die Bewertung von Ressourcenkonfigurationen und Sicherheitsstandards, um Sie bei der Einhaltung verschiedener Compliance-Frameworks zu unterstützen. Weitere Informationen finden Sie unter AWS Key Management Service - Steuerungen im AWS Security Hub -Benutzerhandbuch.

Konformitätsvalidierung für AWS Key Management Service

Externe Prüfer bewerten die Sicherheit und Einhaltung von Vorschriften im AWS Key Management Service Rahmen mehrerer AWS Compliance-Programme. Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Themen

- Compliance-und Sicherheitsdokumente
- Weitere Informationen

Compliance-und Sicherheitsdokumente

Die folgenden Konformitäts- und Sicherheitsdokumente decken Folgendes ab AWS KMS. Um sie anzuzeigen, verwenden Sie AWS Artifact.

- Cloud Computing Compliance Controls Catalogue (C5)
- ISO 27001:2013 Erklärung zur Anwendbarkeit (SoA)
- ISO 27001:2013 Zertifizierung
- ISO 27017:2015 Erklärung zur Anwendbarkeit (SoA)
- ISO 27017:2015 Zertifizierung
- ISO 27018:2015 Erklärung zur Anwendbarkeit (SoA)
- ISO 27018:2014 Zertifizierung
- ISO 9001:2015 Zertifizierung
- PCI-DSS-Compliance-Nachweis (AOC) und Zusammenfassung der Verantwortlichkeiten
- Service Organization Controls (SOC)-Bericht 1
- Service Organization Controls (SOC)-Bericht 2
- Service Organization Controls (SOC)-Bericht 2 zur Vertraulichkeit
- FedRAMP-hoch

Hilfe zur Verwendung AWS Artifact finden Sie unter Berichte in AWS Artifact herunterladen.

Weitere Informationen

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS KMS hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Wenn Ihre Verwendung von der Einhaltung eines veröffentlichten Standards AWS KMS unterliegt, AWS bietet Ihnen Ressourcen, die Sie bei folgenden Themen unterstützen:

- AWS Services im Geltungsbereich nach Compliance-Programmen — Auf dieser Seite sind AWS Services aufgeführt, die in den Geltungsbereich bestimmter Compliance-Programme fallen. Allgemeine Informationen finden Sie unter AWS -Compliance-Programme.
- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Umgebungen beschrieben, auf denen Sicherheit und Compliance im Vordergrund stehen. AWS
- AWS Ressourcen zur Einhaltung von Vorschriften — Diese Sammlung von Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- AWS Config— Dieser AWS Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Dieser AWS Service bietet einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS Ressourcen zu bewerten und Ihre Einhaltung der Sicherheitsstandards und Best Practices der Sicherheitsbranche zu überprüfen. Die Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerelementreferenz.

Resilienz in AWS Key Management Service

Die AWS globale Infrastruktur basiert auf Availability AWS-Regionen Zones. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Zusätzlich zur AWS globalen Infrastruktur AWS KMS bietet es mehrere Funktionen zur Unterstützung Ihrer Datenausfallsicherheit und Backup-Anforderungen. Weitere Informationen zu AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Regionale Isolierung

AWS Key Management Service (AWS KMS) ist ein sich selbst tragender regionaler Service, der überall verfügbar ist. AWS-Regionen Das regional isolierte Design von AWS KMS stellt sicher, dass ein Verfügbarkeitsproblem in einer Region den AWS KMS Betrieb in keiner anderen Region beeinträchtigen AWS-Region kann. AWS KMS ist so konzipiert, dass keine geplanten Ausfallzeiten auftreten und alle Softwareupdates und Skalierungsvorgänge nahtlos und unmerklich ausgeführt werden.

Das AWS KMS Service Level Agreement (SLA) beinhaltet eine Serviceverpflichtung von 99,999% für alle KMS. APIs Um diese Verpflichtung zu erfüllen, stellt AWS KMS sicher, dass alle Daten und Berechtigungsinformationen, die zur Ausführung einer API-Anfrage erforderlich sind, auf allen regionalen Hosts verfügbar sind, die die Anfrage erhalten.

Die AWS KMS Infrastruktur wird in mindestens drei Availability Zones (AZs) in jeder Region repliziert. Um sicherzustellen, dass mehrere Hostausfälle die AWS KMS Leistung nicht beeinträchtigen, AWS KMS ist es so konzipiert, dass Kundendatenverkehr aus allen Regionen einer AZs Region bedient wird.

Änderungen, die Sie an den Eigenschaften oder Berechtigungen eines KMS-Schlüssels vornehmen, werden auf alle Hosts in der Region repliziert, um sicherzustellen, dass nachfolgende Anfragen von jedem Host in der Region korrekt verarbeitet werden können. Anfragen für kryptografische Operationen unter Verwendung Ihres KMS-Schlüssels werden an eine Reihe von AWS KMS Hardware-Sicherheitsmodulen (HSMs) weitergeleitet, von denen jedes den Vorgang mit dem KMS-Schlüssel ausführen kann.

Design mit mehreren Mandanten

Das mandantenfähige Design von AWS KMS ermöglicht es, das SLA für eine Verfügbarkeit von 99,999% zu erfüllen und hohe Anforderungsraten aufrechtzuerhalten und gleichzeitig die Vertraulichkeit Ihrer Schlüssel und Daten zu schützen.

Es werden mehrere integritätserzwingende Mechanismen eingesetzt, um sicherzustellen, dass der KMS-Schlüssel, den Sie für die kryptografische Operation angegeben haben, immer verwendet wird.

Das Klartext-Schlüsselmaterial für Ihre KMS-Schlüssel ist umfangreich geschützt. Das Schlüsselmaterial wird im HSM verschlüsselt, sobald es erstellt wird, und das verschlüsselte Schlüsselmaterial wird sofort in einen sicheren Speicher mit geringer Latenzzeit übertragen. Der verschlüsselte Schlüssel wird im HSM abgerufen und entschlüsselt, sobald er benutzt wird.

Der Klartextschlüssel verbleibt nur so lange im HSM-Speicher, wie er für den Abschluss der kryptografischen Operation benötigt wird. Dann wird er im HSM erneut verschlüsselt und der verschlüsselte Schlüssel wird wieder gespeichert. Schlüsselmaterial im Klartext-Format verlässt niemals den und wird niemals in HSMs den persistenten Speicher geschrieben.

Bewährte Methoden zur Resilienz in AWS KMS

Um die Widerstandsfähigkeit Ihrer AWS KMS Ressourcen zu optimieren, sollten Sie die folgenden Strategien in Betracht ziehen.

- Um Ihre Sicherungs- und Notfallwiederherstellungs-Strategie zu unterstützen, sollten Sie multiregionale Schlüssel in Betracht ziehen. Dabei handelt es sich um KMS-Schlüssel, die in einer AWS-Region erstellt und nur in von Ihnen angegebene Regionen repliziert werden. Mit Schlüsseln für mehrere Regionen können Sie verschlüsselte Ressourcen zwischen AWS-Regionen (innerhalb derselben Partition) verschieben, ohne jemals den Klartext preiszugeben, und die Ressource bei Bedarf in einer der Zielregionen entschlüsseln. Zusammengehörige multiregionale Schlüssel sind interoperabel, da sie dasselbe Schlüsselmaterial und dieselbe Schlüssel-ID verwenden, aber sie haben unabhängige Schlüsselrichtlinien für eine hochauflösende Zugangskontrolle. Weitere Informationen zu multiregionalen Schlüsseln finden Sie unter AWS KMS.
- Um Ihre Schlüssel in einem Dienst mit mehreren Mandanten zu schützen AWS KMS, sollten Sie Zugriffskontrollen verwenden, einschließlich Schlüsselrichtlinien und IAM-Richtlinien. Darüber hinaus können Sie Ihre Anfragen an AWS KMS einen VPC-Schnittstellenendpunkt senden, der von AWS PrivateLink betrieben wird. Wenn Sie dies tun, AWS KMS erfolgt die gesamte Kommunikation zwischen Ihrer Amazon VPC und ausschließlich innerhalb des AWS Netzwerks über einen dedizierten AWS KMS Endpunkt, der auf Ihre VPC beschränkt ist. Sie können diese Anfragen weiter absichern, indem Sie mit VPC-Endpunktrichtlinien eine zusätzliche Autorisierungsebene schaffen. Weitere Informationen finden Sie unter Verbinden mit AWS KMS über einen VPC-Endpunkt.

Sicherheit der Infrastruktur in AWS Key Management Service

Als verwalteter Service ist AWS Key Management Service (AWS KMS) durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die in Amazon Web Services: Überblick über Sicherheitsprozesse beschrieben sind.

Für den Zugriff AWS KMS über das Netzwerk können Sie die AWS KMS API-Operationen aufrufen, die in der AWS Key Management Service API-Referenz beschrieben sind. AWS KMS erfordert TLS

1.2 und empfiehlt TLS 1.3 in allen Regionen. AWS KMS unterstützt auch hybrides Post-Quantum-TLS für AWS KMS Service-Endpunkte in allen Regionen außer den Regionen China. AWS KMS unterstützt kein hybrides Post-Quantum-TLS für FIPS-Endpunkte in. AWS GovCloud (US) Um die Standard- AWS KMS -Endpunkte oder AWS KMS -FIPS-Endpunkte zu verwenden, müssen Clients TLS 1.2 oder höher unterstützen. Clients müssen außerdem Cipher-Suites mit Perfect Forward Secrecy (PFS) wie Ephemeral Diffie-Hellman (DHE) oder Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) unterstützen. Die meisten modernen Systeme, z. B. Java 7 und höher, unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Sie können diese API-Operationen von jedem Netzwerkstandort aus aufrufen, AWS KMS unterstützt jedoch globale Richtlinienbedingungen, mit denen Sie den Zugriff auf einen KMS-Schlüssel auf der Grundlage der Quell-IP-Adresse, der VPC und des VPC-Endpunkts steuern können. Sie können diese Bedingungsschlüssel in Schlüsselrichtlinien und IAM-Richtlinien verwenden. Diese Bedingungen können jedoch AWS verhindern, dass der KMS-Schlüssel in Ihrem Namen verwendet wird. Details hierzu finden Sie unter AWS globale Bedingungsschlüssel.

Die folgende wichtige Richtlinienanweisung ermöglicht es beispielsweise Benutzern, die diese `KMSTestRole` Rolle übernehmen können, diese AWS KMS key für die angegebenen kryptografischen Operationen zu verwenden, sofern es sich bei der Quell-IP-Adresse nicht um eine der in der Richtlinie angegebenen IP-Adressen handelt.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS":
    "arn:aws:iam::111122223333:role/KMSTestRole"},
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
      "kms:GenerateDataKey*",
      "kms:DescribeKey"
```

```
    ],
    "Resource": "*",
    "Condition": {
      "NotIpAddress": {
        "aws:SourceIp": [
          "192.0.2.0/24",
          "203.0.113.0/24"
        ]
      }
    }
  }
}
```

Isolierung auf physischen Hosts

Die Sicherheit der AWS KMS verwendeten physischen Infrastruktur unterliegt den Kontrollen, die im Abschnitt Physische Sicherheit und Umgebungssicherheit von Amazon Web Services: Überblick über Sicherheitsprozesse beschrieben sind. Weitere Details finden Sie in Compliance-Berichten und Prüfungsergebnissen von Drittanbietern, die im vorherigen Abschnitt aufgeführt sind.

AWS KMS wird von speziellen Sicherheitsmodulen für gehärtete Hardware (HSMs) unterstützt, die mit speziellen Kontrollen ausgestattet sind, um physischen Angriffen standzuhalten. Dabei HSMs handelt es sich um physische Geräte ohne Virtualisierungsebene, wie z. B. einen Hypervisor, der das physische Gerät von mehreren logischen Mandanten gemeinsam genutzt wird. Das Schlüsselmaterial für AWS KMS keys wird nur im flüchtigen Speicher auf dem gespeichert HSMs, und nur solange der KMS-Schlüssel verwendet wird. Dieser Speicher wird gelöscht, wenn das HSM den Betriebszustand verlässt, einschließlich beabsichtigtem und unbeabsichtigtem Herunterfahren und Zurücksetzen. Ausführliche Informationen zur Funktionsweise von finden Sie AWS KMS HSMs unter AWS Key Management Service Kryptografische Details.

Kontingente

AWS KMS Wendet zwei Arten von Kontingenten an: Ressourcenkontingente und Anforderungskontingente, um für alle Benutzer AWS KMS reaktionsschnell und performant zu sein. Jedes Kontingent wird unabhängig für jede Region jedes AWS-Konto berechnet.

Alle AWS KMS Kontingente sind anpassbar, mit Ausnahme des Ressourcenkontingents für die On-Demand-Rotation und des Kontingents für AWS CloudHSM Key Store-Anfragen. Informationen zur Erhöhung eines Kontingents finden Sie unter Anfordern einer Kontingenterhöhung im Service-Quotas-Benutzerhandbuch. Um eine Senkung des Kontingents zu beantragen, ein Kontingent zu ändern, das nicht in Service Quotas aufgeführt ist, oder um ein Kontingent AWS-Region in einem Bereich zu ändern, für AWS KMS den Servicekontingente nicht verfügbar sind, besuchen Sie bitte das AWS -Support Center und erstellen Sie einen Fall.

Themen

- Ressourcenkontingente
- Anforderungskontingente
- Drosselung AWS KMS von Anfragen

Ressourcenkontingente

AWS KMS legt Ressourcenkontingente fest, um sicherzustellen, dass all unseren Kunden ein schneller und zuverlässiger Service geboten werden kann. Einige Ressourcenkontingente gelten nur für Ressourcen, die Sie erstellen, nicht jedoch für Ressourcen, die AWS Dienste für Sie erstellen. Ressourcen, die Sie verwenden, die sich jedoch nicht in ihrem AWS-Konto befinden, z. B. AWS-eigene Schlüssel, werden auf diese Kontingente nicht angerechnet.

Wenn Sie ein Ressourcenlimit erreicht haben, generieren Anforderungen zum Erstellen einer zusätzlichen Ressource dieses Typs die Fehlermeldung `LimitExceededException`.

Alle AWS KMS Ressourcenkontingente sind anpassbar, mit Ausnahme des Ressourcenkontingents für die On-Demand-Rotation. Informationen zur Erhöhung eines Kontingents finden Sie unter Anfordern einer Kontingenterhöhung im Service-Quotas-Benutzerhandbuch. Um eine Senkung des Kontingents zu beantragen, ein Kontingent zu ändern, das nicht in Service Quotas aufgeführt ist, oder um ein Kontingent AWS-Region in einem Bereich zu ändern, für AWS KMS den Servicekontingente nicht verfügbar sind, besuchen Sie bitte das AWS -Support Center und erstellen Sie einen Fall.

In der folgenden Tabelle sind die AWS KMS Ressourcenkontingente in den einzelnen Regionen aufgeführt AWS-Konto und beschrieben.

Kontingentname	Standardwert	Gilt für	Einstellbar
AWS KMS keys	100 000	Kundenverwaltete Schlüssel	Ja
Aliasse pro KMS-Schlüssel	50	Kundenseitig erstellte Aliasse	Ja
Erteilungen pro KMS-Schlüssel	50 000	Kundenverwaltete Schlüssel	Ja
Ressourcenkontingent für benutzerdefinierte Schlüsselspeicher	10	AWS-Konto und Region	Ja
Rotation auf Anfrage	10	Kundenverwaltete Schlüssel	Nein

AWS KMS Verwendet zusätzlich zu den Ressourcenkontingenten Anforderungskontingente, um die Reaktionsfähigkeit des Dienstes sicherzustellen. Details hierzu finden Sie unter the section called “Anforderungskontingente”.

AWS KMS keys: 100.000

In jeder Region Ihres AWS-Konto sind bis zu 100.000 kundenverwaltete Schlüssel zulässig. Dieses Kontingent gilt für alle vom Kunden verwaltete Schlüssel in allen AWS-Regionen , unabhängig von ihrer Schlüsselspezifikation oder ihrem Schlüsselstatus. Jeder KMS-Schlüssel wird als eine Ressource betrachtet. Von AWS verwaltete Schlüssel und AWS-eigene Schlüssel werden nicht auf dieses Kontingent angerechnet.

Aliasse pro KMS-Schlüssel: 50

Sie können jedem kundenverwalteten Schlüssel bis zu 50 Aliasse zuordnen. Aliase, die mit AWS verknüpft sind, werden Von AWS verwaltete Schlüsselnicht auf dieses Kontingent angerechnet. Dieses Kontingent kann auftreten, wenn Sie ein Alias erstellen oder aktualisieren.

> **Note**
>
> Die ResourceAliases Bedingung kms: ist nur wirksam, wenn der KMS-Schlüssel diesem Kontingent entspricht. Wenn ein KMS-Schlüssel dieses Kontingent überschreitet, wird auch Prinzipalen, die berechtigt sind, den KMS-Schlüssel zu nutzen, durch die Bedingung `kms:ResourceAliases` der Zugriff auf den KMS-Schlüssel verweigert. Details hierzu finden Sie unter Zugriff aufgrund eines Aliaskontingents verweigert.

Das Schlüsselkontingent Aliase pro KMS ersetzt das Kontingent Aliase pro Region, das die Gesamtzahl der Aliase in jeder Region eines beschränkte. AWS-Konto AWS KMS hat das Kontingent „Aliase pro Region“ abgeschafft.

Erteilungen pro KMS-Schlüssel: 50 000

Jedem kundenverwalteten Schlüssel können bis zu 50 000 Erteilungen zugewiesen werden, einschließlich der Erteilungen, die von in AWS KMS integrierten AWS -Services erstellt wurden. Dieses Kontingent gilt nicht für Von AWS verwaltete Schlüssel oder AWS-eigene Schlüssel.

Eine Auswirkung dieses Kontingents besteht darin, dass Sie gleichzeitig nicht mehr als 50 000 per Erteilung autorisierte Operationen durchführen können, die denselben KMS-Schlüssel verwenden. Nachdem Sie das Kontingent erschöpft haben, können Sie neue Erteilungen für den KMS-Schlüssel erst erstellen, nachdem eine aktive Erteilung aufgehoben oder widerrufen wird.

Wenn Sie beispielsweise ein Amazon Elastic Block Store (Amazon EBS) -Volume an eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance anhängen, wird das Volume entschlüsselt, sodass Sie es lesen können. Um die Berechtigung zum Entschlüsseln der Daten zu erhalten, erstellt Amazon EBS für jedes Volume eine Erteilung. Wenn also alle Ihre Amazon-EBS-Volumes denselben KMS-Schlüssel verwenden, können Sie nicht mehr als 50 000 Volumes gleichzeitig anhängen.

Ressourcenkontingent für benutzerdefinierte Schlüsselspeicher: 10

Sie können in jeder AWS-Konto Region bis zu 10 benutzerdefinierte Schlüsselspeicher erstellen. Wenn Sie versuchen, weitere zu erstellen, schlägt der CreateCustomKeyStoreVorgang fehl.

Dieses Kontingent gilt für die Gesamtzahl der benutzerdefinierten Schlüsselspeicher in jedem Konto und jeder Region, einschließlich aller AWS CloudHSM -Schlüsselspeicher und externer Schlüsselspeicher, unabhängig von ihrem Verbindungsstatus.

Rotation auf Anfrage: 10

Sie können die Schlüsselrotation bei Bedarf maximal 10 Mal pro KMS-Schlüssel durchführen. Wenn Sie versuchen, mehr Rotationen auf Anforderung durchzuführen, schlägt der RotateKeyOnDemandVorgang fehl.

Dieses Kontingent ist nicht anpassbar. Sie können es nicht erhöhen, indem Sie Servicekontingente verwenden oder einen Fall in erstellen AWS -Support. Um zu verhindern, dass das Kontingent für die On-Demand-Rotation erreicht wird, empfehlen wir, wann immer möglich die automatische Schlüsselrotation zu verwenden.

Anforderungskontingente

AWS KMS legt Kontingente für die Anzahl der in jeder Sekunde angeforderten API-Operationen fest. Die Anforderungskontingente unterscheiden sich je nach API-Vorgang AWS-Region, dem und anderen Faktoren, wie dem KMS-Schlüsseltyp. Wenn Sie ein API-Anforderungskontingent überschreiten, wird die Anfrage AWS KMS gedrosselt.

Alle AWS KMS Anforderungskontingente sind einstellbar, mit Ausnahme des Anforderungskontingents für AWS CloudHSM Schlüsselspeicher. Informationen zur Erhöhung eines Kontingents finden Sie unter Anfordern einer Kontingenterhöhung im Service-Quotas-Benutzerhandbuch. Um eine Senkung des Kontingents zu beantragen, ein Kontingent zu ändern, das nicht in Service Quotas aufgeführt ist, oder um ein Kontingent AWS-Region in einem Bereich zu ändern, für AWS KMS den Servicekontingente nicht verfügbar sind, besuchen Sie bitte das AWS -Support Center und erstellen Sie einen Fall.

Wenn Sie das Anforderungskontingent für den GenerateDataKeyVorgang überschreiten, sollten Sie in Erwägung ziehen, die Funktion zum Zwischenspeichern von Datenschlüsseln zu verwenden. AWS Encryption SDK Durch die Wiederverwendung von Datenschlüsseln kann sich die Häufigkeit Ihrer Anfragen auf verringern. AWS KMS

Zusätzlich zu den Anforderungskontingenten werden Ressourcenkontingente AWS KMS verwendet, um die Kapazität für alle Benutzer sicherzustellen. Details hierzu finden Sie unter Ressourcenkontingente.

Um Trends in Ihren Anforderungsraten anzuzeigen, verwenden Sie die Service-Quotas-Konsole. Sie können auch einen CloudWatchAmazon-Alarm einrichten, der Sie benachrichtigt, wenn Ihre Anforderungsrate einen bestimmten Prozentsatz eines Kontingents erreicht. Einzelheiten finden Sie

im AWS Sicherheitsblog unter Verwalten Sie Ihre AWS KMS API-Anforderungsraten mithilfe von Service Quotas und Amazon CloudWatch.

Themen

- Fordern Sie Kontingente für jeden AWS KMS API-Vorgang an
- Anwenden von Anforderungskontingenten
- Gemeinsame Kontingente für kryptografische Operationen
- API-Anforderungen in Ihrem Namen
- Kontoübergreifende Anforderungen
- Anforderungskontingente für benutzerdefinierte Schlüsselspeicher

Fordern Sie Kontingente für jeden AWS KMS API-Vorgang an

In dieser Tabelle sind der Quota-Code für Service-Kontingente und der Standardwert für jedes AWS KMS Anforderungskontingent aufgeführt. Alle AWS KMS Anforderungskontingente sind einstellbar, mit Ausnahme des Anforderungskontingents für AWS CloudHSM Schlüsselspeicher.

> **Note**
>
> Möglicherweise müssen Sie horizontal oder vertikal Scrollen, um alle Daten in dieser Tabelle anzuzeigen.

Kontingentname	Standardwert (Anforderungen pro Sekunde)

`Cryptographic operations (symmetric) request rate`
 	
Gilt für:
 	
• `Decrypt`
• `Encrypt`
• `GenerateDataKey`
• `GenerateDataKeyWithoutPlaintext`
	Diese gemeinsamen Kontingente variieren je nach dem AWS-Region und dem Typ des in der Anfrage verwendeten KMS-Schlüssels. Jedes Kontingent wird separat berechnet.
	
	• 10.000 (gemeinsam genutzt)
	• 20.000 (gemeinsam genutzt) in den folgenden Regionen:
	  • USA Ost (Ohio), us-east-2
	  • Asien-Pazifik (Singapur), ap-southeast-1

Kontingentname	Standardwert (Anforderungen pro Sekunde)

• `GenerateMac`
• `GenerateRandom`
• `ReEncrypt`
• `VerifyMac`
	• Asien-Pazifik (Sydney), ap-southeast-2
	• Asien-Pazifik (Tokio), ap-northeast-1
	• Europa (Frankfurt) eu-central-1
	• Europa (London) eu-west-2
	• 100.000 (gemeinsam genutzt) in den folgenden Regionen:
	  • USA Ost (Nord-Virginia), us-east-1
	  • USA West (Oregon), us-west-2
	  • Europa (Irland), eu-west-1

`Cryptographic operations (RSA) request rate` Gilt für: • `Decrypt` • `Encrypt` • `ReEncrypt` • `Sign` • `Verify`	1.000 (gemeinsam genutzt) für RSA-KMS-Schlüssel
`Cryptographic operations (ML-DSA) request rate` Gilt für: • `Sign` • `Verify`	1.000 (gemeinsam genutzt) für ML-DSA-KMS-Schlüssel

Kontingentname	Standardwert (Anforderungen pro Sekunde)
`Cryptographic operations (ECC and SM2) request rate` Gilt für: • `Decrypt`— nur für SM2 KMS-Schlüssel (nur Regionen China) unterstützt • `DeriveSharedSecret` • `Encrypt`— nur für SM2 KMS-Schlüssel (nur Regionen China) unterstützt • `ReEncrypt` — nur für SM2 KMS-Schlüssel (nur Regionen China) unterstützt • `Sign` • `Verify`	1.000 (gemeinsam genutzt) für KMS-Schlüssel mit elliptischen Kurven (ECC) und SM2 (nur Regionen China)
`Custom key store request quotas` Gilt für: • `Decrypt` • `DeriveSharedSecret` • `Encrypt` • `GenerateDataKey` • `GenerateDataKeyWithoutPlaintext` • `GenerateRandom` • `ReEncrypt`	Anforderungskontingente für benutzerdefinierte Schlüsselspeicher werden für jeden benutzerd efinierten Schlüsselspeicher separat berechnet. • 1.800 (gemeinsam genutzt) für jeden Schlüsselspeicher AWS CloudHSM • 1 800 (freigegeben) für jeden externen Schlüsselspeicher.
`CancelKeyDeletion request rate`	5
`ConnectCustomKeyStore request rate`	5
`CreateAlias request rate`	5

Kontingentname	Standardwert (Anforderungen pro Sekunde)
`CreateCustomKeyStore request rate`	5
`CreateGrant request rate`	50
`CreateKey request rate`	5
`DeleteAlias request rate`	15
`DeleteCustomKeyStore request rate`	5
`DeleteImportedKeyMaterial request rate`	15
`DescribeCustomKeyStores request rate`	5
`DescribeKey request rate`	2000
`DisableKey request rate`	5
`DisableKeyRotation request rate`	5
`DisconnectCustomKeyStore request rate`	5
`EnableKey request rate`	5
`EnableKeyRotation request rate`	15
`GenerateDataKeyPair (ECC_NIST_P256) request rate` Gilt für: • `GenerateDataKeyPair` • `GenerateDataKeyPairWithoutPlaintext`	100

Kontingentname	Standardwert (Anforderungen pro Sekunde)
`GenerateDataKeyPair (ECC_NIST_P384) request rate` Gilt für: • `GenerateDataKeyPair` • `GenerateDataKeyPairWithoutPlaintext`	100
`GenerateDataKeyPair (ECC_NIST_P521) request rate` Gilt für: • `GenerateDataKeyPair` • `GenerateDataKeyPairWithoutPlaintext`	100
`GenerateDataKeyPair (ECC_SECG_P256K1) request rate` Gilt für: • `GenerateDataKeyPair` • `GenerateDataKeyPairWithoutPlaintext`	100
`GenerateDataKeyPair (RSA_2048) request rate` Gilt für: • `GenerateDataKeyPair` • `GenerateDataKeyPairWithoutPlaintext`	1

Kontingentname	Standardwert (Anforderungen pro Sekunde)
`GenerateDataKeyPair (RSA_3072) request rate` Gilt für: • `GenerateDataKeyPair` • `GenerateDataKeyPairWithoutPlaintext`	0,5 (1 in jedem 2-Sekunden-Intervall)
`GenerateDataKeyPair (RSA_4096) request rate` Gilt für: • `GenerateDataKeyPair` • `GenerateDataKeyPairWithoutPlaintext`	0,1 (1 in jedem 10-Sekunden-Intervall)
`GenerateDataKeyPair (SM2 — China Regions only) request rate` Gilt für: • `GenerateDataKeyPair` • `GenerateDataKeyPairWithoutPlaintext`	25
`GetKeyPolicy request rate`	1000
`GetKeyRotationStatus request rate`	1000
`GetParametersForImport request rate`	0,25 (1 in jedem 4-Sekunden-Intervall)
`GetPublicKey request rate`	2000
`ImportKeyMaterial request rate`	15

Kontingentname	Standardwert (Anforderungen pro Sekunde)
`ListAliases request rate`	500
`ListGrants request rate`	100
`ListKeyPolicies request rate`	100
`ListKeys request rate`	500
`ListKeyRotations request rate`	100
`ListResourceTags request rate`	2000
`ListRetirableGrants request rate`	100
`PutKeyPolicy request rate`	15
`ReplicateKey request rate` Eine `ReplicateKey` -Produktion zählt als eine `ReplicateKey` -Anforderung in der Region des Primärschlüssels und zwei `CreateKey` -Anforderungen in der Region des Replikats. Eine der `CreateKey` -Anforderungen ist ein Trockenlauf, um potenzielle Probleme zu erkennen, bevor der Schlüssel erstellt wird.	5
`RetireGrant request rate`	50
`RevokeGrant request rate`	50
`RotateKeyOnDemand request rate`	5
`ScheduleKeyDeletion request rate`	15
`TagResource request rate`	10
`UntagResource request rate`	5

Kontingentname	Standardwert (Anforderungen pro Sekunde)
`UpdateAlias request rate`	5
`UpdateCustomKeyStore request rate`	5
`UpdateKeyDescription request rate`	5
`UpdatePrimaryRegion request rate` Eine `UpdatePrimaryRegion` -Produktion zählt als zwei `UpdatePrimaryRegion` -Anforderungen; eine Anforderung in jeder der beiden betroffenen Regionen.	5

Anwenden von Anforderungskontingenten

Beachten Sie beim Anzeigen der Anforderungskontingente die folgenden Informationen.

- Anforderungskontingente gelten sowohl für kundenverwaltete Schlüssel als auch für Von AWS verwaltete Schlüssel. Die Verwendung von AWS-eigene Schlüsselwird nicht auf Ihre Anforderungskontingente angerechnet AWS-Konto, auch wenn sie zum Schutz der Ressourcen in Ihrem Konto verwendet werden.
- Anforderungskontingente gelten für Anforderungen, die an FIPS-Endpunkte und Nicht-FIPS-Endpunkte gesendet werden. Eine Liste der AWS KMS Dienstendpunkte finden Sie unter AWS Key Management Service Endpunkte und Kontingente in der. Allgemeine AWS-Referenz
- Die Drosselung basiert auf allen Anforderungen an KMS-Schlüssel aller Typen in der Region. Diese Summe beinhaltet Anfragen von allen Principals in der AWS-Konto, einschließlich Anfragen von AWS Diensten in Ihrem Namen.
- Jedes Anforderungskontingent wird separat berechnet. Anfragen für den CreateKeyVorgang haben beispielsweise keine Auswirkung auf das Anforderungskontingent für den CreateAliasVorgang. Wenn Ihre `CreateAlias`-Anforderungen gedrosselt werden, können Ihre `CreateKey`-Anforderungen weiterhin erfolgreich abgeschlossen werden.
- Obwohl für kryptografische Operationen ein gemeinsames Kontingent gilt, wird das gemeinsame Kontingent unabhängig von den Kontingenten für andere Operationen berechnet. Beispielsweise teilen sich Aufrufe der Vorgänge Verschlüsseln und Entschlüsseln ein Anforderungskontingent, aber dieses Kontingent ist unabhängig von dem Kontingent für Verwaltungsvorgänge, wie z.

EnableKey Beispielsweise können Sie in der Region Europa (London) 10 000 kryptografische Operationen für symmetrische KMS-Schlüssel plus 5 `EnableKey`-Operationen pro Sekunde ausführen, ohne gedrosselt zu werden.

Gemeinsame Kontingente für kryptografische Operationen

AWS KMS Bei kryptografischen Vorgängen werden Anforderungsquoten gemeinsam genutzt. Sie können eine beliebige Kombination der vom KMS-Schlüssel unterstützten kryptografischen Operationen anfordern. Dabei darf nur die Gesamtzahl der kryptografischen Operationen das Anforderungskontingent für diesen KMS-Schlüsseltyp nicht überschreiten. Die Ausnahmen sind GenerateDataKeyPairund GenerateDataKeyPairWithoutPlaintext, die sich ein separates Kontingent teilen.

Die Kontingente für unterschiedliche Schlüsseltypen werden ebenfalls unabhängig berechnet. Jedes Kontingent gilt für alle Anfragen für diese Operationen in der Region AWS-Konto und mit dem angegebenen Schlüsseltyp in jedem Intervall von einer Sekunde.

- Die Anforderungsrate für kryptographische Operationen (symmetrisch) ist das für kryptographische Operationen freigegebene Anforderungskontingent unter Verwendung symmetrischer KMS-Schlüssel in einem Konto und in einer Region. Dieses Kontingent gilt für kryptografische Operationen mit symmetrischen Verschlüsselungsschlüsseln und HMAC-Schlüsseln, die ebenfalls symmetrisch sind.

  Beispielsweise könnten Sie symmetrische KMS-Schlüssel in einem System AWS-Region mit einem gemeinsamen Kontingent von 10.000 Anfragen pro Sekunde verwenden. Wenn Sie 7.000 GenerateDataKeyAnfragen pro Sekunde und 2.000 Decrypt-Anfragen pro Sekunde stellen, werden Ihre Anfragen AWS KMS nicht gedrosselt. Wenn jedoch 9 500 `GenerateDataKey`-Anforderungen und 1 000 Encrypt-Anforderungen pro Sekunde anfallen, drosselt AWS KMS die Anforderungen, da sie das gemeinsame Kontingent überschreiten.

  Kryptografische Operationen mit den KMS-Schlüsseln für symmetrische Verschlüsselung in einem benutzerdefinierten Schlüsselspeicher werden sowohl auf die Anforderungsrate kryptografischer Operationen (symmetrisch) für das Konto als auch auf das Anforderungskontingent für benutzerdefinierte Schlüsselspeicher für den benutzerdefinierten Schlüsselspeicher angerechnet.
- Die Anforderungsrate für kryptografische Operationen (RSA) ist das für kryptografische Operationen freigegebene Anforderungskontingent unter Verwendung von asymmetrischen RSA-KMS-Schlüssel.

Bei einem Anforderungskontingent von 1.000 Vorgängen pro Sekunde können Sie beispielsweise 400 Verschlüsselungsanforderungen und 200 Entschlüsselungsanforderungen mit RSA-KMS-Schlüsseln, die verschlüsseln und entschlüsseln können, sowie 250 Signieranfragen und 150 Verify-Anfragen mit RSA-KMS-Schlüsseln, die signieren und verifizieren können, stellen.

- Die Anforderungsrate für kryptografische Operationen (ECC) ist das gemeinsame Anforderungskontingent für kryptografische Operationen, bei denen asymmetrische KMS-Schlüssel mit elliptischen Kurven (ECC) und asymmetrische SM-KMS-Schlüssel verwendet werden.

  Bei einem Anforderungskontingent von 1.000 Vorgängen pro Sekunde können Sie beispielsweise 400 Signieranfragen und 200 Verify-Anfragen mit ECC-KMS-Schlüsseln stellen, die signieren und verifizieren können, sowie 250 Signieranfragen und 150 Verifizierungsanfragen mit SM2 KMS-Schlüsseln, die signieren und verifizieren können.
- Das Anforderungskontingent für KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher ist das gemeinsame Anforderungskontingent für kryptografische Operationen mit KMS-Schlüsseln in einem benutzerdefinierten Schlüsselspeicher. Dieses Kontingent wird für jeden benutzerdefinierten Schlüsselspeicher separat berechnet.

  Kryptografische Operationen mit den KMS-Schlüsseln für symmetrische Verschlüsselung in einem benutzerdefinierten Schlüsselspeicher werden sowohl auf die Anforderungsrate kryptografischer Operationen (symmetrisch) für das Konto als auch auf das Anforderungskontingent für benutzerdefinierte Schlüsselspeicher für den benutzerdefinierten Schlüsselspeicher angerechnet.

Die Kontingente für unterschiedliche Schlüsseltypen werden ebenfalls unabhängig berechnet. Wenn Sie in der Region Asien-Pazifik (Singapur) beispielsweise symmetrische und asymmetrische KMS-Schlüssel verwenden, können Sie bis zu 10 000 Aufrufe pro Sekunde mit symmetrischen KMS-Schlüsseln (einschließlich HMAC-Schlüsseln) plus bis zu 500 zusätzliche Aufrufe pro Sekunde mit Ihren asymmetrischen RSA-KMS-Schlüsseln plus bis zu 300 zusätzliche Anforderungen pro Sekunde mit Ihren ECC-basierten KMS-Schlüsseln ausgeben.

API-Anforderungen in Ihrem Namen

Sie können API-Anfragen direkt oder mithilfe eines integrierten AWS Dienstes stellen, an den API-Anfragen in AWS KMS Ihrem Namen gestellt werden. Das Kontingent gilt für beide Arten von Anforderungen.

Sie speichern Daten beispielsweise in Simple Storage Service (Amazon S3) und verwenden serverseitige Verschlüsselung mit einem KMS-Schlüssel (SSE-KMS). Jedes Mal, wenn Sie ein

mit SSE-KMS verschlüsseltes S3-Objekt hoch- oder herunterladen, stellt Amazon S3 in Ihrem `GenerateDataKey` Namen eine Anfrage (für Uploads) oder `Decrypt` (für Downloads) AWS KMS an. Diese Anfragen werden auf Ihr Kontingent angerechnet. Das bedeutet, dass die Anfragen AWS KMS gedrosselt werden, wenn Sie insgesamt 5.500 (oder 10.000 oder 50.000, je nach Ihren Anforderungen AWS-Region) Uploads oder Downloads von mit SSE-KMS verschlüsselten S3-Objekten pro Sekunde überschreiten.

Kontoübergreifende Anforderungen

Wenn eine Anwendung in einer Anwendung einen KMS-Schlüssel AWS-Konto verwendet, der einem anderen Konto gehört, spricht man von einer kontoübergreifenden Anfrage. Bei kontoübergreifenden Anforderungen drosselt AWS KMS das Konto, das die Anforderungen sendet, nicht das Konto, das den KMS-Schlüssel besitzt. Wenn beispielsweise eine Anwendung in Konto A einen KMS-Schlüssel in Konto B verwendet, wird die KMS-Schlüsselnutzung nur auf die Kontingente in Konto A angerechnet.

Anforderungskontingente für benutzerdefinierte Schlüsselspeicher

AWS KMS verwaltet Anforderungskontingente für kryptografische Operationen mit den KMS-Schlüsseln in einem benutzerdefinierten Schlüsselspeicher. Diese Anforderungskontingente werden für jeden benutzerdefinierten Schlüsselspeicher separat berechnet.

Anforderungskontingent für benutzerdefinierte Schlüsselspeicher	Standardwert (Anforderungen pro Sekunde) für jeden benutzerdefinierten Schlüsselspeicher	Einstellbar
AWS CloudHSM Kontingent für Schlüsselspeicher-Anfragen	1800	Nein
Anforderungskontingent für externen Schlüsselspeicher	1800	Ja

> **Note**
>
> AWS KMS benutzerdefinierte Schlüsselspeicher-Anforderungskontingente werden nicht in der Service Quotas Quotas-Konsole angezeigt. Sie können diese Kontingente nicht mithilfe von Service-Quotas-API-Vorgängen anzeigen oder verwalten. Um eine Änderung Ihres Anforderungskontingents für externe Schlüsselspeicher zu beantragen, erstellen Sie im AWS -Support Center einen Fall.
>
> Wenn der einem AWS CloudHSM Schlüsselspeicher zugeordnete AWS CloudHSM Cluster zahlreiche Befehle verarbeitet, auch solche, die nichts mit dem benutzerdefinierten Schlüsselspeicher zu tun haben, erhalten Sie möglicherweise einen AWS KMS `ThrottlingException` lower-than-expected AT-Schlüssel. Wenn dies der Fall ist, reduzieren Sie Ihre Anforderungsrate auf AWS KMS, reduzieren Sie die damit verbundene Last oder verwenden Sie einen dedizierten AWS CloudHSM Cluster für Ihren AWS CloudHSM Schlüsselspeicher.
>
> AWS KMS meldet die Drosselung von externen Schlüsselspeicher-Anfragen in der Metrik. `ExternalKeyStoreThrottle` CloudWatch Sie können diese Metrik verwenden, um Drosselungsmuster anzuzeigen, Alarme zu erstellen und Ihr Kontingent für externe Schlüsselspeicheranforderungen anzupassen.

Eine Anforderung einer kryptografischen Operation zu einem KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher wird auf zwei Kontingente angerechnet:

- Anforderungsrate für kryptografische Operationen (symmetrisch. pro Konto)

  Anfragen für kryptografische Operationen mit KMS-Schlüsseln in einem benutzerdefinierten Schlüsselspeicher werden auf das `Cryptographic operations (symmetric) request rate`-Kontingent für jedes AWS-Konto und jede Region angerechnet. In USA Ost (Nord-Virginia) (us-east-1) AWS-Konto können beispielsweise jeweils bis zu 100.000 Anfragen pro Sekunde für KMS-Schlüssel mit symmetrischer Verschlüsselung vorliegen, einschließlich Anfragen, die einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher verwenden.

- Anforderungskontingent für benutzerdefinierte Schlüsselspeicher (pro benutzerdefiniertem Schlüsselspeicher)

  Anforderungen für kryptografische Operationen an KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher werden ebenfalls auf ein `Custom key store request quota` von 1 800 Operationen pro Sekunde. Diese Kontingente werden für jeden benutzerdefinierten

Schlüsselspeicher separat berechnet. Sie können Anfragen von mehreren Personen enthalten AWS-Konten , die KMS-Schlüssel im benutzerdefinierten Schlüsselspeicher verwenden.

Beispielsweise wird ein Verschlüsselungsvorgang für einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher (jeder Typ) in der Region USA Ost (Nord-Virginia) (US-East-1) auf das Kontingent auf `Cryptographic operations (symmetric) request rate` Kontoebene (100.000 Anfragen pro Sekunde) für sein Konto und seine Region und auf `Custom key store request quota` (1.800 Anfragen pro Sekunde) für seinen benutzerdefinierten Schlüsselspeicher angerechnet. Eine Anforderung für einen Verwaltungsvorgang, z. B. für einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher PutKeyPolicy, gilt jedoch nur für das Kontingent auf Kontoebene (15 Anfragen pro Sekunde).

Drosselung AWS KMS von Anfragen

Um sicherzustellen, dass API-Anfragen aller Kunden schnell und zuverlässig beantwortet werden AWS KMS können, werden API-Anfragen, die bestimmte Grenzen überschreiten, gedrosselt.

Eine Drosselung erfolgt, wenn eine Anfrage, die andernfalls gültig sein könnte, AWS KMS zurückgewiesen wird und ein `ThrottlingException` Fehler wie der folgende zurückgegeben wird.

```
You have exceeded the rate at which you may call KMS. Reduce the frequency of your
 calls.
(Service: AWSKMS; Status Code: 400; Error Code: ThrottlingException; Request ID: <ID>
```

AWS KMS drosselt Anfragen für die folgenden Bedingungen.

- Die Rate der Anfragen pro Sekunde überschreitet das AWS KMS Anforderungskontingent für ein Konto und eine Region.

  Wenn Benutzer in Ihrem Konto beispielsweise 1000 `DescribeKey` Anfragen in einer Sekunde einreichen, werden alle nachfolgenden `DescribeKey` Anfragen in dieser Sekunde AWS KMS gedrosselt.

  Um auf Drosselung zu reagieren, verwenden Sie eine Backoff- und Wiederholungsstrategie. In einigen Fällen wird diese Strategie automatisch für HTTP 400-Fehler implementiert. AWS SDKs
- Ein Stoß oder eine anhaltende hohe Rate von Anforderungen, um den Status desselben KMS-Schlüssels zu ändern. Diese Bedingung wird oft als „Hotkey“ bezeichnet.

Wenn beispielsweise eine Anwendung in Ihrem Konto ständig eine Vielzahl von `EnableKey` und `DisableKey` Anfragen nach demselben KMS-Schlüssel sendet, werden die Anfragen AWS KMS gedrosselt. Diese Drosselung erfolgt auch dann, wenn die Anfragen das request-per-second Anforderungslimit für die Operationen und nicht überschreiten. `EnableKey` `DisableKey`

Um auf Drosselung zu reagieren, passen Sie Ihre Anwendungslogik so an, dass nur erforderliche Anforderungen ausgeführt werden, oder es konsolidiert die Anforderungen mehrerer Funktionen.

- Anfragen für Operationen mit KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher werden möglicherweise mit einer lower-than-expected Geschwindigkeit gedrosselt, wenn der dem AWS CloudHSM Schlüsselspeicher zugeordnete AWS CloudHSM Cluster zahlreiche Befehle verarbeitet, auch solche, die nichts mit dem Schlüsselspeicher zu tun haben. AWS CloudHSM

  (AWS KMS Drosselt Anfragen für Operationen mit KMS-Schlüsseln in einem AWS CloudHSM Schlüsselspeicher nicht mehr, wenn keine PKCS #11 -Sitzungen für den Cluster verfügbar sind. AWS CloudHSM Stattdessen wird eine Meldung ausgelöst `KMSInternalException` und es wird empfohlen, dass Sie Ihre Anfrage erneut versuchen.)

Um Trends in Ihren Anforderungsraten anzuzeigen, verwenden Sie die Service-Quotas-Konsole. Sie können auch einen CloudWatchAmazon-Alarm einrichten, der Sie benachrichtigt, wenn Ihre Anforderungsrate einen bestimmten Prozentsatz eines Kontingents erreicht. Einzelheiten finden Sie im AWS Sicherheitsblog unter Verwalten Sie Ihre AWS KMS API-Anforderungsraten mithilfe von Service Quotas und Amazon CloudWatch.

Alle AWS KMS Kontingente sind anpassbar, mit Ausnahme des Ressourcenkontingents für die On-Demand-Rotation und des Kontingents für AWS CloudHSM wichtige Store-Anfragen. Informationen zur Erhöhung eines Kontingents finden Sie unter Anfordern einer Kontingenterhöhung im Service-Quotas-Benutzerhandbuch. Um eine Senkung des Kontingents zu beantragen, ein Kontingent zu ändern, das nicht in Service Quotas aufgeführt ist, oder um ein Kontingent AWS-Region in einem Bereich zu ändern, für AWS KMS den Servicekontingente nicht verfügbar sind, besuchen Sie bitte das AWS -Support Center und erstellen Sie einen Fall.

> **Note**
>
> AWS KMS benutzerdefinierte Schlüsselspeicher-Anforderungskontingente werden nicht in der Service Quotas Quotas-Konsole angezeigt. Sie können diese Kontingente nicht mithilfe von Service-Quotas-API-Vorgängen anzeigen oder verwalten. Um eine Änderung Ihres

Anforderungskontingents für externe Schlüsselspeicher zu beantragen, erstellen Sie im AWS -Support Center einen Fall.

Codebeispiele für die AWS KMS Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Verwendung AWS KMS mit einem AWS Software Development Kit (SDK) funktioniert.

Bei Grundlagen handelt es sich um Code-Beispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarios anzeigen.

Szenarien sind Code-Beispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte

Hallo AWS Key Management Service

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von AWS Key Management Service beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
 Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " +
 exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
```

```
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get
 the keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or
 to break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
```

```
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
}
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK für PHP API-Referenz.

Codebeispiele

- Grundlegende Beispiele für die AWS KMS Verwendung AWS SDKs
  - Hallo AWS Key Management Service
  - Lernen Sie die Grundlagen AWS KMS mit einem AWS SDK kennen
  - Aktionen zur AWS KMS Verwendung AWS SDKs
    - Verwendung CreateAlias mit einem AWS SDK oder CLI
    - Verwendung CreateGrant mit einem AWS SDK oder CLI
    - Verwendung CreateKey mit einem AWS SDK oder CLI
    - Verwendung Decrypt mit einem AWS SDK oder CLI
    - Verwendung DeleteAlias mit einem AWS SDK oder CLI
    - Verwendung DescribeKey mit einem AWS SDK oder CLI
    - Verwendung DisableKey mit einem AWS SDK oder CLI
    - Verwendung EnableKey mit einem AWS SDK oder CLI
    - Verwendung EnableKeyRotation mit einem AWS SDK oder CLI
    - Verwendung Encrypt mit einem AWS SDK oder CLI
    - Verwendung GenerateDataKey mit einem AWS SDK oder CLI
    - Verwendung GenerateDataKeyWithoutPlaintext mit einem AWS SDK oder CLI
    - Verwendung GenerateRandom mit einem AWS SDK oder CLI
    - Verwendung GetKeyPolicy mit einem AWS SDK oder CLI
    - Verwendung ListAliases mit einem AWS SDK oder CLI
    - Verwendung ListGrants mit einem AWS SDK oder CLI
    - Verwendung ListKeyPolicies mit einem AWS SDK oder CLI
    - Verwendung ListKeys mit einem AWS SDK oder CLI
    - Verwendung PutKeyPolicy mit einem AWS SDK oder CLI

    - Verwendung ReEncrypt mit einem AWS SDK oder CLI
    - Verwendung RetireGrant mit einem AWS SDK oder CLI
    - Verwendung RevokeGrant mit einem AWS SDK oder CLI
    - Verwendung ScheduleKeyDeletion mit einem AWS SDK oder CLI
    - Verwendung Sign mit einem AWS SDK oder CLI
    - Verwendung TagResource mit einem AWS SDK oder CLI
    - Verwendung UpdateAlias mit einem AWS SDK oder CLI
    - Verwendung Verify mit einem AWS SDK oder CLI
- Szenarien für die AWS KMS Verwendung AWS SDKs
  - Arbeiten Sie mit DynamoDB-Tabellenverschlüsselung mit Version 2 AWS Command Line Interface

Grundlegende Beispiele für die AWS KMS Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Key Management Service with verwendet AWS SDKs werden.

Beispiele

- Hallo AWS Key Management Service
- Lernen Sie die Grundlagen AWS KMS mit einem AWS SDK kennen
- Aktionen zur AWS KMS Verwendung AWS SDKs
  - Verwendung CreateAlias mit einem AWS SDK oder CLI
  - Verwendung CreateGrant mit einem AWS SDK oder CLI
  - Verwendung CreateKey mit einem AWS SDK oder CLI
  - Verwendung Decrypt mit einem AWS SDK oder CLI
  - Verwendung DeleteAlias mit einem AWS SDK oder CLI
  - Verwendung DescribeKey mit einem AWS SDK oder CLI
  - Verwendung DisableKey mit einem AWS SDK oder CLI
  - Verwendung EnableKey mit einem AWS SDK oder CLI
  - Verwendung EnableKeyRotation mit einem AWS SDK oder CLI
  - Verwendung Encrypt mit einem AWS SDK oder CLI
  - Verwendung GenerateDataKey mit einem AWS SDK oder CLI

- Verwendung GenerateDataKeyWithoutPlaintext mit einem AWS SDK oder CLI
- Verwendung GenerateRandom mit einem AWS SDK oder CLI
- Verwendung GetKeyPolicy mit einem AWS SDK oder CLI
- Verwendung ListAliases mit einem AWS SDK oder CLI
- Verwendung ListGrants mit einem AWS SDK oder CLI
- Verwendung ListKeyPolicies mit einem AWS SDK oder CLI
- Verwendung ListKeys mit einem AWS SDK oder CLI
- Verwendung PutKeyPolicy mit einem AWS SDK oder CLI
- Verwendung ReEncrypt mit einem AWS SDK oder CLI
- Verwendung RetireGrant mit einem AWS SDK oder CLI
- Verwendung RevokeGrant mit einem AWS SDK oder CLI
- Verwendung ScheduleKeyDeletion mit einem AWS SDK oder CLI
- Verwendung Sign mit einem AWS SDK oder CLI
- Verwendung TagResource mit einem AWS SDK oder CLI
- Verwendung UpdateAlias mit einem AWS SDK oder CLI
- Verwendung Verify mit einem AWS SDK oder CLI

Hallo AWS Key Management Service

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von AWS Key Management Service beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
```

```
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
 Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
```

```
                    System.err.println("Error occurred: " +
 exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get
 the keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or
 to break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
```

```
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
}
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK für PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Lernen Sie die Grundlagen AWS KMS mit einem AWS SDK kennen

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen eines KMS-Schlüssels.
- Listen Sie die KMS-Schlüssel für Ihr Konto auf und informieren Sie sich über sie.
- Aktivieren und Deaktivieren von KMS-Schlüsseln
- Generieren Sie einen symmetrischen Datenschlüssel, der für die clientseitige Verschlüsselung verwendet werden kann.
- Generieren Sie einen asymmetrischen Schlüssel, der zum digitalen Signieren von Daten verwendet wird.
- Kennzeichnen Sie Schlüssel.
- Löschen Sie KMS-Schlüssel.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String accountId = "";
```

```
   private static final Logger logger =
LoggerFactory.getLogger(KMSScenario.class);

   static KMSActions kmsActions = new KMSActions();

   static Scanner scanner = new Scanner(System.in);

   static String aliasName = "alias/dev-encryption-key";

   public static void main(String[] args) {
       final String usage = """
           Usage: <granteePrincipal>

           Where:
              granteePrincipal - The principal (user, service account, or group)
to whom the grant or permission is being given.
           """;

       if (args.length != 1) {
           logger.info(usage);
           return;
       }
       String granteePrincipal = args[0];
       String policyName = "default";

       accountId = kmsActions.getAccountId();
       String keyDesc = "Created by the AWS KMS API";

       logger.info(DASHES);
       logger.info("""
           Welcome to the AWS Key Management SDK Basics scenario.

           This program demonstrates how to interact with AWS Key Management
using the AWS SDK for Java (v2).
           The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
           and manage AWS KMS keys and control their use across a wide range of
AWS services and applications.
           KMS provides a centralized and unified approach to managing
encryption keys, making it easier to meet your
           data protection and regulatory compliance requirements.

           This Basics scenario creates two key types:
```

```
            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create a symmetric KMS key\n");
        logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId;
        try {
            CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
            targetKeyId = futureKeyId.join();
            logger.info("A symmetric key was successfully created " +
targetKeyId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            2. Enable a KMS key

            By default, when the SDK creates an AWS key, it is enabled. The next
bit of code checks to
            determine if the key is enabled.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled;
        try {
            CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
            isEnabled = futureIsKeyEnabled.join();
            logger.info("Is the key enabled? {}", isEnabled);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
```

```
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt
data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
        SdkBytes encryptedData;
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
```

```
            """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            By default, when the SDK enables automatic rotation of a KMS key,
            KMS rotates the key material of the KMS key one year (approximately
365 days) from the enable date and every year
            thereafter.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            7. Create a grant

            A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
            It also can allow them to view a KMS key (DescribeKey) and create and
manage grants.
            When authorizing access to a KMS key, grants are considered along
with key policies and IAM policies.
            """);

        waitForInputToContinue(scanner);
        String grantId = null;
```

```
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
```

```
            The revocation of a grant immediately removes the permissions and
access that the grant had provided.
            This means that any principal (user, role, or service) that was
granted access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            Lets decrypt the data that was encrypted in an early step.
            The code uses the same key to decrypt the string that we encrypted
earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptedData = "";
        try {
```

```
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
            A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
            access to KMS keys. Every KMS key must have exactly one key policy.
The statements in the key policy
            determine who has permission to use the KMS key and how they can use
it.
            You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
            must have a key policy.

            By default, when you create a key by using the SDK, a policy is
created that
            gives the AWS account that owns the KMS key full access to the KMS
key.

            Let's try to replace the automatically created policy with the
following policy.

                "Version": "2012-10-17",
                "Statement": [{
                "Effect": "Allow",
```

```
                "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make
sure it exists.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             Signing your data with an AWS key can provide several benefits that
make it an attractive option
             for your data signing needs. By using an AWS KMS key, you can
leverage the
             security controls and compliance features provided by AWS,
             which can help you meet various regulatory requirements and enhance
the overall security posture
             of your organization.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
            boolean success = future.join();
            if (success) {
                logger.info("Sign and verify data operation succeeded.");
            } else {
                logger.error("Sign and verify data operation failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("14. Tag your symmetric KMS Key\n");
        logger.info("""
            By using tags, you can improve the overall management, security, and
governance of your
            KMS keys, making it easier to organize, track, and control access to
your encrypted data within
            your AWS environment
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
kmsActions.tagKMSKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }

    // Deletes KMS resources with user input.
    private static void requestDeleteResources(String aliasName, String
targetKeyId) {
        logger.info(DASHES);
        logger.info("15. Schedule the deletion of the KMS key\n");
        logger.info("""
            By default, KMS applies a waiting period of 30 days,
```

```
            but you can specify a waiting period of 7-30 days. When this
 operation is successful,
            the key state of the KMS key changes to PendingDeletion and the key
 can't be used in any
            cryptographic operations. It remains in this state for the duration
 of the waiting period.

            Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted,
            all data that was encrypted under the KMS key is unrecoverable.
            """);
        logger.info("Would you like to delete the Key Management resources? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS KMS resources.");
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
 kmsActions.deleteSpecificAliasAsync(aliasName);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
 code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
 rt.getMessage());
                }
            }
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
 kmsActions.disableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
 code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
```

```
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteKey(String targetKeyId) {
        try {
            CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
```

```
            }
        }

        try {
            CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteAliasName(String aliasName) {
        try {
            CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
```

```
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Definieren Sie eine Klasse, die KMS-Aktionen umschließt.

```
public class KMSActions {
    private static final Logger logger =
 LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async
 client, with the following configurations:
     * <ul>
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
 creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
```

```
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
```

```
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key
state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
```

```
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and
key ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
```

```
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }

    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of
aliases has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }

    /**
     * Enables key rotation asynchronously for the specified key ID.
     *
     * @param keyId the ID of the key for which to enable key rotation
     * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
     * @throws RuntimeException if there was an error enabling key rotation,
either due to a KMS exception or an unexpected error
     */
    public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
```

```
        EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to enable key rotation: "
+ kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }

    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID
of the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
```

```
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }

    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null
if the operation succeeded, or will throw a {@link RuntimeException} if the
operation failed
     * @throws RuntimeException if there was an error listing the grants, either
due to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();
```

```
        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }

    /**
     * Revokes a grant for the specified AWS KMS key asynchronously.
     *
     * @param keyId   The ID or key ARN of the AWS KMS key.
     * @param grantId The identifier of the grant to be revoked.
     * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
     *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
     *         if the operation is successful, or with a {@code null} value if an
error occurs.
     */
    public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
```

```
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }


    /**
     * Asynchronously decrypts the given encrypted data using the specified key
ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
```

```
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId       the ID of the KMS key to replace the policy for
     * @param policyName  the name of the policy to be replaced
     * @param accountId   the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean
indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = """
    {
      "Version": "2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::%s:root"},
        "Action": "kms:*",
        "Resource": "*"
      }]
    }
    """.formatted(accountId);

        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
```

```
            .policyName(policyName)
            .policy(policy)
            .build();

        // First, get the current policy to check if it exists
        return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
            .thenCompose(response -> {
                logger.info("Current policy exists. Replacing it...");
                return getAsyncClient().putKeyPolicy(keyPolicyRequest);
            })
            .thenApply(putPolicyResponse -> {
                logger.info("The key policy has been replaced.");
                return true;
            })
            .exceptionally(throwable -> {
                if (throwable.getCause() instanceof LimitExceededException) {
                    logger.error("Cannot replace policy, as only one policy is
allowed per key.");
                    return false;
                }
                throw new RuntimeException("Error replacing policy", throwable);
            });
    }


    /**
     * Asynchronously retrieves the key policy for the specified key ID and
policy name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the
policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
```

```
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
     *     <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();
```

```
                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes =
signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging
operation is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
```

```
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key",
throwable);
            });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
            });
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS)
key.
```

```
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key
has been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
    }

    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
     * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted
using the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when
deleting KMS keys.</p>
     *
     * @param keyId the ID of the KMS key to delete
     * @return a {@link CompletableFuture} that completes when the key deletion
is scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();
```

```
        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for
 key ID: " + keyId, throwable);
            });
    }


    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign

- TagResource

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "\n";
        echo "--------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management Service using
 the AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service
 that allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys,
 making it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;
        echo "--------------------------------------\n";
        $this->pressEnter();

        $this->kmsClient = new KmsClient([]);
        // Initialize the KmsService class with the client. This allows you to
 override any defaults in the client before giving it to the service class.
        $this->kmsService = new KmsService($this->kmsClient);
```

```
        // 1. Create a symmetric KMS key.
        echo "\n";
        echo "1. Create a symmetric KMS key.\n";
        echo "First, we will create a symmetric KMS key that is used to encrypt
 and decrypt data by invoking createKey().\n";
        $this->pressEnter();

        $key = $this->kmsService->createKey();
        $this->resources['symmetricKey'] = $key['KeyId'];
        echo "Created a customer key with ARN {$key['Arn']}.\n";
        $this->pressEnter();

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code
 checks to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

        $keyInfo = $this->kmsService->describeKey($key['KeyId']);
        if(!$keyInfo['Enabled']){
            echo "The key was not enabled, so we will enable it.\n";
            $this->pressEnter();
            $this->kmsService->enableKey($key['KeyId']);
            echo "The key was successfully enabled.\n";
        }else{
            echo "The key was already enabled, so there was no need to enable it.
\n";
        }
        $this->pressEnter();

        // 3. Encrypt data using the symmetric KMS key.
        echo "\n";
        echo "3. Encrypt data using the symmetric KMS key.\n";
        echo "One of the main uses of symmetric keys is to encrypt and decrypt
 data.\n";
        echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
 SYMMETRIC_DEFAULT encryption algorithm.\n";
        $this->pressEnter();
        $text = "Hello, AWS KMS!";
        $encryption = $this->kmsService->encrypt($key['KeyId'], $text);
```

```
        echo "The plaintext data was successfully encrypted with the algorithm:
 {$encryption['EncryptionAlgorithm']}.\n";
        $this->pressEnter();

        // 4. Create an alias.
        echo "\n";
        echo "4. Create an alias.\n";
        $aliasInput = testable_readline("Please enter an alias prefixed with
 \"alias/\" or press enter to use a default value: ");
        if($aliasInput == ""){
            $aliasInput = "alias/dev-encryption-key";
        }
        $this->kmsService->createAlias($key['KeyId'], $aliasInput);
        $this->resources['alias'] = $aliasInput;
        echo "The alias \"$aliasInput\" was successfully created.\n";
        $this->pressEnter();

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
        echo "6. Enable automatic rotation of the KMS key.\n";
        echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
 from the enable date and every year
thereafter.";
        $this->pressEnter();
        $this->kmsService->enableKeyRotation($key['KeyId']);
        echo "The key's rotation was successfully set for key:
 {$key['KeyId']}\n";
        $this->pressEnter();
```

```
        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
 grants.
When authorizing access to a KMS key, grants are considered along with key
 policies and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon
 Web Services accounts, IAM users, IAM roles, federated users, and assumed
 role users. For help with the ARN syntax for a principal, see IAM ARNs in the
 Identity and Access Management User Guide. \nTo skip this step, press enter
 without any other values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
 $operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

        // 8. List grants for the KMS key.
        if($granteeARN){
            echo "8. List grants for the KMS key.\n\n";
            $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
            foreach($grantsPaginator as $page){
                foreach($page['Grants'] as $grant){
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

        // 9. Revoke the grant.
```

```
        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted
 earlier in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
 $encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are
 the primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in
 the key policy determine who has permission to use the KMS key and how they can
 use it.\n";
        echo " You can also use IAM policies and grants to control access to the
 KMS key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
```

```
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

        // 12. Retrieve the key policy.
        echo "\n";
        echo "12. Retrieve the key policy.\n";
        echo "Let's get some information about the new policy and print it to the
 screen.\n";
        $this->pressEnter();
        $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
        echo "We got the info! Here is the policy: \n";
        echo $policyInfo['Policy'] . "\n";
        $this->pressEnter();

        // 13. Create an asymmetric KMS key and sign data.
        echo "\n";
        echo "13. Create an asymmetric KMS key and sign data.\n";
        echo "Signing your data with an AWS key can provide several benefits that
 make it an attractive option for your data signing needs.\n";
        echo "By using an AWS KMS key, you can leverage the security controls and
 compliance features provided by AWS, which can help you meet various regulatory
 requirements and enhance the overall security posture of your organization.\n";
        echo "First we'll create the asymmetric key.\n";
        $this->pressEnter();
        $keySpec = "RSA_2048";
        $keyUsage = "SIGN_VERIFY";
        $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
        $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text,
 $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
 $sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
```

```
        $this->pressEnter();

        // 14. Tag the symmetric KMS key.
        echo "\n";
        echo "14. Tag the symmetric KMS key.\n";
        echo "By using tags, you can improve the overall management, security,
 and governance of your KMS keys, making it easier to organize, track, and
 control access to your encrypted data within your AWS environment.\n";
        echo "Let's tag our symmetric key as Environment->Production\n";
        $this->pressEnter();
        $this->kmsService->tagResource($key['KeyId'], [
            [
                'TagKey' => "Environment",
                'TagValue' => "Production",
            ],
        ]);
        echo "The key was successfully tagged!\n";
        $this->pressEnter();

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
 specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
 changes to PendingDeletion and the key can't be used in any cryptographic
 operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n
\n";

        echo "Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted, all data that was encrypted under the KMS
 key is unrecoverable.\n\n";

        $cleanUp = testable_readline("Would you like to delete the resources
 created during this scenario, including the keys? (y/n): ");
        if($cleanUp == "Y" || $cleanUp == "y"){
            $this->cleanUp();
        }

        echo
 "-------------------------------------------------------------------------------
\n";
        echo "This concludes the AWS Key Management SDK Basics scenario\n";
```

```
        echo
 "--------------------------------------------------------------------------------
\n";



namespace Kms;

use Aws\Kms\Exception\KmsException;
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

    /***
     * @param KmsClient|null $client
     * @param bool $verbose
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }


    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
 $description = "Created by the SDK for PHP")
    {
```

```
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
 more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
 = "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is
 not enabled.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
 where pagination should next begin is not valid.\n";
            }
```

```
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name
 is not valid.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
 $operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
```

```
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token
is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
```

```
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
```

```
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity
 or resource could not be found.\n";
            }
            throw $caught;
        }
    }


    /***
     * @param string $keyId
     * @return Result
     */
    public function getKeyPolicy(string $keyId)
    {
        try {
            return $this->client->getKeyPolicy([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $rotationPeriodInDays
     * @return void
     */
    public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
365)
    {
        try{
            $this->client->enableKeyRotation([
                'KeyId' => $keyId,
                'RotationPeriodInDays' => $rotationPeriodInDays,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature,
 string $signingAlgorithm)
    {
        try {
            $result = $this->client->verify([
                'KeyId' => $keyId,
                'Message' => $message,
                'Signature' => $signature,
                'SigningAlgorithm' => $signingAlgorithm,
            ]);
```

```
            return $result['SignatureValid'];
        }catch(KmsException $caught){
            echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


}
```

- API-Details finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KMSScenario:
    """Runs an interactive scenario that shows how to get started with KMS."""

    def __init__(
        self,
        key_manager: KeyManager,
        key_encryption: KeyEncrypt,
        alias_manager: AliasManager,
        grant_manager: GrantManager,
        key_policy: KeyPolicy,
    ):
        self.key_manager = key_manager
        self.key_encryption = key_encryption
        self.alias_manager = alias_manager
        self.grant_manager = grant_manager
        self.key_policy = key_policy
        self.key_id = ""
        self.alias_name = ""
        self.asymmetric_key_id = ""

    def kms_scenario(self):
        key_description = "Created by the AWS KMS API"

        print(DASHES)
        print(
            """
Welcome to the AWS Key Management SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management using the AWS
 SDK for Python (Boto3).
The AWS Key Management Service (KMS) is a secure and highly available service
 that allows you to create
```

```
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys,
 making it easier to meet your
data protection and regulatory compliance requirements.

This Basics scenario creates two key types:

- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...
        """
        )
        q.ask("Press Enter to continue...")

        print(DASHES)
        print(f"1. Create a symmetric KMS key\n")
        print(
            f"First, the program will creates a symmetric KMS key that you can
 used to encrypt and decrypt data."
        )
        q.ask("Press Enter to continue...")
        self.key_id = self.key_manager.create_key(key_description)["KeyId"]
        print(f"A symmetric key was successfully created {self.key_id}.")
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(
            """
2. Enable a KMS key

By default, when the SDK creates an AWS key, it is enabled. The next bit of code
 checks to
determine if the key is enabled.
        """
        )
        q.ask("Press Enter to continue...")
        is_enabled = self.is_key_enabled(self.key_id)
        print(f"Is the key enabled? {is_enabled}")
        if not is_enabled:
            self.key_manager.enable_key(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"3. Encrypt data using the symmetric KMS key")
```

```
        plain_text = "Hello, AWS KMS!"
        print(
            f"""
One of the main uses of symmetric keys is to encrypt and decrypt data.
Next, the code encrypts the string "{plain_text}" with the SYMMETRIC_DEFAULT
 encryption algorithm.
        """
        )
        q.ask("Press Enter to continue...")
        encrypted_text = self.key_encryption.encrypt(self.key_id, plain_text)
        print(DASHES)
        print(f"4. Create an alias")
        print(
            """
Now, the program will create an alias for the KMS key. An alias is a friendly
 name that you
can associate with a KMS key. The alias name should be prefixed with 'alias/'.
        """
        )
        alias_name = q.ask("Enter an alias name: ", q.non_empty)
        self.alias_manager.create_alias(self.key_id, alias_name)
        print(f"{alias_name} was successfully created.")
        self.alias_name = alias_name
        print(DASHES)
        print(f"5. List all of your aliases")
        q.ask("Press Enter to continue...")
        self.alias_manager.list_aliases(10)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"6. Enable automatic rotation of the KMS key")
        print(
            """

By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
 from the enable date and every year
thereafter.
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.enable_key_rotation(self.key_id)
        print(DASHES)
        print(f"Key rotation has been enabled for key with id {self.key_id}")
        print(
```

```
            """
7. Create a grant

A grant is a policy instrument that allows Amazon Web Services principals to use
 KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
 grants.
When authorizing access to a KMS key, grants are considered along with key
 policies and IAM policies.
        """
        )
        print(
            """
To create a grant you must specify a account_id. To specify the grantee
 account_id, use the Amazon Resource Name (ARN)
of an AWS account_id. Valid principals include AWS accounts, IAM users, IAM
 roles, federated users,
and assumed role users.
        """
        )
        account_id = q.ask(
            "Enter an account_id, or press enter to skip creating a grant... "
        )
        grant = None
        if account_id != "":
            grant = self.grant_manager.create_grant(
                self.key_id,
                account_id,
                [
                    "Encrypt",
                    "Decrypt",
                    "DescribeKey",
                ],
            )
            print(f"Grant created successfully with ID: {grant['GrantId']}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(DASHES)
        print(f"8. List grants for the KMS key")
        q.ask("Press Enter to continue...")
        self.grant_manager.list_grants(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
```

```
        print(f"9. Revoke the grant")
        print(
            """
The revocation of a grant immediately removes the permissions and access that the
 grant had provided.
This means that any account_id (user, role, or service) that was granted access
 to perform specific
KMS operations on a KMS key will no longer be able to perform those operations.
        """
        )
        q.ask("Press Enter to continue...")

        if grant is not None:
            self.grant_manager.revoke_grant(self.key_id, grant["GrantId"])
            print(f"Grant ID: {grant['GrantId']} was successfully revoked!")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"10. Decrypt the data\n")
        print(
            """
Lets decrypt the data that was encrypted in an early step.
The code uses the same key to decrypt the string that we encrypted earlier in the
 program.
        """
        )
        q.ask("Press Enter to continue...")
        decrypted_data = self.key_encryption.decrypt(self.key_id, encrypted_text)
        print(f"Data decrypted successfully for key ID: {self.key_id}")
        print(f"Decrypted data: {decrypted_data}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"11. Replace a key policy\n")
        print(
            """
A key policy is a resource policy for a KMS key. Key policies are the primary way
 to control
access to KMS keys. Every KMS key must have exactly one key policy. The
 statements in the key policy
determine who has permission to use the KMS key and how they can use it.
You can also use IAM policies and grants to control access to the KMS key, but
 every KMS key
must have a key policy.
```

```
By default, when you create a key by using the SDK, a policy is created that
gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.
{
"Version": "2012-10-17",
"Statement": [{
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::0000000000:root"},
"Action": "kms:*",
"Resource": "*"
}]
}
        """
        )
        account_id = q.ask("Enter your account ID or press enter to skip: ")
        if account_id != "":
            policy = {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": f"arn:aws:iam::{account_id}:root"},
                        "Action": "kms:*",
                        "Resource": "*",
                    }
                ],
            }

            self.key_policy.set_new_policy(self.key_id, policy)
            print("Key policy replacement succeeded.")
            q.ask("Press Enter to continue...")
        else:
            print("Skipping replacing the key policy.")

        print(DASHES)
        print(f"12. Get the key policy\n")
        print(
            f"The next bit of code that runs gets the key policy to make sure it
 exists."
        )
        q.ask("Press Enter to continue...")
        policy = self.key_policy.get_policy(self.key_id)
```

```
        print(f"The key policy is: {policy}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"13. Create an asymmetric KMS key and sign your data\n")
        print(
            """
        Signing your data with an AWS key can provide several benefits that make
it an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage
the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
        """
        )
        q.ask("Press Enter to continue...")
        print(f"Sign and verify data operation succeeded.")
        self.asymmetric_key_id = self.key_manager.create_asymmetric_key()
        message = "Here is the message that will be digitally signed"
        signature = self.key_encryption.sign(self.asymmetric_key_id, message)
        if self.key_encryption.verify(self.asymmetric_key_id, message,
signature):
            print("Signature verification succeeded.")
        else:
            print("Signature verification failed.")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"14. Tag your symmetric KMS Key\n")
        print(
            """
        By using tags, you can improve the overall management, security, and
governance of your
        KMS keys, making it easier to organize, track, and control access to your
encrypted data within
        your AWS environment
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.tag_resource(self.key_id, "Environment", "Production")
        self.clean_up()
```

```
    def is_key_enabled(self, key_id: str) -> bool:
        """
        Check if the key is enabled or not.

        :param key_id: The key to check.
        :return: True if the key is enabled, otherwise False.
        """
        response = self.key_manager.describe_key(key_id)
        return response["Enabled"] is True

    def clean_up(self):
        """
        Delete resources created by this scenario.
        """
        if self.alias_name != "":
            print(f"Deleting the alias {self.alias_name}.")
            self.alias_manager.delete_alias(self.alias_name)
        window = 7  # The window in days for a scheduled deletion.
        if self.key_id != "":
            print(
                """
Warning:
Deleting a KMS key is a destructive and potentially dangerous operation. When a
 KMS key is deleted,
all data that was encrypted under the KMS key is unrecoverable.
                """
            )
            if q.ask(
                f"Do you want to delete the key with ID {self.key_id} (y/n)?",
                q.is_yesno,
            ):
                print(
                    f"The key {self.key_id} will be deleted with a window of
 {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.key_id, window)
                self.key_id = ""

        if self.asymmetric_key_id != "":
            if q.ask(
                f"Do you want to delete the asymmetric key with ID
 {self.asymmetric_key_id} (y/n)?",
                q.is_yesno,
```

```
            ):
                print(
                    f"The key {self.asymmetric_key_id} will be deleted with a
 window of {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.asymmetric_key_id, window)
                self.asymmetric_key_id = ""


if __name__ == "__main__":
    kms_scenario = None
    try:
        kms_client = boto3.client("kms")
        a_key_manager = KeyManager(kms_client)
        a_key_encrypt = KeyEncrypt(kms_client)
        an_alias_manager = AliasManager(kms_client)
        a_grant_manager = GrantManager(kms_client)
        a_key_policy = KeyPolicy(kms_client)
        kms_scenario = KMSScenario(
            key_manager=a_key_manager,
            key_encryption=a_key_encrypt,
            alias_manager=an_alias_manager,
            grant_manager=a_grant_manager,
            key_policy=a_key_policy,
        )
        kms_scenario.kms_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo!")
        if kms_scenario is not None:
            kms_scenario.clean_up()
```

Wrapper-Klasse und Methoden für die KMS-Schlüsselverwaltung.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
```

```
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
```

```
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_asymmetric_key(self) -> str:
        """
        Creates an asymmetric key in AWS KMS for signing messages.

        :return: The ID of the created key.
        """
        try:
            key = self.kms_client.create_key(
                KeySpec="RSA_2048", KeyUsage="SIGN_VERIFY", Origin="AWS_KMS"
            )["KeyMetadata"]
            self.created_keys.append(key)
            return key["KeyId"]
        except ClientError as err:
            logger.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
```

```
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
```

```
            raise
```

Wrapper-Klasse und Methoden für KMS-Schlüsselaliase.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
```

```
                raise


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)?
"
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
```

```
                err.response["Error"]["Message"],
            )
            raise
```

Wrapper-Klasse und Methoden für die KMS-Schlüsselverschlüsselung.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
 Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
 {response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
```

```
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id,
CiphertextBlob=cipher_text)[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
```

```
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Wrapper-Klasse und Methoden für die Gewährung von KMS-Schlüsseln.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

```
    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

Wrapper-Klasse und Methoden für KMS-Schlüsselrichtlinien.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_new_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
 existing
        policy, so care is taken to add a statement to the existing list of
 statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: A new key policy. The key policy must allow the calling
 principal to make a subsequent
                       PutKeyPolicy request on the KMS key. This reduces the risk
 that the KMS key becomes unmanageable
        """

        try:
            self.kms_client.put_key_policy(KeyId=key_id,
 Policy=json.dumps(policy))
        except ClientError as err:
            logger.error(
                "Couldn't set policy for key %s. Here's why %s",
                key_id,
                err.response["Error"]["Message"],
```

```
            )
            raise



    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey

- EnableKey
- Encrypt
- GetKeyPolicy
- ListAliases
- ListGrants
- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen zur AWS KMS Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie einzelne AWS KMS Aktionen mit ausführen können AWS SDKs. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die AWS KMS API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die AWS KMS Verwendung AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Key Management Service -API-Referenz.

Beispiele

- Verwendung CreateAlias mit einem AWS SDK oder CLI
- Verwendung CreateGrant mit einem AWS SDK oder CLI
- Verwendung CreateKey mit einem AWS SDK oder CLI
- Verwendung Decrypt mit einem AWS SDK oder CLI
- Verwendung DeleteAlias mit einem AWS SDK oder CLI

- Verwendung DescribeKey mit einem AWS SDK oder CLI
- Verwendung DisableKey mit einem AWS SDK oder CLI
- Verwendung EnableKey mit einem AWS SDK oder CLI
- Verwendung EnableKeyRotation mit einem AWS SDK oder CLI
- Verwendung Encrypt mit einem AWS SDK oder CLI
- Verwendung GenerateDataKey mit einem AWS SDK oder CLI
- Verwendung GenerateDataKeyWithoutPlaintext mit einem AWS SDK oder CLI
- Verwendung GenerateRandom mit einem AWS SDK oder CLI
- Verwendung GetKeyPolicy mit einem AWS SDK oder CLI
- Verwendung ListAliases mit einem AWS SDK oder CLI
- Verwendung ListGrants mit einem AWS SDK oder CLI
- Verwendung ListKeyPolicies mit einem AWS SDK oder CLI
- Verwendung ListKeys mit einem AWS SDK oder CLI
- Verwendung PutKeyPolicy mit einem AWS SDK oder CLI
- Verwendung ReEncrypt mit einem AWS SDK oder CLI
- Verwendung RetireGrant mit einem AWS SDK oder CLI
- Verwendung RevokeGrant mit einem AWS SDK oder CLI
- Verwendung ScheduleKeyDeletion mit einem AWS SDK oder CLI
- Verwendung Sign mit einem AWS SDK oder CLI
- Verwendung TagResource mit einem AWS SDK oder CLI
- Verwendung UpdateAlias mit einem AWS SDK oder CLI
- Verwendung Verify mit einem AWS SDK oder CLI

Verwendung **`CreateAlias`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAlias` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Creates an alias for an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class CreateAlias
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The alias name must start with alias/ and can be
        // up to 256 alphanumeric characters long.
        var aliasName = "alias/ExampleAlias";

        // The value supplied as the TargetKeyId can be either
        // the key ID or key Amazon Resource Name (ARN) of the
        // AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new CreateAliasRequest
        {
            AliasName = aliasName,
            TargetKeyId = keyId,
        };

        var response = await client.CreateAliasAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                Console.WriteLine($"Alias, {aliasName}, successfully created.");
            }
            else
            {
                Console.WriteLine($"Could not create alias.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateAliasin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um einen Alias für einen KMS-Schlüssel zu erstellen

Der folgende `create-alias` Befehl erstellt einen Alias, der `example-alias` nach dem KMS-Schlüssel benannt ist, der durch die Schlüssel-ID identifiziert wird`1234abcd-12ab-34cd-56ef-1234567890ab`.

Aliasnamen müssen mit beginnen`alias/`. Verwenden Sie keine Aliasnamen, die mit `alias/aws` beginnen. Diese sind für die Verwendung durch reserviert AWS.

```
aws kms create-alias \
    --alias-name alias/example-alias \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Dieser Befehl gibt keine Ausgabe zurück. Verwenden Sie den `list-aliases` Befehl, um den neuen Alias zu sehen.

Weitere Informationen finden Sie unter Verwenden von Aliasen im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie unter CreateAlias AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
               if (exception instanceof ResourceExistsException) {
                   logger.info("Alias [{}] already exists. Moving on...",
aliasName);
               } else if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
               }
           }
```

```
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Einzelheiten zur API finden Sie CreateAliasin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- Einzelheiten zur API finden Sie CreateAliasin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name
is not valid.";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie CreateAliasin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
```

```
        else:
            logger.error(
                "Couldn't encrypt text. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateAliasin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`CreateGrant`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateGrant` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identity that is given permission to perform the operations
        // specified in the grant.
```

```
            var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

            // The identifier of the AWS KMS key to which the grant applies. You
            // can use the key ID or the Amazon Resource Name (ARN) of the KMS
key.
            var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";

            var request = new CreateGrantRequest
            {
                GranteePrincipal = grantee,
                KeyId = keyId,

                // A list of operations that the grant allows.
                Operations = new List<string>
                {
                    "Encrypt",
                    "Decrypt",
                },
            };

            var response = await client.CreateGrantAsync(request);

            string grantId = response.GrantId; // The unique identifier of the
grant.
            string grantToken = response.GrantToken; // The grant token.

            Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
        }
    }
```

- Einzelheiten zur API finden Sie CreateGrantin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um einen Zuschuss zu erstellen

Im folgenden `create-grant` Beispiel wird ein Grant erstellt, der es dem `exampleUser` Benutzer ermöglicht, den `decrypt` Befehl für den `1234abcd-12ab-34cd-56ef-1234567890ab` KMS-Beispielschlüssel zu verwenden.

Der ausscheidende Schulleiter ist die `adminRole` Rolle. Die Gewährung verwendet die `EncryptionContextSubset` Grant-Beschränkung, um diese Berechtigung nur dann zuzulassen, wenn der Verschlüsselungskontext in der `decrypt` Anforderung das `"Department": "IT"` Schlüssel-Wert-Paar enthält.

```
aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::123456789012:user/exampleUser \
    --operations Decrypt \
    --constraints EncryptionContextSubset={Department=IT} \
    --retiring-principal arn:aws:iam::123456789012:role/adminRole
```

Ausgabe:

```
{
    "GrantId":
 "1a2b3c4d2f5e69f440bae30eaec9570bb1fb7358824f9ddfa1aa5a0dab1a59b2",
    "GrantToken": "<grant token here>"
}
```

Verwenden Sie den Befehl, um detaillierte Informationen zur Gewährung anzuzeigen. `list-grants`

Weitere Informationen finden Sie unter Grants in AWS KMS im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie CreateGrantin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
    *
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID
of the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
    */
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
       List<GrantOperation> grantPermissions = List.of(
           GrantOperation.ENCRYPT,
           GrantOperation.DECRYPT,
           GrantOperation.DESCRIBE_KEY
       );

       CreateGrantRequest grantRequest = CreateGrantRequest.builder()
           .keyId(keyId)
           .name("grant1")
           .granteePrincipal(granteePrincipal)
           .operations(grantPermissions)
           .build();

       CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex == null) {
               logger.info("Grant created successfully with ID: " +
response.grantId());
           } else {
               if (ex instanceof KmsException kmsEx) {
                   throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
               }
           }
       });
```

```
        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }
```

- Einzelheiten zur API finden Sie CreateGrantin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- Einzelheiten zur API finden Sie CreateGrantin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /***
    * @param string $keyId
    * @param string $granteePrincipal
    * @param array $operations
    * @param array $grantTokens
    * @return Result
    */
   public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
   {
       $args = [
           'KeyId' => $keyId,
           'GranteePrincipal' => $granteePrincipal,
           'Operations' => $operations,
       ];
       if($grantTokens){
           $args['GrantTokens'] = $grantTokens;
       }
       try{
           return $this->client->createGrant($args);
       }catch(KmsException $caught){
           if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
               echo "The request was rejected because the specified grant token
is not valid.\n";
           }
           throw $caught;
       }
   }
```

- Einzelheiten zur API finden Sie CreateGrantin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
```

```
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateGrantin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`CreateKey`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateKey` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erlernen der Grundlagen
- Arbeiten Sie mit Tabellenverschlüsselung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// Shows how to create a new AWS Key Management Service (AWS KMS)
    /// key.
    /// </summary>
    public class CreateKey
    {
        public static async Task Main()
        {
            // Note that if you need to create a Key in an AWS Region
            // other than the Region defined for the default user, you need to
            // pass the Region to the client constructor.
            var client = new AmazonKeyManagementServiceClient();

            // The call to CreateKeyAsync will create a symmetrical AWS KMS
            // key. For more information about symmetrical and asymmetrical
            // keys, see:
            //
            // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
            var response = await client.CreateKeyAsync(new CreateKeyRequest());

            // The KeyMetadata object contains information about the new AWS KMS
 key.
            KeyMetadata keyMetadata = response.KeyMetadata;

            if (keyMetadata is not null)
            {
                Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
 created.");
            }
            else
            {
                Console.WriteLine("Could not create KMS Key.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen vom Kunden verwalteten KMS-Schlüssel in AWS KMS

Im folgenden `create-key` Beispiel wird ein KMS-Schlüssel für die symmetrische Verschlüsselung erstellt.

Um den grundlegenden KMS-Schlüssel, einen symmetrischen Verschlüsselungsschlüssel, zu erstellen, müssen Sie keine Parameter angeben. Die Standardwerte für diese Parameter erstellen einen symmetrischen Verschlüsselungsschlüssel.

Da dieser Befehl keine Schlüsselrichtlinie angibt, erhält der KMS-Schlüssel die Standardschlüsselrichtlinie für programmgesteuert erstellte KMS-Schlüssel. Verwenden Sie den Befehl, um die Schlüsselrichtlinie anzuzeigen. `get-key-policy` Verwenden Sie den `put-key-policy` Befehl, um die Schlüsselrichtlinie zu ändern.

```
aws kms create-key
```

Der `create-key` Befehl gibt die wichtigsten Metadaten zurück, einschließlich der Schlüssel-ID und des ARN des neuen KMS-Schlüssels. Sie können diese Werte verwenden, um den KMS-Schlüssel in anderen AWS KMS-Vorgängen zu identifizieren. Die Ausgabe enthält die Tags nicht. Um die Tags für einen KMS-Schlüssel anzuzeigen, verwenden Sie den`list-resource-tags command`.

Ausgabe:

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2017-07-05T14:04:55-07:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "Description": "",
```

```
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_KMS"
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

Hinweis: `create-key` Mit dem Befehl können Sie keinen Alias angeben. Verwenden Sie den `create-alias` Befehl, um einen Alias für den neuen KMS-Schlüssel zu erstellen.

Weitere Informationen finden Sie im AWS Key Management Service Developer Guide unter Creating Keys.

Beispiel 2: So erstellen Sie einen asymmetrischen RSA-KMS-Schlüssel für die Verschlüsselung und Entschlüsselung

Im folgenden `create-key` Beispiel wird ein KMS-Schlüssel erstellt, der ein asymmetrisches RSA-Schlüsselpaar für die Verschlüsselung und Entschlüsselung enthält. Die Schlüsselspezifikation und die Schlüsselverwendung können nach der Erstellung des Schlüssels nicht geändert werden. :

```
aws kms create-key \
   --key-spec RSA_4096 \
   --key-usage ENCRYPT_DECRYPT
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2021-04-05T14:04:55-07:00",
        "CustomerMasterKeySpec": "RSA_4096",
```

```
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "RSAES_OAEP_SHA_1",
            "RSAES_OAEP_SHA_256"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "RSA_4096",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_KMS"
    }
}
```

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 3: So erstellen Sie einen KMS-Schlüssel mit asymmetrischer elliptischer Kurve zum Signieren und Überprüfen

Um einen asymmetrischen KMS-Schlüssel zu erstellen, der ein ECC-Schlüsselpaar (Asymmetric Elliptic Curve) zum Signieren und Überprüfen enthält. Der `--key-usage` Parameter ist erforderlich, obwohl er der einzig gültige Wert für ECC-KMS-Schlüssel `SIGN_VERIFY` ist. Die Schlüsselspezifikation und die Schlüsselverwendung können nach der Erstellung des Schlüssels nicht geändert werden. :

```
aws kms create-key \
    --key-spec ECC_NIST_P521 \
    --key-usage SIGN_VERIFY
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "ECC_NIST_P521",
```

```
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "ECC_NIST_P521",
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "MultiRegion": false,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ECDSA_SHA_512"
        ]
    }
}
```

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 4: So erstellen Sie einen asymmetrischen ML-DSA-KMS-Schlüssel zum Signieren und Überprüfen

In diesem Beispiel wird ein ML-DSA-Schlüssel (Module-Lattice Digital Signature Algorithm) zum Signieren und Überprüfen erstellt. Der Schlüsselverwendungsparameter ist erforderlich, obwohl er der einzig gültige Wert für ML-DSA-Schlüssel `SIGN_VERIFY` ist.

```
aws kms create-key \
    --key-spec ML_DSA_65 \
    --key-usage SIGN_VERIFY
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "ML_DSA_65",
```

```
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "MultiRegion": false,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ML_DSA_SHAKE_256"
        ]
    }
}
```

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 5: So erstellen Sie einen HMAC-KMS-Schlüssel

Im folgenden `create-key` Beispiel wird ein 384-Bit-HMAC-KMS-Schlüssel erstellt. Der `GENERATE_VERIFY_MAC` Wert für den `--key-usage` Parameter ist erforderlich, obwohl er der einzig gültige Wert für HMAC-KMS-Schlüssel ist.

```
aws kms create-key \
    --key-spec HMAC_384 \
    --key-usage GENERATE_VERIFY_MAC
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2022-04-05T14:04:55-07:00",
        "CustomerMasterKeySpec": "HMAC_384",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "HMAC_384",
        "KeyState": "Enabled",
        "KeyUsage": "GENERATE_VERIFY_MAC",
        "MacAlgorithms": [
            "HMAC_SHA_384"
        ],
```

```
        "MultiRegion": false,
        "Origin": "AWS_KMS"
    }
}
```

Weitere Informationen finden Sie unter HMAC-Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 6: So erstellen Sie einen primären KMS-Schlüssel für mehrere Regionen

Im folgenden `create-key` Beispiel wird ein primärer symmetrischer Verschlüsselungsschlüssel für mehrere Regionen erstellt. Da die Standardwerte für alle Parameter einen symmetrischen Verschlüsselungsschlüssel erzeugen, ist nur der `--multi-region` Parameter für diesen KMS-Schlüssel erforderlich. Um in der AWS CLI anzugeben, dass ein boolescher Parameter wahr ist, geben Sie einfach den Parameternamen an.

```
aws kms create-key \
    --multi-region
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2021-09-02T016:15:21-09:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
          "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "mrk-1234abcd12ab34cd56ef12345678990ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": true,
        "MultiRegionConfiguration": {
```

```
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
                "Region": "us-west-2"
            },
            "ReplicaKeys": []
        },
        "Origin": "AWS_KMS"
    }
}
```

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 7: So erstellen Sie einen KMS-Schlüssel für importiertes Schlüsselmaterial

Im folgenden `create-key` Beispiel wird ein KMS-Schlüssel ohne Schlüsselmaterial erstellt. Wenn der Vorgang abgeschlossen ist, können Sie Ihr eigenes Schlüsselmaterial in den KMS-Schlüssel importieren. Um diesen KMS-Schlüssel zu erstellen, setzen Sie den `--origin` Parameter auf`EXTERNAL`.

```
aws kms create-key \
    --origin EXTERNAL
```

Ausgabe:

```
{
     "KeyMetadata": {
         "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
         "AWSAccountId": "111122223333",
         "CreationDate": "2019-12-02T07:48:55-07:00",
         "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
         "Description": "",
         "Enabled": false,
         "EncryptionAlgorithms": [
             "SYMMETRIC_DEFAULT"
         ],
         "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
         "KeyManager": "CUSTOMER",
         "KeySpec": "SYMMETRIC_DEFAULT",
```

```
            "KeyState": "PendingImport",
            "KeyUsage": "ENCRYPT_DECRYPT",
            "MultiRegion": false,
            "Origin": "EXTERNAL"
        }
    }
```

Weitere Informationen finden Sie unter Importieren von Schlüsselmaterial in AWS KMS-Schlüssel im AWS Key Management Service Developer Guide.

Beispiel 6: So erstellen Sie einen KMS-Schlüssel in einem AWS CloudHSM-Schlüsselspeicher

Das folgende `create-key` Beispiel erstellt einen KMS-Schlüssel im angegebenen AWS CloudHSM-Schlüsselspeicher. Der Vorgang erstellt den KMS-Schlüssel und seine Metadaten in AWS KMS und erstellt das Schlüsselmaterial im AWS CloudHSM-Cluster, der dem benutzerdefinierten Schlüsselspeicher zugeordnet ist. Die Parameter `--custom-key-store-id` und `--origin` müssen angegeben werden.

```
aws kms create-key \
    --origin AWS_CLOUDHSM \
    --custom-key-store-id cks-1234567890abcdef0
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CloudHsmClusterId": "cluster-1a23b4cdefg",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "CustomKeyStoreId": "cks-1234567890abcdef0",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
```

```
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_CLOUDHSM"
    }
}
```

Weitere Informationen finden Sie unter AWS CloudHSM Key Stores im AWS Key Management Service Developer Guide.

Beispiel 8: So erstellen Sie einen KMS-Schlüssel in einem externen Schlüsselspeicher

Im folgenden `create-key` Beispiel wird ein KMS-Schlüssel im angegebenen externen Schlüsselspeicher erstellt. Die `--xks-key-id` Parameter `--custom-key-store-id--origin`, und sind in diesem Befehl erforderlich.

Der `--xks-key-id` Parameter gibt die ID eines vorhandenen symmetrischen Verschlüsselungsschlüssels in Ihrem externen Schlüsselmanager an. Dieser Schlüssel dient als externes Schlüsselmaterial für den KMS-Schlüssel. Der Wert des `--origin` Parameters muss sein. Der Parameter muss `EXTERNAL_KEY_STORE` einen externen Schlüsselspeicher identifizieren, der mit seinem externen `custom-key-store-id` Schlüsselspeicher-Proxy verbunden ist.

```
aws kms create-key \
    --origin EXTERNAL_KEY_STORE \
    --custom-key-store-id cks-9876543210fedcba9 \
    --xks-key-id bb8562717f809024
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2022-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "CustomKeyStoreId": "cks-9876543210fedcba9",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
```

```
            "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "EXTERNAL_KEY_STORE",
        "XksKeyConfiguration": {
            "Id": "bb8562717f809024"
        }
    }
}
```

Weitere Informationen finden Sie unter Externe Schlüsselspeicher im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie CreateKeyunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
```

```
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
 key: " + ex.getMessage(), ex);
            });
    }
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- Einzelheiten zur API finden Sie CreateKeyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /***
    * @param string $keySpec
    * @param string $keyUsage
    * @param string $description
    * @return array
    */
   public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
   {
       $parameters = ['Description' => $description];
       if($keySpec && $keyUsage){
           $parameters['KeySpec'] = $keySpec;
           $parameters['KeyUsage'] = $keyUsage;
       }
       try {
           $result = $this->client->createKey($parameters);
           return $result['KeyMetadata'];
       }catch(KmsException $caught){
           // Check for error specific to createKey operations
           if ($caught->getAwsErrorMessage() == "LimitExceededException"){
               echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
           }
           throw $caught;
       }
   }
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateKeyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: 'CreatedBy',
                               tag_value: 'ExampleUser'
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client) -> Result<(), Error> {
    let resp = client.create_key().send().await?;

    let id = resp.key_metadata.as_ref().unwrap().key_id();

    println!("Key: {}", id);

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateKeyin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Decrypt`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Decrypt` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: Um eine verschlüsselte Nachricht mit einem symmetrischen KMS-Schlüssel zu entschlüsseln (Linux und macOS)

Das folgende `decrypt` Befehlsbeispiel zeigt die empfohlene Methode zum Entschlüsseln von Daten mit der AWS CLI. Diese Version zeigt, wie Daten unter einem symmetrischen KMS-Schlüssel entschlüsselt werden.

Geben Sie den Geheimtext in einer Datei an. Verwenden Sie im Wert des `--ciphertext-blob` Parameters das `fileb://` Präfix, das die CLI anweist, die Daten aus einer Binärdatei zu lesen. Wenn sich die Datei nicht im aktuellen Verzeichnis befindet, geben Sie den vollständigen Dateipfad ein. Weitere Informationen zum Lesen von AWS CLI-Parameterwerten aus einer Datei finden Sie unter AWS CLI-Parameter aus einer Datei laden < https://docs.aws.amazon.com/cli/ latest/userguide/cli - usage-parameters-file .html> im AWS Command Line Interface User Guide und Best Practices for Local File Parameters< https://aws.amazon.com/blogs/ developer/ best-practices-for-local -file-parameters/> im AWS Command Line Tool Blog .Geben Sie den KMS-Schlüssel an, um den Chiffretext zu entschlüsseln. Der Parameter ist bei der Entschlüsselung mit einem symmetrischen KMS-Schlüssel nicht erforderlich. `--key-id` AWS KMS kann die Schlüssel-ID des KMS-Schlüssels, der zur Verschlüsselung der Daten verwendet wurde, aus den Metadaten im Chiffretext abrufen. Es ist jedoch immer eine bewährte Methode, den von Ihnen verwendeten KMS-Schlüssel anzugeben. Diese Vorgehensweise stellt sicher, dass Sie den gewünschten KMS-Schlüssel verwenden, und verhindert, dass Sie versehentlich einen Chiffretext mit einem KMS-Schlüssel entschlüsseln, dem Sie nicht vertrauen. Fordern Sie die Klartext-Ausgabe als Textwert an. Der `--query` Parameter weist die CLI an, nur den Wert des Felds aus der Ausgabe abzurufen. `Plaintext` Der `--output` Parameter gibt die Ausgabe als Text zurück. Base64-dekodiert den Klartext und speichert ihn in einer Datei. Im folgenden Beispiel wird der Wert des `Plaintext` Parameters über die Pipeline (|) an das Base64-Hilfsprogramm übergeben, das ihn dekodiert. Anschließend leitet er die dekodierte Ausgabe in die Datei um (>). `ExamplePlaintext`

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto.

```
aws kms decrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
```

```
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --output text \
    --query Plaintext | base64 \
    --decode > ExamplePlaintextFile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Die Ausgabe des `decrypt` Befehls wird base64-dekodiert und in einer Datei gespeichert.

Weitere Informationen finden Sie unter Decrypt in der API-Referenz für den AWS Key Management Service.

Beispiel 2: So entschlüsseln Sie eine verschlüsselte Nachricht mit einem symmetrischen KMS-Schlüssel (Windows-Eingabeaufforderung)

Das folgende Beispiel ist dasselbe wie das vorherige, außer dass es das `certutil` Hilfsprogramm zur Base64-Decodierung der Klartextdaten verwendet. Für dieses Verfahren sind zwei Befehle erforderlich, wie in den folgenden Beispielen gezeigt.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto.

```
aws kms decrypt ^
    --ciphertext-blob fileb://ExampleEncryptedFile ^
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
    --output text ^
    --query Plaintext > ExamplePlaintextFile.base64
```

Führen Sie den Befehl `certutil` aus.

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

Ausgabe:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Weitere Informationen finden Sie unter Decrypt in der API-Referenz für den AWS Key Management Service.

Beispiel 3: Um eine verschlüsselte Nachricht mit einem asymmetrischen KMS-Schlüssel zu entschlüsseln (Linux und macOS)

Das folgende `decrypt` Befehlsbeispiel zeigt, wie Daten entschlüsselt werden, die unter einem asymmetrischen RSA-KMS-Schlüssel verschlüsselt wurden.

Bei Verwendung eines asymmetrischen KMS-Schlüssels ist der `encryption-algorithm` Parameter erforderlich, der den Algorithmus angibt, der zur Verschlüsselung des Klartextes verwendet wird.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem Konto. AWS

```
aws kms decrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --encryption-algorithm RSAES_OAEP_SHA_256 \
    --output text \
    --query Plaintext | base64 \
    --decode > ExamplePlaintextFile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Die Ausgabe des `decrypt` Befehls wird base64-dekodiert und in einer Datei gespeichert.

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie unter Decrypt in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Asynchronously decrypts the given encrypted data using the specified key
ID.
    *
    * @param encryptedData The encrypted data to be decrypted.
    * @param keyId The ID of the key to be used for decryption.
    * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
    *         If an error occurs during the decryption process, the
CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
String.
    */
   public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
       DecryptRequest decryptRequest = DecryptRequest.builder()
           .ciphertextBlob(encryptedData)
           .keyId(keyId)
           .build();

       CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
       responseFuture.whenComplete((decryptResponse, exception) -> {
           if (exception == null) {
               logger.info("Data decrypted successfully for key ID: " + keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
               }
           }
       });

       return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
   }
```

- Einzelheiten zur API finden Sie unter Decrypt in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext
```

```
        // Print the decrypted data.
        print(myVal)
    }
}
```

- Einzelheiten zur API finden Sie unter Decrypt in AWS SDK for Kotlin API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
 = "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie unter Decrypt in der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id,
 CiphertextBlob=cipher_text)[
                "Plaintext"
```

```
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie unter Decrypt in AWS SDK for Python (Boto3) API-Referenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Decrypted blob

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593
blob_packed = [blob].pack('H*')

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts 'Raw text: '
puts resp.plaintext
```

- Einzelheiten zur API finden Sie unter Decrypt in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(),
 Error> {
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(filename)
        .map(|input| {
            base64::decode(input).expect("Input file does not contain valid base
 64 characters.")
        })
        .map(Blob::new);

    let resp = client
        .decrypt()
        .key_id(key)
        .ciphertext_blob(data.unwrap())
        .send()
        .await?;

    let inner = resp.plaintext.unwrap();
    let bytes = inner.as_ref();

    let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to
 UTF-8");

    println!();
    println!("Decoded string:");
    println!("{}", s);

    Ok(())
}
```

- Einzelheiten zur API finden Sie unter Decrypt in AWS SDK for Rust API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DeleteAlias`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAlias` verwendet wird.

CLI

AWS CLI

Um einen AWS KMS-Alias zu löschen

Im folgenden `delete-alias` Beispiel wird der Alias `alias/example-alias` gelöscht. Der Aliasname muss mit alias/ beginnen.

```
aws kms delete-alias \
    --alias-name alias/example-alias
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den Befehl, um den Alias zu finden. `list-aliases`

Weitere Informationen finden Sie unter Löschen eines Alias im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie DeleteAliasunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
            });
    }
```

- Einzelheiten zur API finden Sie DeleteAliasin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
```

```
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteAliasin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteAliasin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DescribeKey`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Retrieve information about an AWS Key Management Service (AWS KMS) key.
/// You can supply either the key Id or the key Amazon Resource Name (ARN)
/// to the DescribeKeyRequest KeyId property.
/// </summary>
public class DescribeKey
{
    public static async Task Main()
    {
        var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
        var request = new DescribeKeyRequest
        {
            KeyId = keyId,
        };

        var client = new AmazonKeyManagementServiceClient();

        var response = await client.DescribeKeyAsync(request);
        var metadata = response.KeyMetadata;

        Console.WriteLine($"{metadata.KeyId} created on:
{metadata.CreationDate}");
        Console.WriteLine($"State: {metadata.KeyState}");
        Console.WriteLine($"{metadata.Description}");
    }
}
```

- Einzelheiten zur API finden Sie DescribeKeyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: Um detaillierte Informationen zu einem KMS-Schlüssel zu finden

Im folgenden `describe-key` Beispiel werden detaillierte Informationen zum AWS verwalteten Schlüssel für Amazon S3 im Beispielkonto und in der Region abgerufen. Sie können diesen Befehl verwenden, um Details zu AWS verwalteten Schlüsseln und kundenverwalteten Schlüsseln zu finden.

Verwenden Sie den `key-id` Parameter, um den KMS-Schlüssel anzugeben. In diesem Beispiel wird ein Aliasnamenwert verwendet, aber Sie können in diesem Befehl eine Schlüssel-ID, einen Schlüssel-ARN, einen Aliasnamen oder einen Alias-ARN verwenden.

```
aws kms describe-key \
    --key-id alias/aws/s3
```

Ausgabe:

```
{
    "KeyMetadata": {
        "AWSAccountId": "846764612917",
        "KeyId": "b8a9477d-836c-491f-857e-07937918959b",
        "Arn": "arn:aws:kms:us-west-2:846764612917:key/
b8a9477d-836c-491f-857e-07937918959b",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CreationDate": 2017-06-30T21:44:32.140000+00:00,
        "Enabled": true,
        "Description": "Default KMS key that protects my S3 objects when no other
 key is defined",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "AWS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
```

```
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

Weitere Informationen finden Sie unter Schlüssel anzeigen im AWS Key Management Service Developer Guide.

Beispiel 2: Um Details zu einem asymmetrischen RSA-KMS-Schlüssel abzurufen

Im folgenden `describe-key` Beispiel werden detaillierte Informationen zu einem asymmetrischen RSA-KMS-Schlüssel abgerufen, der zum Signieren und Überprüfen verwendet wird.

```
aws kms describe-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Ausgabe:

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2019-12-02T19:47:14.861000+00:00",
        "CustomerMasterKeySpec": "RSA_2048",
        "Enabled": false,
        "Description": "",
        "KeyState": "Disabled",
        "Origin": "AWS_KMS",
        "MultiRegion": false,
        "KeyManager": "CUSTOMER",
        "KeySpec": "RSA_2048",
        "KeyUsage": "SIGN_VERIFY",
        "SigningAlgorithms": [
            "RSASSA_PKCS1_V1_5_SHA_256",
            "RSASSA_PKCS1_V1_5_SHA_384",
            "RSASSA_PKCS1_V1_5_SHA_512",
            "RSASSA_PSS_SHA_256",
            "RSASSA_PSS_SHA_384",
            "RSASSA_PSS_SHA_512"
        ]
```

```
    }
}
```

Beispiel 3: Um Details zu einem Replikatschlüssel für mehrere Regionen abzurufen

Im folgenden `describe-key` Beispiel werden Metadaten für einen Replikatschlüssel mit mehreren Regionen abgerufen. Dieser Schlüssel für mehrere Regionen ist ein symmetrischer Verschlüsselungsschlüssel. Die Ausgabe eines `describe-key` Befehls für einen beliebigen Schlüssel mit mehreren Regionen gibt Informationen über den Primärschlüssel und alle zugehörigen Replikate zurück.

```
aws kms describe-key \
    --key-id arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab
```

Ausgabe:

```
{
    "KeyMetadata": {
        "MultiRegion": true,
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "CreationDate": "2021-06-28T21:09:16.114000+00:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "Description": "",
        "Enabled": true,
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "Origin": "AWS_KMS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
```

```
                "Region": "us-west-2"
            },
            "ReplicaKeys": [
                {
                    "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "eu-west-1"
                },
                {
                    "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "ap-northeast-1"
                },
                {
                    "Arn": "arn:aws:kms:sa-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "sa-east-1"
                }
            ]
        }
    }
}
```

Beispiel 4: Um Details zu einem HMAC-KMS-Schlüssel abzurufen

Im folgenden `describe-key` Beispiel werden detaillierte Informationen zu einem HMAC-KMS-Schlüssel abgerufen.

```
aws kms describe-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Ausgabe:

```
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2022-04-03T22:23:10.194000+00:00",
        "Enabled": true,
        "Description": "Test key",
        "KeyUsage": "GENERATE_VERIFY_MAC",
```

```
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "HMAC_256",
        "MacAlgorithms": [
            "HMAC_SHA_256"
        ],
        "MultiRegion": false
    }
}
```

- Einzelheiten zur API finden Sie unter DescribeKey AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key
state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
```

```
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
 keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- Einzelheiten zur API finden Sie DescribeKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- Einzelheiten zur API finden Sie DescribeKeyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeKeyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id,
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DescribeKeyin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`DisableKey`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisableKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Disable an AWS Key Management Service (AWS KMS) key and then retrieve
/// the key's status to show that it has been disabled.
/// </summary>
```

```
    public class DisableKey
    {
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();

            // The identifier of the AWS KMS key to disable. You can use the
            // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

            var request = new DisableKeyRequest
            {
                KeyId = keyId,
            };

            var response = await client.DisableKeyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                // Retrieve information about the key to show that it has now
                // been disabled.
                var describeResponse = await client.DescribeKeyAsync(new
 DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
 {describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie DisableKeyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um einen KMS-Schlüssel vorübergehend zu deaktivieren

Der folgende `disable-key` Befehl deaktiviert einen vom Kunden verwalteten KMS-Schlüssel. Verwenden Sie den Befehl, um den KMS-Schlüssel erneut zu aktivieren. `enable-key`

```
aws kms disable-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Enabling and Disabling Keys im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie unter DisableKey AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS)
key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key
has been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
```

```
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
 throwable);
            });
    }
```

- Einzelheiten zur API finden Sie DisableKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- Einzelheiten zur API finden Sie DisableKeyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie DisableKeyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def disable_key(self, key_id: str) -> None:
        try:
            self.kms_client.disable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't disable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DisableKeyin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`EnableKey`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Enable an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class EnableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to enable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new EnableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.EnableKeyAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            // Retrieve information about the key to show that it has now
            // been enabled.
            var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
```

```
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
 {describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie EnableKeyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um einen KMS-Schlüssel zu aktivieren

Das folgende `enable-key` Beispiel aktiviert einen vom Kunden verwalteten Schlüssel. Sie können einen Befehl wie diesen verwenden, um einen KMS-Schlüssel zu aktivieren, den Sie mit dem `disable-key` Befehl vorübergehend deaktiviert haben. Sie können ihn auch verwenden, um einen KMS-Schlüssel zu aktivieren, der deaktiviert ist, weil der Löschvorgang geplant war und der Löschvorgang abgebrochen wurde.

Verwenden Sie den `key-id` Parameter, um den KMS-Schlüssel anzugeben. In diesem Beispiel wird ein Schlüssel-ID-Wert verwendet, aber Sie können in diesem Befehl auch eine Schlüssel-ID oder einen Schlüssel-ARN-Wert verwenden.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige.

```
aws kms enable-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den `describe-key` Befehl, um zu überprüfen, ob der KMS-Schlüssel aktiviert ist. Sehen Sie sich die Werte der `Enabled` Felder `KeyState` und in der `describe-key` Ausgabe an.

Weitere Informationen finden Sie unter Enabling and Disabling Keys im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie unter EnableKey AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously enables the specified key.
    *
    * @param keyId the ID of the key to enable
    * @return a {@link CompletableFuture} that completes when the key has been
enabled
    */
   public CompletableFuture<Void> enableKeyAsync(String keyId) {
       EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key with ID [{}] has been enabled.", keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred
while enabling key: " + exception.getMessage(), exception);
               }
           }
       });

       return responseFuture.thenApply(response -> null);
   }
```

- Einzelheiten zur API finden Sie EnableKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- Einzelheiten zur API finden Sie EnableKeyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie EnableKeyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
```

```
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key(self, key_id: str) -> None:
        """
        Enables a key. Gets the key state after each state change.

        :param key_id: The ARN or ID of the key to enable.
        """
        try:
            self.kms_client.enable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie EnableKeyin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`EnableKeyRotation`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableKeyRotation` verwendet wird.

CLI

AWS CLI

Um die automatische Rotation eines KMS-Schlüssels zu aktivieren

Das folgende `enable-key-rotation` Beispiel ermöglicht die automatische Rotation eines vom Kunden verwalteten KMS-Schlüssels mit einem Rotationszeitraum von 180 Tagen. Der KMS-Schlüssel wird ab dem Datum, an dem dieser Befehl abgeschlossen wurde, ein Jahr (ungefähr 365 Tage) rotiert, und danach jedes Jahr.

Der `--key-id` Parameter identifiziert den KMS-Schlüssel. In diesem Beispiel wird ein ARN-Schlüsselwert verwendet, Sie können jedoch entweder die Schlüssel-ID oder den ARN des KMS-Schlüssels verwenden. Der `--rotation-period-in-days` Parameter gibt die Anzahl der Tage zwischen den einzelnen Rotationsdaten an. Geben Sie einen Wert zwischen 90 und 2560 Tagen an. Wenn kein Wert angegeben ist, beträgt der Standardwert 365 Tage.

```
aws kms enable-key-rotation \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --rotation-period-in-days 180
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den `get-key-rotation-status` Befehl, um zu überprüfen, ob der KMS-Schlüssel aktiviert ist.

Weitere Informationen finden Sie unter Rotation von Schlüsseln im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie EnableKeyRotationin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
```

```
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie EnableKeyRotationin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Encrypt`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Encrypt` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: Um den Inhalt einer Datei unter Linux oder macOS zu verschlüsseln

Der folgende `encrypt` Befehl demonstriert die empfohlene Methode zum Verschlüsseln von Daten mit der AWS CLI.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob | base64 \
    --decode > ExampleEncryptedFile
```

Der Befehl macht mehrere Dinge:

Verwendet den `--plaintext` Parameter, um die zu verschlüsselnden Daten anzugeben. Dieser Parameterwert muss Base64-kodiert sein. Der Wert des `plaintext` Parameters muss Base64-kodiert sein, oder Sie müssen das `fileb://` Präfix verwenden, das die AWS CLI anweist, Binärdaten aus der Datei zu lesen. Wenn sich die Datei nicht im aktuellen Verzeichnis befindet, geben Sie den vollständigen Dateipfad ein. Beispiel: `fileb:///var/tmp/ExamplePlaintextFile` oder `fileb://C:\Temp\ExamplePlaintextFile`. Weitere Informationen zum Lesen von AWS CLI-Parameterwerten aus einer Datei finden Sie unter Laden von Parametern aus einer Datei im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle und Best Practices für lokale Dateiparameter im Blog des AWS Befehlszeilentools. Verwendet die `--query` Parameter `--output` und, um die Ausgabe des Befehls zu steuern.Diese Parameter extrahieren die verschlüsselten Daten, den so genannten Chiffretext, aus der Befehlsausgabe. Weitere Informationen zur Steuerung der Ausgabe finden Sie unter Steuerung des Befehls. Ausgabe im AWS Command Line Interface User Guide. Verwendet das `base64` Hilfsprogramm, um die extrahierte Ausgabe in Binärdaten zu dekodieren. Der Chiffretext, der von einem erfolgreichen `encrypt` Befehl zurückgegeben wird, ist Base64-codierter Text. Sie müssen diesen Text dekodieren, bevor Sie ihn mit der AWS CLI entschlüsseln können. Speichert den binären Chiffretext in einer Datei. Der letzte Teil des Befehls (`> ExampleEncryptedFile`) speichert den binären Chiffretext in einer Datei, um die

Entschlüsselung zu vereinfachen. Einen Beispielbefehl, der die AWS CLI zum Entschlüsseln von Daten verwendet, finden Sie in den Entschlüsselungsbeispielen.

Beispiel 2: Verwenden der AWS CLI zum Verschlüsseln von Daten unter Windows

Dieses Beispiel ist dasselbe wie das vorherige, außer dass es das `certutil` Tool anstelle von `base64` verwendet. Für dieses Verfahren sind zwei Befehle erforderlich, wie im folgenden Beispiel gezeigt.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob > C:\Temp\ExampleEncryptedFile.base64

certutil -decode C:\Temp\ExampleEncryptedFile.base64 C:\Temp\ExampleEncryptedFile
```

Beispiel 3: Verschlüsselung mit einem asymmetrischen KMS-Schlüssel

Der folgende `encrypt` Befehl zeigt, wie Klartext mit einem asymmetrischen KMS-Schlüssel verschlüsselt wird. Der Parameter `--encryption-algorithm` muss angegeben werden. Wie bei allen `encrypt` CLI-Befehlen muss der `plaintext` Parameter base64-codiert sein, oder Sie müssen das `fileb://` Präfix verwenden, das die AWS CLI anweist, Binärdaten aus der Datei zu lesen.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --encryption-algorithm RSAES_OAEP_SHA_256 \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob | base64 \
    --decode > ExampleEncryptedFile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie unter Verschlüsseln in der Befehlsreferenz.AWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Encrypts the given text asynchronously using the specified KMS client and
key ID.
    *
    * @param keyId the ID of the KMS key to use for encryption
    * @param text the text to encrypt
    * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
    */
   public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
       SdkBytes myBytes = SdkBytes.fromUtf8String(text);
       EncryptRequest encryptRequest = EncryptRequest.builder()
           .keyId(keyId)
           .plaintext(myBytes)
           .build();

       CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
       return responseFuture.whenComplete((response, ex) -> {
           if (response != null) {
               String algorithm = response.encryptionAlgorithm().toString();
               logger.info("The string was encrypted with algorithm {}.",
algorithm);
           } else {
               throw new RuntimeException(ex);
           }
       }).thenApply(EncryptResponse::ciphertextBlob);
   }
```

- Einzelheiten zur API finden Sie unter Verschlüsseln in der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
```

```
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Einzelheiten zur API finden Sie unter Encrypt in AWS SDK for Kotlin API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is
 not enabled.\n";
            }
            throw $caught;
```

```
        }
    }
```

- Einzelheiten zur API finden Sie unter Verschlüsseln in der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
```

```
            response = self.kms_client.encrypt(KeyId=key_id,
 Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
 {response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie unter Encrypt in AWS SDK for Python (Boto3) API-Referenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'
```

```
text = '1234567890'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.encrypt({
                        key_id: keyId,
                        plaintext: text
                      })

# Display a readable version of the resulting encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Einzelheiten zur API finden Sie unter Verschlüsseln in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn encrypt_string(
    verbose: bool,
    client: &Client,
    text: &str,
    key: &str,
    out_file: &str,
) -> Result<(), Error> {
    let blob = Blob::new(text.as_bytes());

    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();
```

```
    let s = base64::encode(bytes);

    let mut ofile = File::create(out_file).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {:?}", out_file);
        println!("{}", s);
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie in der API-Referenz zum Verschlüsseln im AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GenerateDataKey`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GenerateDataKey` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um einen symmetrischen 256-Bit-Datenschlüssel zu generieren

Im folgenden `generate-data-key` Beispiel wird ein symmetrischer 256-Bit-Datenschlüssel zur Verwendung außerhalb von angefordert. AWS Der Befehl gibt einen Klartext-Datenschlüssel zur sofortigen Verwendung und Löschung sowie eine Kopie dieses Datenschlüssels zurück, die unter dem angegebenen KMS-Schlüssel verschlüsselt wurde. Sie können den verschlüsselten Datenschlüssel sicher neben den verschlüsselten Daten speichern.

Um einen 256-Bit-Datenschlüssel anzufordern, verwenden Sie den `key-spec` Parameter mit dem Wert. `AES_256` Um einen 128-Bit-Datenschlüssel anzufordern, verwenden Sie

den `key-spec` Parameter mit dem Wert. `AES_128` Verwenden Sie für alle anderen Datenschlüssellängen den `number-of-bytes` Parameter.

Bei dem von Ihnen angegebenen KMS-Schlüssel muss es sich um einen KMS-Schlüssel mit symmetrischer Verschlüsselung handeln, d. h. um einen KMS-Schlüssel mit dem Schlüsselspezifikationswert SYMMETRIC_DEFAULT.

```
aws kms generate-data-key \
    --key-id alias/ExampleAlias \
    --key-spec AES_256
```

Ausgabe:

```
{
    "Plaintext": "VdzKNHGzUAzJeRBVY+uUmofUGGiDzyB3+i9fVkh3piw=",
    "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
    "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
+YdhV8MrkBQPeac0ReRVNDt9qleAt+SHgIRF8P0H+7U="
}
```

Der `Plaintext` (Klartext-Datenschlüssel) und der `CiphertextBlob` (verschlüsselte Datenschlüssel) werden im Base64-codierten Format zurückgegeben.

Weitere Informationen finden Sie unter Datenschlüssel im AWS Key Management Service Developer Guide. Beispiel 2: So generieren Sie einen symmetrischen 512-Bit-Datenschlüssel

Im folgenden `generate-data-key` Beispiel wird ein symmetrischer 512-Bit-Datenschlüssel für die Verschlüsselung und Entschlüsselung angefordert. Der Befehl gibt einen Klartext-Datenschlüssel zur sofortigen Verwendung und Löschung sowie eine Kopie dieses Datenschlüssels zurück, die unter dem angegebenen KMS-Schlüssel verschlüsselt wurde. Sie können den verschlüsselten Datenschlüssel sicher neben den verschlüsselten Daten speichern.

Verwenden Sie den `number-of-bytes` Parameter, um eine andere Schlüssellänge als 128 oder 256 Bit anzufordern. Um einen 512-Bit-Datenschlüssel anzufordern, verwendet das folgende Beispiel den `number-of-bytes` Parameter mit einem Wert von 64 (Byte).

Bei dem von Ihnen angegebenen KMS-Schlüssel muss es sich um einen KMS-Schlüssel mit symmetrischer Verschlüsselung handeln, d. h. um einen KMS-Schlüssel mit dem Schlüsselspezifikationswert SYMMETRIC_DEFAULT.

HINWEIS: Die Werte in der Ausgabe dieses Beispiels sind zur Anzeige gekürzt.

```
aws kms generate-data-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --number-of-bytes 64
```

Ausgabe:

```
{
    "CiphertextBlob": "AQIBAHi6LtupRpdKl2aJTzkK6FbhOtQkMlQJJH3PdtHvS/y+hAEnX/
QQNmMwDfg2korNMEc8AAACaDCCAmQGCSqGSIb3DQEHBqCCAlUwggJRAgEAMIICSgYJKoZ...",
    "Plaintext": "ty8Lr0Bk6OF07M2BWt6qbFdNB
+G00ZLtf5MSEb4al3R2UKWGOp06njAwy2n72VRm2m7z/
Pm9Wpbvttz6a4lSo9hgPvKhZ5y6RTm4OovEXiVfBveyX3DQxDzRSwbKDPk/...",
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

`Plaintext`(Klartext-Datenschlüssel) und `CiphertextBlob` (verschlüsselter Datenschlüssel) werden im Base64-codierten Format zurückgegeben.

Weitere Informationen finden Sie unter Datenschlüssel im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie GenerateDataKeyunter AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def generate_data_key(self, key_id):
        """
        Generates a symmetric data key that can be used for client-side
 encryption.
        """
        answer = input(
            f"Do you want to generate a symmetric data key from key {key_id} (y/
n)? "
        )
        if answer.lower() == "y":
            try:
                data_key = self.kms_client.generate_data_key(
                    KeyId=key_id, KeySpec="AES_256"
                )
            except ClientError as err:
                logger.error(
                    "Couldn't generate a data key for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
            else:
                pprint(data_key)
```

- Einzelheiten zur API finden Sie GenerateDataKeyin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Einzelheiten zur API finden Sie GenerateDataKeyin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GenerateDataKeyWithoutPlaintext`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GenerateDataKeyWithoutPlaintext` verwendet wird.

CLI

AWS CLI

Um einen symmetrischen 256-Bit-Datenschlüssel ohne Klartext-Schlüssel zu generieren

Im folgenden `generate-data-key-without-plaintext` Beispiel wird eine verschlüsselte Kopie eines symmetrischen 256-Bit-Datenschlüssels zur Verwendung außerhalb von angefordert. AWS Sie können AWS KMS aufrufen, um den Datenschlüssel zu entschlüsseln, wenn Sie bereit sind, ihn zu verwenden.

Um einen 256-Bit-Datenschlüssel anzufordern, verwenden Sie den `key-spec` Parameter mit dem Wert. `AES_256` Um einen 128-Bit-Datenschlüssel anzufordern, verwenden Sie den `key-spec` Parameter mit dem Wert. `AES_128` Verwenden Sie für alle anderen Datenschlüssellängen den `number-of-bytes` Parameter.

Bei dem von Ihnen angegebenen KMS-Schlüssel muss es sich um einen KMS-Schlüssel mit symmetrischer Verschlüsselung handeln, d. h. um einen KMS-Schlüssel mit dem Schlüsselspezifikationswert SYMMETRIC_DEFAULT.

```
aws kms generate-data-key-without-plaintext \
    --key-id "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" \
    --key-spec AES_256
```

Ausgabe:

```
{
    "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
    "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
```

```
}
```

Der `CiphertextBlob` (verschlüsselte Datenschlüssel) wird im Base64-codierten Format zurückgegeben.

Weitere Informationen finden Sie unter Datenschlüssel im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie GenerateDataKeyWithoutPlaintextunter AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Einzelheiten zur API finden Sie GenerateDataKeyWithoutPlaintextin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GenerateRandom`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GenerateRandom` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um eine 256-Bit-Zufallsbytezeichenfolge zu generieren (Linux oder macOs)

Das folgende `generate-random` Beispiel generiert eine Base64-kodierte 256-Bit-Zufallsbytezeichenfolge (32 Byte). Das Beispiel dekodiert die Bytezeichenfolge und speichert sie in der Zufallsdatei.

Wenn Sie diesen Befehl ausführen, müssen Sie den `number-of-bytes` Parameter verwenden, um die Länge des Zufallswerts in Byte anzugeben.

Sie geben keinen KMS-Schlüssel an, wenn Sie diesen Befehl ausführen. Die zufällige Byte-Zeichenfolge hat nichts mit einem KMS-Schlüssel zu tun.

Standardmäßig generiert AWS KMS die Zufallszahl. Wenn Sie jedoch einen benutzerdefinierten Schlüsselspeicher angeben, wird die zufällige Bytezeichenfolge in dem AWS CloudHSM-Cluster generiert, der dem benutzerdefinierten Schlüsselspeicher zugeordnet ist.

In diesem Beispiel werden die folgenden Parameter und Werte verwendet:

Es verwendet den erforderlichen `--number-of-bytes` Parameter mit dem Wert von, `32` um eine 32-Byte-Zeichenfolge (256-Bit) anzufordern.Es verwendet den `--output` Parameter mit dem Wert von, `text` um die AWS CLI anzuweisen, die Ausgabe als Text und nicht als JSON zurückzugeben.Es verwendet den, um den Wert der `Plaintext` Eigenschaft aus der Antwort `--query parameter` zu extrahieren.Es leitet (|) die Ausgabe des Befehls an das

`base64` Hilfsprogramm weiter, das die extrahierte Ausgabe dekodiert. Es verwendet den Umleitungsoperator (**>**), um die dekodierte Bytezeichenfolge zu speichern File.it verwendet den Umleitungsoperator (**>**) `ExampleRandom` um den binären Chiffretext in einer Datei zu speichern.

```
aws kms generate-random \
    --number-of-bytes 32 \
    --output text \
    --query Plaintext | base64 --decode > ExampleRandom
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie GenerateRandomin der API-Referenz für den AWS Key Management Service.

Beispiel 2: So generieren Sie eine 256-Bit-Zufallszahl (Windows-Eingabeaufforderung)

Im folgenden Beispiel wird der `generate-random` Befehl verwendet, um eine Base64-kodierte 256-Bit- (32-Byte) -Bit-Zufallsbytezeichenfolge zu generieren. Das Beispiel dekodiert die Bytezeichenfolge und speichert sie in der Zufallsdatei. Dieses Beispiel entspricht dem vorherigen Beispiel, außer dass es das `certutil` Hilfsprogramm in Windows verwendet, um die zufällige Bytezeichenfolge base64-dekodieren zu lassen, bevor sie in einer Datei gespeichert wird.

Generieren Sie zunächst eine Base64-kodierte Zufallsbytezeichenfolge und speichern Sie sie in einer temporären Datei,. `ExampleRandom.base64`

```
aws kms generate-random \
    --number-of-bytes 32 \
    --output text \
    --query Plaintext > ExampleRandom.base64
```

Da die Ausgabe des `generate-random` Befehls in einer Datei gespeichert wird, erzeugt dieses Beispiel keine Ausgabe.

Verwenden Sie nun den `certutil -decode` Befehl, um die Base64-kodierte Bytezeichenfolge in der Datei zu dekodieren. `ExampleRandom.base64` Anschließend wird die dekodierte Bytezeichenfolge in der Datei gespeichert. `ExampleRandom`

```
certutil -decode ExampleRandom.base64 ExampleRandom
```

Ausgabe:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Weitere Informationen finden Sie GenerateRandomin der API-Referenz für den AWS Key Management Service.

- Einzelheiten zur API finden Sie GenerateRandomunter AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
        .number_of_bytes(length)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.plaintext.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Einzelheiten zur API finden Sie GenerateRandomin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`GetKeyPolicy`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetKeyPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

CLI

AWS CLI

Um eine Schlüsselrichtlinie von einem KMS-Schlüssel auf einen anderen KMS-Schlüssel zu kopieren

Im folgenden `get-key-policy` Beispiel wird die Schlüsselrichtlinie von einem KMS-Schlüssel abgerufen und in einer Textdatei gespeichert. Anschließend wird die Richtlinie eines anderen KMS-Schlüssels ersetzt, wobei die Textdatei als Richtlinieneingabe verwendet wird.

Da für den `--policy` Parameter von eine Zeichenfolge `put-key-policy` erforderlich ist, müssen Sie die `--output text` Option verwenden, um die Ausgabe als Textzeichenfolge statt als JSON zurückzugeben.

```
aws kms get-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --query Policy \
    --output text > policy.txt

aws kms put-key-policy \
    --policy-name default \
    --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
```

```
    --policy file://policy.txt
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie PutKeyPolicyin der AWS KMS-API-Referenz.

- Einzelheiten zur API finden Sie GetKeyPolicyunter AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
```

```
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Einzelheiten zur API finden Sie GetKeyPolicyin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListAliases`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListAliases` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   using System;
   using System.Threading.Tasks;
   using Amazon.KeyManagementService;
   using Amazon.KeyManagementService.Model;

   /// <summary>
   /// List the AWS Key Management Service (AWS KMS) aliases that have been
defined for
   /// the keys in the same AWS Region as the default user. If you want to list
   /// the aliases in a different Region, pass the Region to the client
   /// constructor.
   /// </summary>
   public class ListAliases
   {
       public static async Task Main()
       {
           var client = new AmazonKeyManagementServiceClient();
           var request = new ListAliasesRequest();
           var response = new ListAliasesResponse();

           do
           {
               response = await client.ListAliasesAsync(request);

               response.Aliases.ForEach(alias =>
               {
                   Console.WriteLine($"Created: {alias.CreationDate} Last
Update: {alias.LastUpdatedDate} Name: {alias.AliasName}");
               });

               request.Marker = response.NextMarker;
           }
```

```
            while (response.Truncated);
        }
    }
```

- Einzelheiten zur API finden Sie ListAliasesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: Um alle Aliase in einem AWS Konto und einer Region aufzulisten

Im folgenden Beispiel wird der `list-aliases` Befehl verwendet, um alle Aliase in der Standardregion des AWS Kontos aufzulisten. Die Ausgabe umfasst Aliase, die AWS verwalteten KMS-Schlüsseln und kundenverwalteten KMS-Schlüsseln zugeordnet sind.

```
aws kms list-aliases
```

Ausgabe:

```
{
    "Aliases": [
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/testKey",
            "AliasName": "alias/testKey",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/FinanceDept",
            "AliasName": "alias/FinanceDept",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
            "AliasName": "alias/aws/dynamodb",
            "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
```

```
            "AliasName": "alias/aws/ebs",
            "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef"
        },
        ...
    ]
}
```

Beispiel 2: Um alle Aliase für einen bestimmten KMS-Schlüssel aufzulisten

Im folgenden Beispiel werden der `list-aliases` Befehl und sein `key-id` Parameter verwendet, um alle Aliase aufzulisten, die einem bestimmten KMS-Schlüssel zugeordnet sind.

Jeder Alias ist nur einem KMS-Schlüssel zugeordnet, aber ein KMS-Schlüssel kann mehrere Aliase haben. Dieser Befehl ist sehr nützlich, da die AWS KMS-Konsole nur einen Alias für jeden KMS-Schlüssel auflistet. Um alle Aliase für einen KMS-Schlüssel zu finden, müssen Sie den `list-aliases` Befehl verwenden.

In diesem Beispiel wird die Schlüssel-ID des KMS-Schlüssels für den `--key-id` Parameter verwendet, aber Sie können in diesem Befehl eine Schlüssel-ID, einen Schlüssel-ARN, einen Aliasnamen oder einen Alias-ARN verwenden.

```
aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Ausgabe:

```
{
    "Aliases": [
        {
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/oregon-test-
key",
            "AliasName": "alias/oregon-test-key"
        },
        {
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project121-
test",
            "AliasName": "alias/project121-test"
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit Aliasen im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie unter ListAliases AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously lists all the aliases in the current AWS account.
    *
    * @return a {@link CompletableFuture} that completes when the list of
aliases has been processed
    */
   public CompletableFuture<Object> listAllAliasesAsync() {
       ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
           .limit(15)
           .build();

       ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
       return paginator.subscribe(response -> {
               response.aliases().forEach(alias ->
                   logger.info("The alias name is: " + alias.aliasName())
               );
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               if (ex.getCause() instanceof KmsException) {
                   KmsException e = (KmsException) ex.getCause();
                   throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
```

```
                }
            });
    }
```

- Einzelheiten zur API finden Sie ListAliasesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListAliasesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie ListAliasesin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
```

```
                        f"Do you want to see the next {page_size} aliases (y/n)?
"
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListAliasesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListGrants`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListGrants` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   using System;
   using System.Threading.Tasks;
   using Amazon.KeyManagementService;
   using Amazon.KeyManagementService.Model;

   /// <summary>
   /// List the AWS Key Management Service (AWS KMS) grants that are associated
with
   /// a specific key.
   /// </summary>
   public class ListGrants
   {
       public static async Task Main()
       {
           // The identifier of the AWS KMS key to disable. You can use the
           // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
           var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
           var client = new AmazonKeyManagementServiceClient();
           var request = new ListGrantsRequest
           {
               KeyId = keyId,
           };

           var response = new ListGrantsResponse();

           do
           {
               response = await client.ListGrantsAsync(request);

               response.Grants.ForEach(grant =>
               {
                   Console.WriteLine($"{grant.GrantId}");
               });

               request.Marker = response.NextMarker;
           }
           while (response.Truncated);
       }
   }
```

- Einzelheiten zur API finden Sie ListGrantsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So zeigen Sie die Grants für einen AWS KMS-Schlüssel an

Im folgenden `list-grants` Beispiel werden alle Grants für den angegebenen AWS verwalteten KMS-Schlüssel für Amazon DynamoDB in Ihrem Konto angezeigt. Dieser Zuschuss ermöglicht es DynamoDB, den KMS-Schlüssel in Ihrem Namen zu verwenden, um eine DynamoDB-Tabelle zu verschlüsseln, bevor sie auf die Festplatte geschrieben wird. Sie können einen Befehl wie diesen verwenden, um die Grants für die verwalteten KMS-Schlüssel und die vom Kunden AWS verwalteten KMS-Schlüssel im Konto und in der Region anzuzeigen. AWS

Dieser Befehl verwendet den `key-id` Parameter mit einer Schlüssel-ID, um den KMS-Schlüssel zu identifizieren. Sie können eine Schlüssel-ID oder einen Schlüssel-ARN verwenden, um den KMS-Schlüssel zu identifizieren. Verwenden Sie den `list-aliases` Befehl or, um die Schlüssel-ID oder den Schlüssel-ARN eines AWS verwalteten KMS-Schlüssels abzurufen. `list-keys`

```
aws kms list-grants \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Die Ausgabe zeigt, dass Amazon DynamoDB durch die Gewährung die Erlaubnis erhält, den KMS-Schlüssel für kryptografische Operationen zu verwenden und Details zum KMS-Schlüssel (`DescribeKey`) einzusehen und Grants zurückzuziehen (). `RetireGrant` Die `EncryptionContextSubset` Einschränkung beschränkt diese Berechtigungen auf Anfragen, die die angegebenen Verschlüsselungskontextpaare enthalten. Daher sind die Berechtigungen in der Gewährung nur für das angegebene Konto und die angegebene DynamoDB-Tabelle wirksam.

```
{
    "Grants": [
        {
            "Constraints": {
                "EncryptionContextSubset": {
                    "aws:dynamodb:subscriberId": "123456789012",
                    "aws:dynamodb:tableName": "Services"
                }
            },
```

```
            "IssuingAccount": "arn:aws:iam::123456789012:root",
            "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "ReEncryptFrom",
                "ReEncryptTo",
                "RetireGrant",
                "DescribeKey"
            ],
            "GrantId":
 "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59",
            "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",
            "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",
            "CreationDate": "2021-05-13T18:32:45.144000+00:00"
        }
    ]
}
```

Weitere Informationen finden Sie unter Grants in AWS KMS im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie ListGrantsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
```

```
    * @return a {@link CompletableFuture} that, when completed, will be null
if the operation succeeded, or will throw a {@link RuntimeException} if the
operation failed
    * @throws RuntimeException if there was an error listing the grants, either
due to an {@link KmsException} or an unexpected error
    */
   public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
       ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
           .keyId(keyId)
           .limit(15)
           .build();

       ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
       return paginator.subscribe(response -> {
               response.grants().forEach(grant -> {
                   logger.info("The grant Id is: " + grant.grantId());
               });
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               Throwable cause = ex.getCause();
               if (cause instanceof KmsException) {
                   throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
               }
           });
   }
```

- Einzelheiten zur API finden Sie ListGrantsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListGrantsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
```

```
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity
 or resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie ListGrantsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.
```

```
        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListGrantsin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListKeyPolicies`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListKeyPolicies` verwendet wird.

CLI

AWS CLI

Um die Namen der wichtigsten Richtlinien für einen KMS-Schlüssel abzurufen

Im folgenden `list-key-policies` Beispiel werden die Namen der wichtigsten Richtlinien für einen vom Kunden verwalteten Schlüssel im Beispielkonto und in der Region abgerufen. Sie können diesen Befehl verwenden, um die Namen der wichtigsten Richtlinien für AWS verwaltete Schlüssel und vom Kunden verwaltete Schlüssel zu finden.

Da der einzig gültige Name der Schlüsselrichtlinie lautet`default`, ist dieser Befehl nicht nützlich.

Verwenden Sie den `key-id` Parameter, um den KMS-Schlüssel anzugeben. In diesem Beispiel wird ein Schlüssel-ID-Wert verwendet, aber Sie können in diesem Befehl auch eine Schlüssel-ID oder einen Schlüssel-ARN verwenden.

```
aws kms list-key-policies \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Ausgabe:

```
{
    "PolicyNames": [
    "default"
    ]
}
```

Weitere Informationen zu AWS KMS-Schlüsselrichtlinien finden Sie unter Verwenden von Schlüsselrichtlinien in AWS KMS im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie ListKeyPoliciesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously retrieves the key policy for the specified key ID and
policy name.
    *
    * @param keyId       the ID of the AWS KMS key for which to retrieve the
policy
    * @param policyName the name of the key policy to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
    */
   public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
       GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
           .keyId(keyId)
           .policyName(policyName)
           .build();

       return getAsyncClient().getKeyPolicy(policyRequest)
           .thenApply(response -> {
               String policy = response.policy();
               logger.info("The response is: " + policy);
               return policy;
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Failed to get key policy", ex);
           });
   }
```

- Einzelheiten zur API finden Sie ListKeyPoliciesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)[
                "PolicyNames"
            ]
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- Einzelheiten zur API finden Sie ListKeyPoliciesin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ListKeys`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListKeys` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// List the AWS Key Managements Service (AWS KMS) keys for the AWS Region
```

```
    /// of the default user. To list keys in another AWS Region, supply the
Region
    /// as a parameter to the client constructor.
    /// </summary>
    public class ListKeys
    {
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();
            var request = new ListKeysRequest();
            var response = new ListKeysResponse();

            do
            {
                response = await client.ListKeysAsync(request);

                response.Keys.ForEach(key =>
                {
                    Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
                });

                // Set the Marker property when response.Truncated is true
                // in order to get the next keys.
                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie die KMS-Schlüssel für ein Konto und eine Region ab

Im folgenden `list-keys` Beispiel werden die KMS-Schlüssel für ein Konto und eine Region abgerufen. Dieser Befehl gibt sowohl AWS verwaltete Schlüssel als auch vom Kunden verwaltete Schlüssel zurück.

```
aws kms list-keys
```

Ausgabe:

```
{
    "Keys": [
        {
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
            "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
        },
        {
            "KeyArn": "arn:aws:kms:us-
east-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
            "KeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        }
    ]
}
```

Weitere Informationen finden Sie unter Schlüssel anzeigen im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie ListKeysunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
```

```
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
 Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
```

```
                    System.err.println("Error occurred: " +
 exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeysin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            key_paginator = self.kms_client.get_paginator("list_keys")
            for key_page in key_paginator.paginate(PaginationConfig={"PageSize":
 10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page["Keys"])
                if key_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} keys (y/n)? "
```

```
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s",
                err.response["Error"]["Message"],
            )
```

- Einzelheiten zur API finden Sie ListKeysin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListKeysin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`PutKeyPolicy`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutKeyPolicy` verwendet wird.

CLI

AWS CLI

Um die Schlüsselrichtlinie für einen KMS-Schlüssel zu ändern

Im folgenden `put-key-policy` Beispiel wird die Schlüsselrichtlinie für einen vom Kunden verwalteten Schlüssel geändert.

Erstellen Sie zunächst eine Schlüsselrichtlinie und speichern Sie sie in einer lokalen JSON-Datei. In diesem Beispiel ist die Dateikey_policy.json. Sie können die Schlüsselrichtlinie auch als Zeichenfolgenwert des `policy` Parameters angeben.

Die erste Anweisung in dieser Schlüsselrichtlinie erteilt dem AWS Konto die Erlaubnis, IAM-Richtlinien zur Steuerung des Zugriffs auf den KMS-Schlüssel zu verwenden. Die zweite Anweisung erteilt dem `test-user` Benutzer die Erlaubnis, die `list-keys` Befehle `describe-key` und auf dem KMS-Schlüssel auszuführen.

Inhalt von `key_policy.json`:

```
{
    "Version" : "2012-10-17",
    "Id" : "key-default-1",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:root"
            },
```

```
            "Action" : "kms:*",
            "Resource" : "*"
        },
        {
            "Sid" : "Allow Use of Key",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:user/test-user"
            },
            "Action" : [
                "kms:DescribeKey",
                "kms:ListKeys"
            ],
            "Resource" : "*"
        }
    ]
}
```

Um den KMS-Schlüssel zu identifizieren, verwendet dieses Beispiel die Schlüssel-ID, Sie können aber auch einen Schlüssel-ARN verwenden. Um die Schlüsselrichtlinie anzugeben, verwendet der Befehl den `policy` Parameter. Um anzugeben, dass sich die Richtlinie in einer Datei befindet, wird das erforderliche `file://` Präfix verwendet. Dieses Präfix ist erforderlich, um Dateien auf allen unterstützten Betriebssystemen zu identifizieren. Schließlich verwendet der Befehl den `policy-name` Parameter mit dem Wert`default`. Wenn kein Richtlinienname angegeben ist, ist der Standardwert`default`. Der einzige gültige Wert ist `default`.

```
aws kms put-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --policy file://key_policy.json
```

Dieser Befehl erzeugt keine Ausgabe. Verwenden Sie den Befehl, um zu überprüfen, ob der `get-key-policy` Befehl wirksam war. Mit dem folgenden Beispielbefehl wird die Schlüsselrichtlinie für denselben KMS-Schlüssel abgerufen. Der `output` Parameter mit dem Wert von `text` gibt ein Textformat zurück, das leicht zu lesen ist.

```
aws kms get-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --output text
```

Ausgabe:

```
{
    "Version" : "2012-10-17",
    "Id" : "key-default-1",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:root"
            },
            "Action" : "kms:*",
            "Resource" : "*"
            },
            {
            "Sid" : "Allow Use of Key",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:user/test-user"
            },
            "Action" : [ "kms:Describe", "kms:List" ],
            "Resource" : "*"
        }
    ]
}
```

Weitere Informationen finden Sie unter Ändern einer Schlüsselrichtlinie im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie PutKeyPolicyunter AWS CLI Befehlsreferenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie PutKeyPolicyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
```

```
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
existing
        policy, so care is taken to add a statement to the existing list of
statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: The existing policy of the key.
        :return: None
        """
        principal = input(
            "Enter the ARN of an IAM role to set as the principal on the policy:
"
        )
        if key_id != "" and principal != "":
            # The updated policy replaces the existing policy. Add a new
statement to
            # the list along with the original policy statements.
            policy["Statement"].append(
                {
                    "Sid": "Allow access for ExampleRole",
                    "Effect": "Allow",
                    "Principal": {"AWS": principal},
                    "Action": [
                        "kms:Encrypt",
                        "kms:GenerateDataKey*",
                        "kms:Decrypt",
                        "kms:DescribeKey",
                        "kms:ReEncrypt*",
                    ],
                    "Resource": "*",
                }
            )
```

```
            try:
                self.kms_client.put_key_policy(KeyId=key_id,
Policy=json.dumps(policy))
            except ClientError as err:
                logger.error(
                    "Couldn't set policy for key %s. Here's why %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                print(f"Set policy for key {key_id}.")
        else:
            print("Skipping set policy demo.")
```

- Einzelheiten zur API finden Sie PutKeyPolicyin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ReEncrypt`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ReEncrypt` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um eine verschlüsselte Nachricht unter einem anderen symmetrischen KMS-Schlüssel (Linux und macOS) erneut zu verschlüsseln.

Das folgende `re-encrypt` Befehlsbeispiel zeigt die empfohlene Methode zum erneuten Verschlüsseln von Daten mit der AWS CLI.

Geben Sie den Geheimtext in einer Datei an. Verwenden Sie im Wert des `--ciphertext-blob` Parameters das `fileb://` Präfix, das die CLI anweist, die Daten aus einer Binärdatei zu lesen. Wenn sich die Datei nicht im aktuellen Verzeichnis befindet, geben Sie den vollständigen Dateipfad ein. Weitere Informationen zum Lesen von AWS CLI-Parameterwerten

aus einer Datei finden Sie unter Laden von AWS CLI-Parametern aus einer Datei im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle und Best Practices für lokale Dateiparameter im AWS Command Line Tool-Blog. Geben Sie den KMS-Quellschlüssel an, der den Chiffretext entschlüsselt. Der `--source-key-id` Parameter ist bei der Entschlüsselung mit symmetrischer Verschlüsselung von KMS-Schlüsseln nicht erforderlich. AWS KMS kann den KMS-Schlüssel, der zur Verschlüsselung der Daten verwendet wurde, aus den Metadaten im Chiffretext-Blob abrufen. Es ist jedoch immer eine bewährte Methode, den von Ihnen verwendeten KMS-Schlüssel anzugeben. Diese Vorgehensweise stellt sicher, dass Sie den KMS-Schlüssel verwenden, den Sie beabsichtigen, und verhindert, dass Sie versehentlich einen Chiffretext mit einem KMS-Schlüssel entschlüsseln, dem Sie nicht vertrauen. Geben Sie den KMS-Zielschlüssel an, mit dem die Daten erneut verschlüsselt werden. Der Parameter ist immer erforderlich. `--destination-key-id` In diesem Beispiel wird ein Schlüssel-ARN verwendet, Sie können jedoch jeden gültigen Schlüsselbezeichner verwenden.Fordern Sie die Klartext-Ausgabe als Textwert an. Der `--query` Parameter weist die CLI an, nur den Wert des Felds aus der Ausgabe abzurufen. `Plaintext` Der `--output` Parameter gibt die Ausgabe als Text zurück. Base64-dekodieren Sie den Klartext und speichern Sie ihn in einer Datei. Im folgenden Beispiel wird der Wert des Parameters (|) an das Base64-Hilfsprogramm übergeben, das ihn dekodiert. `Plaintext` Anschließend leitet er die dekodierte Ausgabe in die Datei um (**>**). `ExamplePlaintext`

Bevor Sie diesen Befehl ausführen, ersetzen Sie den Beispielschlüssel IDs durch gültige Schlüsselkennungen aus Ihrem AWS Konto.

```
aws kms re-encrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --query CiphertextBlob \
    --output text | base64 --decode > ExampleReEncryptedFile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Die Ausgabe des `re-encrypt` Befehls wird base64-dekodiert und in einer Datei gespeichert.

Weitere Informationen finden Sie ReEncryptin der API-Referenz für den AWS Key Management Service.

Beispiel 2: Um eine verschlüsselte Nachricht unter einem anderen symmetrischen KMS-Schlüssel erneut zu verschlüsseln (Windows-Eingabeaufforderung).

Das folgende `re-encrypt` Befehlsbeispiel ist dasselbe wie das vorherige, außer dass es das `certutil` Hilfsprogramm zur Base64-Decodierung der Klartextdaten verwendet. Für dieses Verfahren sind zwei Befehle erforderlich, wie in den folgenden Beispielen gezeigt.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto.

```
aws kms re-encrypt ^
    --ciphertext-blob fileb://ExampleEncryptedFile ^
    --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
    --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 ^
    --query CiphertextBlob ^
    --output text > ExampleReEncryptedFile.base64
```

Verwenden Sie dann das `certutil` Hilfsprogramm

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

Ausgabe:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Weitere Informationen finden Sie ReEncryptin der API-Referenz für den AWS Key Management Service.

- Einzelheiten zur API finden Sie ReEncryptunter AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
```

```
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def re_encrypt(self, source_key_id, cipher_text):
        """
        Takes ciphertext previously encrypted with one key and reencrypt it by
using
        another key.

        :param source_key_id: The ARN or ID of the original key used to encrypt
the
                              ciphertext.
        :param cipher_text: The encrypted ciphertext.
        :return: The ciphertext encrypted by the second key.
        """
        destination_key_id = input(
            f"Your ciphertext is currently encrypted with key {source_key_id}. "
            f"Enter another key ID or ARN to reencrypt it: "
        )
        if destination_key_id != "":
            try:
                cipher_text = self.kms_client.re_encrypt(
                    SourceKeyId=source_key_id,
                    DestinationKeyId=destination_key_id,
                    CiphertextBlob=cipher_text,
                )["CiphertextBlob"]
            except ClientError as err:
                logger.error(
                    "Couldn't reencrypt your ciphertext. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            else:
```

```
                print(f"Reencrypted your ciphertext as: {cipher_text}")
                return cipher_text
        else:
            print("Skipping reencryption demo.")
```

- Einzelheiten zur API finden Sie ReEncryptin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593
sourceCiphertextBlob = [blob].pack('H*')

# Replace the fictitious key ARN with a valid key ID

destinationKeyId = 'arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.re_encrypt({
                           ciphertext_blob: sourceCiphertextBlob,
                           destination_key_id: destinationKeyId
                         })

# Display a readable version of the resulting re-encrypted blob.
puts 'Blob:'
```

```
puts resp.ciphertext_blob.unpack('H*')
```

- Einzelheiten zur API finden Sie ReEncryptin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn reencrypt_string(
    verbose: bool,
    client: &Client,
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client
        .re_encrypt()
        .ciphertext_blob(data.unwrap())
        .source_key_id(first_key)
        .destination_key_id(new_key)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();
```

```
    let s = base64::encode(bytes);
    let o = &output_file;

    let mut ofile = File::create(o).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {}:", output_file);
        println!("{}", s);
    } else {
        println!("Wrote base64-encoded output to {}", output_file);
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ReEncryptin der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`RetireGrant`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RetireGrant` verwendet wird.

CLI

AWS CLI

Um einen Zuschuss für einen Kundenhauptschlüssel zurückzuziehen

Im folgenden `retire-grant` Beispiel wird ein Zuschuss aus einem KMS-Schlüssel gelöscht.

Der folgende Beispielbefehl spezifiziert die `key-id` Parameter `grant-id` und. Der Wert des `key-id` Parameters muss der Schlüssel-ARN des KMS-Schlüssels sein.

```
aws kms retire-grant \
    --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den `list-grants` Befehl, um zu bestätigen, dass der Zuschuss zurückgezogen wurde.

Weitere Informationen finden Sie im AWS Key Management Service Developer Guide unter Zurückziehen und Widerrufen von Zuschüssen.

- Einzelheiten zur API finden Sie unter RetireGrant AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def retire_grant(self, grant):
        """
        Retires a grant so that it can no longer be used.

        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant["GrantToken"])
```

```
        except ClientError as err:
            logger.error(
                "Couldn't retire grant %s. Here's why: %s",
                grant["GrantId"],
                err.response["Error"]["Message"],
            )
        else:
            print(f"Grant {grant['GrantId']} retired.")
```

- Einzelheiten zur API finden Sie RetireGrantin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`RevokeGrant`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RevokeGrant` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

CLI

AWS CLI

Um eine Erteilung für einen Kundenhauptschlüssel zu widerrufen

Im folgenden `revoke-grant` Beispiel wird ein Zuschuss aus einem KMS-Schlüssel gelöscht. Der folgende Beispielbefehl spezifiziert die `key-id` Parameter `grant-id` und. Der Wert des `key-id` Parameters kann die Schlüssel-ID oder der Schlüssel-ARN des KMS-Schlüssels sein.

```
aws kms revoke-grant \
    --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den `list-grants` Befehl, um zu bestätigen, dass der Zuschuss widerrufen wurde.

Weitere Informationen finden Sie im AWS Key Management Service Developer Guide unter Zurückziehen und Widerrufen von Zuschüssen.

- Einzelheiten zur API finden Sie unter RevokeGrant AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
```

```
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }
```

- Einzelheiten zur API finden Sie RevokeGrantin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
```

```
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie RevokeGrantin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie RevokeGrantin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`ScheduleKeyDeletion`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ScheduleKeyDeletion` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

CLI

AWS CLI

Um das Löschen eines vom Kunden verwalteten KMS-Schlüssels zu planen.

Im folgenden `schedule-key-deletion` Beispiel wird geplant, dass der angegebene vom Kunden verwaltete KMS-Schlüssel innerhalb von 15 Tagen gelöscht wird.

Der `--key-id` Parameter identifiziert den KMS-Schlüssel. In diesem Beispiel wird ein ARN-Schlüsselwert verwendet, Sie können jedoch entweder die Schlüssel-ID oder den ARN des KMS-Schlüssels verwenden. Der `--pending-window-in-days` Parameter gibt die Länge der Wartezeit von 7 bis 30 Tagen an. Standardmäßig beträgt die Wartezeit 30 Tage. In diesem Beispiel wird der Wert 15 angegeben, der angibt, dass der KMS-Schlüssel 15 Tage nach Abschluss des Befehls dauerhaft gelöscht werden AWS soll.

```
aws kms schedule-key-deletion \
    --key-id arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --pending-window-in-days 15
```

Die Antwort enthält den Schlüssel-ARN, den Schlüsselstatus, die Wartezeit (`PendingWindowInDays`) und das Löschdatum in Unix-Zeit. Verwenden Sie die AWS KMS-Konsole, um das Löschdatum in Ortszeit anzuzeigen. KMS-Schlüssel im `PendingDeletion` Schlüsselstatus können nicht für kryptografische Operationen verwendet werden.

```
{
    "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "DeletionDate": "2022-06-18T23:43:51.272000+00:00",
    "KeyState": "PendingDeletion",
    "PendingWindowInDays": 15
}
```

Weitere Informationen finden Sie unter Löschen von Schlüsseln im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie ScheduleKeyDeletionunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
    * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted
using the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when
deleting KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion
is scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
           .build();

       return getAsyncClient().scheduleKeyDeletion(deletionRequest)
           .thenRun(() -> {
               logger.info("Key {} will be deleted in 7 days", keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to schedule key deletion for
key ID: " + keyId, throwable);
           });
   }
```

- Einzelheiten zur API finden Sie ScheduleKeyDeletionin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie ScheduleKeyDeletionin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
 When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
 deleted.
        """

        try:
```

```
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ScheduleKeyDeletionin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Sign`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Sign` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: Um eine digitale Signatur für eine Nachricht zu generieren

Das folgende `sign` Beispiel generiert eine kryptografische Signatur für eine Kurznachricht. Die Ausgabe des Befehls enthält ein Base-64-codiertes `Signature` Feld, das Sie mithilfe des Befehls überprüfen können. `verify`

Sie müssen eine zu signierende Nachricht und einen Signierungsalgorithmus angeben, den Ihr asymmetrischer KMS-Schlüssel unterstützt. Verwenden Sie den `describe-key` Befehl, um die Signaturalgorithmen für Ihren KMS-Schlüssel abzurufen.

In AWS CLI v2 muss der Wert des `message` Parameters Base64-codiert sein. Oder Sie können die Nachricht in einer Datei speichern und das `fileb://` Präfix verwenden, das die AWS CLI anweist, Binärdaten aus der Datei zu lesen.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto. Die Schlüssel-ID muss einen asymmetrischen KMS-Schlüssel mit der Schlüsselverwendung SIGN_VERIFY darstellen.

```
msg=(echo 'Hello World' | base64)

aws kms sign \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://UnsignedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256
```

Ausgabe:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "Signature": "ABCDEFhpyVYyTxbafE74ccSvEJLJr3zuoV1Hfymz4qv+/
fxmxNLA7SE1SiF8lHw80fKZZ3bJ...",
    "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

Weitere Informationen zur Verwendung asymmetrischer KMS-Schlüssel in AWS KMS finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 2: Um eine digitale Signatur in einer Datei zu speichern (Linux und macOs)

Das folgende `sign` Beispiel generiert eine kryptografische Signatur für eine Kurznachricht, die in einer lokalen Datei gespeichert ist. Der Befehl ruft auch die `Signature` Eigenschaft aus der Antwort ab, dekodiert sie mit Base64 und speichert sie in der Datei. ExampleSignature Sie können die Signaturdatei in einem `verify` Befehl verwenden, der die Signatur überprüft.

Für den `sign` Befehl sind eine Base64-codierte Nachricht und ein Signaturalgorithmus erforderlich, den Ihr asymmetrischer KMS-Schlüssel unterstützt. Verwenden Sie den Befehl, um die Signaturalgorithmen abzurufen, die Ihr KMS-Schlüssel unterstützt. `describe-key`

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto. Die Schlüssel-ID muss einen asymmetrischen KMS-Schlüssel mit der Schlüsselverwendung SIGN_VERIFY darstellen.

```
echo 'hello world' | base64 > EncodedMessage

aws kms sign \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://EncodedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
    --output text \
    --query Signature | base64 --decode > ExampleSignature
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. In diesem Beispiel wird die `Signature` Eigenschaft der Ausgabe extrahiert und in einer Datei gespeichert.

Weitere Informationen zur Verwendung asymmetrischer KMS-Schlüssel in AWS KMS finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie unter AWS CLI Befehlsreferenz für die Anmeldung.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
```

```
    * <ol>
    *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
    *     <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
    *     <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
    * </ol>
    *
    * @return a {@link CompletableFuture} that completes with the result of the
signature verification,
    *         {@code true} if the signature is valid, {@code false} otherwise.
    * @throws KmsException if any error occurs during the KMS operations.
    * @throws RuntimeException if an unexpected error occurs.
    */
   public CompletableFuture<Boolean> signVerifyDataAsync() {
       String signMessage = "Here is the message that will be digitally signed";

       // Create an AWS KMS key used to digitally sign data.
       CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
           .keySpec(KeySpec.RSA_2048)
           .keyUsage(KeyUsageType.SIGN_VERIFY)
           .origin(OriginType.AWS_KMS)
           .build();

       return getAsyncClient().createKey(createKeyRequest)
           .thenCompose(createKeyResponse -> {
               String keyId = createKeyResponse.keyMetadata().keyId();

               SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
               SignRequest signRequest = SignRequest.builder()
                   .keyId(keyId)
                   .message(messageBytes)
                   .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                   .build();

               return getAsyncClient().sign(signRequest)
                   .thenCompose(signResponse -> {
                       byte[] signedBytes =
signResponse.signature().asByteArray();

                       VerifyRequest verifyRequest = VerifyRequest.builder()
                           .keyId(keyId)
```

```
.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
   }
```

- Einzelheiten zur API finden Sie unter AWS SDK for Java 2.x API-Referenz für die Anmeldung.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/***
 * @param string $keyId
 * @param string $message
 * @param string $algorithm
 * @return Result
 */
```

```
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie unter AWS SDK für PHP API-Referenz für die Anmeldung.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
```

```
        return cls(kms_client)


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie unter API-Referenz zum AWS AnmeldesDK SDK for Python (Boto3).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`TagResource`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erlernen der Grundlagen

CLI

AWS CLI

Um einem KMS-Schlüssel ein Tag hinzuzufügen

Im folgenden `tag-resource` Beispiel wird ein vom Kunden verwalteter KMS-Schlüssel hinzugefügt `"Purpose":"Test"` und mit `"Dept":"IT"` Tags versehen. Sie können Tags wie diese verwenden, um KMS-Schlüssel zu kennzeichnen und Kategorien von KMS-Schlüsseln für Berechtigungen und Prüfungen zu erstellen.

Verwenden Sie den `key-id` Parameter, um den KMS-Schlüssel anzugeben. In diesem Beispiel wird ein Schlüssel-ID-Wert verwendet, aber Sie können in diesem Befehl auch eine Schlüssel-ID oder einen Schlüssel-ARN verwenden.

```
aws kms tag-resource \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --tags TagKey='Purpose',TagValue='Test' TagKey='Dept',TagValue='IT'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den `list-resource-tags` Befehl, um die Tags auf einem AWS KMS-KMS-Schlüssel anzuzeigen.

Weitere Informationen zur Verwendung von Tags in AWS KMS finden Sie unter Tagging Keys im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
```

```
     * @return a {@link CompletableFuture} that completes when the tagging
operation is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key",
throwable);
            });
    }
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param array $tags
     * @return void
```

```
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
```

```
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie TagResourcein AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`UpdateAlias`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAlias` verwendet wird.

CLI

AWS CLI

Um einen Alias einem anderen KMS-Schlüssel zuzuordnen

Im folgenden `update-alias` Beispiel wird der Alias `alias/test-key` einem anderen KMS-Schlüssel zugeordnet.

Der `--alias-name` Parameter gibt den Alias an. Der Wert des Aliasnamens muss mit `alias/` .beginnen. Der `--target-key-id` Parameter gibt den KMS-Schlüssel an, der dem Alias zugeordnet werden soll. Sie müssen den aktuellen KMS-Schlüssel für den Alias nicht angeben.

```
aws kms update-alias \
    --alias-name alias/test-key \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den `list-aliases` Befehl, um den Alias zu finden.

Weitere Informationen finden Sie unter Aliase aktualisieren im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie unter UpdateAlias AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.
```

```
        :return: An instance of AliasManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def update_alias(self, alias, current_key_id):
        """
        Updates an alias by assigning it to another key.

        :param alias: The alias to reassign.
        :param current_key_id: The ARN or ID of the key currently associated with
the alias.
        """
        new_key_id = input(
            f"Alias {alias} is currently associated with {current_key_id}. "
            f"Enter another key ID or ARN that you want to associate with
{alias}: "
        )
        if new_key_id != "":
            try:
                self.kms_client.update_alias(AliasName=alias,
TargetKeyId=new_key_id)
            except ClientError as err:
                logger.error(
                    "Couldn't associate alias %s with key %s. Here's why: %s",
                    alias,
                    new_key_id,
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Alias {alias} is now associated with key {new_key_id}.")
        else:
            print("Skipping alias update.")
```

- Einzelheiten zur API finden Sie UpdateAliasin AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung **`Verify`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Verify` verwendet wird.

CLI

AWS CLI

Um eine digitale Signatur zu verifizieren

Mit dem folgenden `verify` Befehl wird eine kryptografische Signatur für eine kurze, Base64-kodierte Nachricht überprüft. Die Schlüssel-ID, die Nachricht, der Nachrichtentyp und der Signaturalgorithmus müssen dieselben sein, die zum Signieren der Nachricht verwendet wurden.

In AWS CLI v2 muss der Wert des `message` Parameters Base64-codiert sein. Oder Sie können die Nachricht in einer Datei speichern und das `fileb://` Präfix verwenden, das die AWS CLI anweist, Binärdaten aus der Datei zu lesen.

Die von Ihnen angegebene Signatur kann nicht Base64-codiert sein. Hilfe zur Dekodierung der Signatur, die der `sign` Befehl zurückgibt, finden Sie in den Befehlsbeispielen. `sign`

Die Ausgabe des Befehls enthält ein boolesches `SignatureValid` Feld, das angibt, dass die Signatur verifiziert wurde. Wenn die Signaturüberprüfung fehlschlägt, schlägt auch der `verify` Befehl fehl.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto.

```
aws kms verify \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://EncodedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
    --signature fileb://ExampleSignature
```

Ausgabe:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "SignatureValid": true,
    "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

Weitere Informationen zur Verwendung asymmetrischer KMS-Schlüssel in AWS KMS finden Sie unter Using asymmetric keys im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie unter Überprüfen in der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.
```

```
        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie unter Verify in AWS SDK for Python (Boto3) API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. Verwenden dieses Dienstes mit einem AWS SDK Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für die AWS KMS Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in AWS KMS with implementieren AWS SDKs. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS KMS oder in Kombination mit anderen aufrufen AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Arbeiten Sie mit DynamoDB-Tabellenverschlüsselung mit Version 2 AWS Command Line Interface

Arbeiten Sie mit DynamoDB-Tabellenverschlüsselung mit Version 2 AWS Command Line Interface

Das folgende Codebeispiel zeigt, wie Verschlüsselungsoptionen für DynamoDB-Tabellen verwaltet werden.

- Erstellen Sie eine Tabelle mit Standardverschlüsselung.
- Erstellen Sie eine Tabelle mit einem vom Kunden verwalteten CMK.
- Aktualisieren Sie die Einstellungen für die Tabellenverschlüsselung.
- Beschreiben Sie die Tabellenverschlüsselung.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit Standardverschlüsselung.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Erstellen Sie eine Tabelle mit einem vom Kunden verwalteten CMK.

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
```

```
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Aktualisieren Sie die Tabellenverschlüsselung.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Beschreiben Sie die Tabellenverschlüsselung.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- API-Details finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateKey
  - CreateTable
  - DescribeTable
  - UpdateTable

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unterVerwenden dieses Dienstes mit einem AWS SDK. Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Kryptografische Bescheinigung für Nitro-Enklaven AWS

AWS KMS unterstützt die kryptografische Bescheinigung für Nitro Enclaves.AWS Anwendungen, die AWS Nitro Enclaves unterstützen, rufen die folgenden AWS KMS kryptografischen Operationen mit einem signierten Bestätigungsdokument für die Enklave auf. Diese AWS KMS APIs bestätigen, dass das Bescheinigungsdokument aus einer Nitro-Enklave stammt. Anstatt in der Antwort Klartextdaten zurückzugeben, APIs verschlüsseln sie den Klartext mit dem öffentlichen Schlüssel aus dem Beglaubigungsdokument und geben Chiffretext zurück, der nur mit dem entsprechenden privaten Schlüssel in der Enklave entschlüsselt werden kann.

- Decrypt
- DeriveSharedSecret
- GenerateDataKey
- GenerateDataKeyPair
- GenerateRandom

Die folgende Tabelle zeigt, wie sich die Antwort auf Nitro-Enclave-Anfragen von der Standardantwort für jeden API-Vorgang unterscheidet.

AWS KMS Operation	Standardantwort	Antwort für AWS Nitro Enclaves
`Decrypt`	Gibt Klartextdaten zurück	Gibt die Klartextdaten zurück, die durch den öffentlichen Schlüssel aus dem Bescheinigungsdokument verschlüsselt wurden
`DeriveSharedSecret`	Gibt rohes Shared Secret zurück	Gibt das rohe gemeinsame geheime Geheimnis zurück, das mit dem öffentlichen Schlüssel aus dem Bescheinigungsdokument verschlüsselt wurde

AWS KMS Operation	Standardantwort	Antwort für AWS Nitro Enclaves
`GenerateDataKey`	Gibt eine Klartextkopie des Datenschlüssels zurück (Gibt auch eine Kopie des Datenschlüssels zurück, der durch einen KMS-Schlüssel verschlüsselt wurde)	Liefert eine Kopie des mit dem öffentlichen Schlüssel des Bescheinigungsdokuments verschlüsselten Datenschl üssels (Gibt auch eine Kopie des Datenschlüssels zurück, der durch einen KMS-Schlüssel verschlüsselt wurde)
`GenerateDataKeyPair`	Gibt eine Klartextkopie des privaten Schlüssels zurück (Gibt auch den öffentlichen Schlüssel und eine Kopie des privaten Schlüssels zurück, der mit einem KMS-Schlüssel verschlüsselt ist)	Gibt eine Kopie des privaten Schlüssels zurück, der mit dem öffentlichen Schlüssel des Bescheinigungsdoku ments verschlüsselt wurde (Gibt auch den öffentlichen Schlüssel und eine Kopie des privaten Schlüssels zurück, der mit einem KMS-Schlüssel verschlüsselt ist)
`GenerateRandom`	Gibt eine zufällige Byte-Zeic henfolge zurück	Gibt die mit dem öffentlic hen Schlüssel verschlüsselte zufällige Bytefolge aus dem Bescheinigungsdokument zurück

AWS KMS unterstützt Richtlinien-Bedingungsschlüssel, mit denen Sie Enklavenoperationen mit einem auf dem Inhalt des AWS KMS Bestätigungsdokuments basierenden Schlüssel zulassen oder verweigern können. Sie können Anfragen an Ihre Nitro-Enklave auch in Ihren Protokollen überwachen. AWS KMS AWS CloudTrail

Weitere Informationen

- Kryptografische Bescheinigung
- AWS KMS Bedingungsschlüssel für AWS Nitro Enclaves
- Wie ruft man nach einer AWS KMS APIs Nitro-Enklave
- Überwachung von Anfragen für Nitro-Enklaven

Wie ruft man nach einer AWS KMS APIs Nitro-Enklave

Um eine Nitro-Enklave aufzurufen AWS KMS APIs , verwenden Sie den `Recipient` Parameter in der Anfrage, um das signierte Bestätigungsdokument für die Enklave und den Verschlüsselungsalgorithmus bereitzustellen, der mit dem öffentlichen Schlüssel der Enklave verwendet werden soll. Wenn eine Anfrage den `Recipient`-Parameter mit einem signierten Beglaubigungsdokument enthält, enthält die Antwort ein `CiphertextForRecipient`-Feld mit dem Geheimtext, der mit dem öffentlichen Schlüssel verschlüsselt ist. Das Klartextfeld ist Null oder leer.

Der `Recipient` Parameter muss ein signiertes Beglaubigungsdokument aus einer Nitro-Enklave angeben. AWS AWS KMS stützt sich auf die digitale Signatur für das Beglaubigungsdokument der Enklave, um nachzuweisen, dass der öffentliche Schlüssel in der Anfrage aus einer gültigen Enklave stammt. Sie können kein eigenes Zertifikat angeben, um das Bescheinigungsdokument digital zu signieren.

Verwenden Sie zur Angabe des `Recipient`-Parameters das AWS Nitro Enclaves SDK oder ein beliebiges AWS -SDK. Das AWS Nitro Enclaves SDK, das nur innerhalb einer Nitro-Enklave unterstützt wird, fügt den Parameter und seine Werte automatisch zu jeder Anfrage hinzu. `Recipient` AWS KMS Um Anfragen für Nitro-Enklaven in der zu stellen AWS SDKs, müssen Sie den Parameter und seine Werte angeben. `Recipient` Die Support für die kryptografische Bescheinigung der Nitro-Enklave in der AWS SDKs wurde im März 2023 eingeführt.

AWS KMS unterstützt Richtlinien-Bedingungsschlüssel, die Sie verwenden können, um Enklave-Operationen mit einem auf dem Inhalt des Bestätigungsdokuments basierenden AWS KMS Schlüssel zuzulassen oder zu verweigern. Sie können Anfragen an Ihre Nitro-Enklave auch in Ihren Protokollen überwachen. AWS KMS AWS CloudTrail

Ausführliche Informationen über den `Recipient` Parameter und das `CiphertextForRecipient` AWS-Antwortfeld finden Sie in den GenerateRandomThemen Decrypt DeriveSharedSecret, GenerateDataKey, GenerateDataKeyPair, und in der AWS Key Management Service API-Referenz, im AWS Nitro Enclaves SDK oder in einem beliebigen SDK. AWS Informationen zur Einrichtung

Ihrer Daten und Datenschlüssel für die Verschlüsselung finden Sie unter Verwenden einer kryptografischen Bescheinigung mit. AWS KMS

Überwachung von Anfragen für Nitro-Enklaven

Sie können Ihre AWS CloudTrail Protokolle verwenden, um Decrypt -,, DeriveSharedSecretGenerateDataKey, und GenerateRandom-Operationen für eine GenerateDataKeyPairNitro-Enklave zu überwachen. AWS In diesen Protokolleinträgen enthält das `additionalEventData` Feld ein `recipient` Feld mit der Modul-ID (`attestationDocumentModuleId`), dem Image-Digest (`attestationDocumentEnclaveImageDigest`) und den Plattformkonfigurationsregistern (PCRs) aus dem Bestätigungsdokument in der Anfrage. Diese Felder sind nur enthalten, wenn der `Recipient` Parameter in der Anfrage ein signiertes Bescheinigungsdokument aus einer Nitro-Enklave angibt. AWS

Die Modul-ID ist die Enklave-ID der Nitro-Enklave. Der Image-Digest ist der SHA384 Hash des Enklavenbilds. Sie können den Image-Digest und die PCR-Werte als Bedingungen für Schlüsselrichtlinien und IAM-Richtlinien verwenden. Weitere Informationen zu finden Sie im PCRs AWS Nitro Enclaves-Benutzerhandbuch unter Wo finde ich die Maße einer Enklave?

Dieser Abschnitt zeigt ein Beispiel für einen CloudTrail Protokolleintrag für jede der unterstützten Nitro-Enklave-Anfragen an. AWS KMS

Entschlüsseln (für eine Enklave)

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag eines Decrypt-Vorgangs für eine Nitro-Enklave. AWS

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2020-07-27T22:58:24Z",
    "eventSource": "kms.amazonaws.com",
```

```
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
            "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
            "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
            "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
            "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
            "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
            "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
            "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
        }
    },
    "requestID": "b4a65126-30d5-4b28-98b9-9153da559963",
    "eventID": "e5a2f202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GenerateDataKey (für eine Enklave)

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag eines GenerateDataKeyVorgangs für eine AWS Nitro-Enklave.

```
{
    "eventVersion": "1.02",
```

```
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "numberOfBytes": 32
    },
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
            "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
            "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
            "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
            "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
            "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
            "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
            "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
        }
    },
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
    "readOnly": true,
    "resources": [{
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GenerateDataKeyPair (für eine Enklave)

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag eines GenerateDataKeyPairVorgangs für eine AWS Nitro-Enklave.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
    },
    "eventTime": "2020-07-27T18:57:57Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyPair",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyPairSpec": "RSA_3072",
        "encryptionContext": {
            "Project": "Alpha"
        },
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
            "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
            "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
            "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
            "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
            "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
            "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
            "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
        }
    },
    "requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
    "eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
    "readOnly": true,
```

```
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

GenerateRandom (für eine Enklave)

Das folgende Beispiel zeigt einen AWS CloudTrail Protokolleintrag eines GenerateRandomVorgangs für eine AWS Nitro-Enklave.

```
{
    "eventVersion": "1.02",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2014-11-04T00:52:37Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateRandom",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
        "recipient": {
            "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
            "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
            "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
            "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
            "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
            "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
```

```
            "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
        }
    },
    "requestID": "df1e3de6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "239cb9f7-ae05-4c94-9221-6ea30eef0442",
    "readOnly": true,
    "resources": [],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

AWS KMS Verschlüsselung mit AWS Diensten verwenden

Mit AWS Key Management Service können Sie Verschlüsselungsschlüssel zum Schutz von Daten in anderen AWS Diensten bereitstellen. Die Verwendung von AWS KMS Verschlüsselung mit AWS Diensten bezieht sich auf den Prozess der Integration AWS KMS mit anderen AWS Diensten, um Daten im Ruhezustand oder während der Übertragung zu verschlüsseln und zu entschlüsseln. Entwickler, Systemadministratoren und Sicherheitsexperten könnten an diesem Thema interessiert sein, um sensible Daten zu schützen, die über AWS Dienste gespeichert oder übertragen werden, gesetzliche Anforderungen zu erfüllen oder bewährte Verschlüsselungsmethoden zu implementieren. Zu den häufigsten Anwendungsfällen gehört die Verschlüsselung von Amazon EBS-Volumes, Amazon S3 S3-Buckets und Amazon RDS-Datenbanken. In den folgenden Abschnitten werden die Schritte zur Konfiguration und Verwaltung von AWS KMS Verschlüsselungsschlüsseln für bestimmte AWS Services behandelt, um die Vertraulichkeit und Integrität der Daten in Ihrer gesamten AWS Umgebung zu gewährleisten. Eine vollständige Liste der integrierten AWS Services finden Sie unter Service Integration AWS KMS.AWS

In den folgenden Themen wird detailliert beschrieben, wie bestimmte Dienste diese verwenden AWS KMS, einschließlich der KMS-Schlüssel, die sie unterstützen, wie sie Datenschlüssel verwalten, welche Berechtigungen sie benötigen und wie Sie die Verwendung der KMS-Schlüssel in Ihrem Konto durch die einzelnen Dienste nachverfolgen können.

> **Important**
>
> AWS Dienste, die integriert sind, AWS KMS verwenden ausschließlich symmetrische KMS-Schlüssel zur Verschlüsselung Ihrer Daten. Diese Services unterstützen keine Verschlüsselung mit asymmetrischen KMS-Schlüsseln. Informationen zur Feststellung, ob ein KMS-Schlüssel symmetrisch oder asymmetrisch ist, finden Sie unter Identifizieren Sie verschiedene Schlüsseltypen.

Themen

- So verwendet Amazon Elastic Block Store (Amazon EBS) AWS KMS
- So verwendet Amazon EMR AWS KMS
- So verwendet Amazon Redshift AWS KMS

So verwendet Amazon Elastic Block Store (Amazon EBS) AWS KMS

In diesem Thema wird ausführlich beschrieben, wie Amazon Elastic Block Store (Amazon EBS) AWS KMS Volumes und Snapshots verschlüsselt. Grundlegende Anweisungen zur Verschlüsselung von EBS-Volumes finden Sie unter Amazon EBS-Verschlüsselung.

Themen

- Amazon-EBS-Verschlüsselung
- Verwenden von KMS-Schlüsseln und Datenschlüsseln
- Amazon-EBS-Verschlüsselungskontext
- Erkennen von Amazon-EBS-Fehlern
- Verwenden AWS CloudFormation , um verschlüsselte Amazon EBS-Volumes zu erstellen

Amazon-EBS-Verschlüsselung

Wenn Sie ein verschlüsseltes Amazon EBS-Volume an einen unterstützten Amazon Elastic Compute Cloud (Amazon EC2) -Instance-Typ anhängen, werden die auf dem Volume gespeicherten Daten, Festplatten-I/O und die vom Volume erstellten Snapshots alle verschlüsselt. Die Verschlüsselung erfolgt auf den Servern, die EC2 Amazon-Instances hosten.

Diese Funktion wird für alle Typen von Amazon-EBS-Volumes unterstützt. Sie greifen auf verschlüsselte Volumes genauso zu wie auf andere Volumes. Verschlüsselung und Entschlüsselung werden transparent gehandhabt und erfordern keine zusätzlichen Maßnahmen von Ihnen, Ihrer EC2 Instance oder Ihrer Anwendung. Snapshots von verschlüsselten Volumes werden automatisch verschlüsselt, genau wie Volumes, die auf Basis verschlüsselter Snapshots erstellt werden.

Der Verschlüsselungsstatus eines EBS-Volume wird bei der Erstellung des Volume bestimmt. Der Verschlüsselungsstatus eines vorhandenen Volume kann nicht geändert werden. Sie können jedoch Daten zwischen verschlüsselten und nicht verschlüsselten Volumes migrieren und beim Kopieren eines Snapshots einen neuen Verschlüsselungsstatus anwenden.

Amazon EBS unterstützt standardmäßig optionale Verschlüsselung. Sie können die Verschlüsselung automatisch für alle neuen EBS-Volumes und Snapshot-Kopien in Ihrer Region und Ihrer AWS-Konto Region aktivieren. Diese Konfigurationseinstellung hat keine Auswirkungen auf vorhandene

Volumes oder Snapshots. Einzelheiten finden Sie unter Amazon EBS-Verschlüsselung im Amazon EBS-Benutzerhandbuch.

Verwenden von KMS-Schlüsseln und Datenschlüsseln

Wenn Sie ein verschlüsseltes Amazon-EBS-Volume erstellen, geben Sie einen AWS KMS key an. Amazon EBS verwendet standardmäßig Von AWS verwalteter Schlüssel für Amazon EBS in Ihrem Konto (`aws/ebs`). Sie können jedoch einen kundenverwalteten Schlüssel angeben, den Sie erstellen und verwalten.

Um einen kundenverwalteten Schlüssel zu verwenden, müssen Sie Amazon EBS die Berechtigung zur Verwendung des KMS-Schlüssels in Ihrem Namen erteilen. Eine Liste der erforderlichen Berechtigungen finden Sie unter Berechtigungen für IAM-Benutzer im EC2 Amazon-Benutzerhandbuch oder EC2 Amazon-Benutzerhandbuch.

> **Important**
>
> Amazon EBS unterstützt nur symmetrische KMS-Schlüssel. Sie können keinen asymmetrischen KMS-Schlüssel verwenden, um ein Amazon-EBS-Volume zu verschlüsseln. Informationen zur Feststellung, ob ein KMS-Schlüssel symmetrisch oder asymmetrisch ist, finden Sie unter Identifizieren Sie verschiedene Schlüsseltypen.

Amazon EBS fordert Sie auf, für jedes Volume einen eindeutigen Datenschlüssel AWS KMS zu generieren, der unter dem von Ihnen angegebenen KMS-Schlüssel verschlüsselt ist. Amazon EBS speichert den verschlüsselten Datenschlüssel mit dem Volume. Wenn Sie dann das Volume an eine EC2 Amazon-Instance anhängen, ruft Amazon EBS auf, AWS KMS um den Datenschlüssel zu entschlüsseln. Amazon EBS verwendet den Klartext-Datenschlüssel im Hypervisor-Speicher, um alle Festplatten I/O auf dem Volume zu verschlüsseln. Einzelheiten finden Sie unter So funktioniert die EBS-Verschlüsselung im EC2 Amazon-Benutzerhandbuch oder EC2 Amazon-Benutzerhandbuch.

Amazon-EBS-Verschlüsselungskontext

In seinen GenerateDataKeyWithoutPlaintextund Decrypt-Anfragen an AWS KMS verwendet Amazon EBS einen Verschlüsselungskontext mit einem Name-Wert-Paar, das das Volume oder den Snapshot in der Anfrage identifiziert. Der Name im Verschlüsselungskontext variiert nicht.

Ein Verschlüsselungskontext ist eine Gruppe von Schlüssel/Wert-Paaren mit willkürlichen, nicht geheimen Daten. Wenn Sie einen Verschlüsselungskontext in eine Anfrage zur Verschlüsselung

von Daten einbeziehen, wird der Verschlüsselungskontext AWS KMS kryptografisch an die verschlüsselten Daten gebunden. Zur Entschlüsselung der Daten müssen Sie denselben Verschlüsselungskontext übergeben.

Für alle Volumes und für verschlüsselte Snapshots, die mit dem Amazon CreateSnapshotEBS-Vorgang erstellt wurden, verwendet Amazon EBS die Volume-ID als Wert für den Verschlüsselungskontext. Im `requestParameters`-Feld eines CloudTrail-Protokolleintrags sieht der Verschlüsselungskontext wie folgt aus:

```
"encryptionContext": {
  "aws:ebs:id": "vol-0cfb133e847d28be9"
}
```

Für verschlüsselte Snapshots, die mit dem EC2 CopySnapshotAmazon-Vorgang erstellt wurden, verwendet Amazon EBS die Snapshot-ID als Verschlüsselungskontextwert. Im `requestParameters`-Feld eines CloudTrail-Protokolleintrags sieht der Verschlüsselungskontext wie folgt aus:

```
"encryptionContext": {
  "aws:ebs:id": "snap-069a655b568de654f"
}
```

Erkennen von Amazon-EBS-Fehlern

Um ein verschlüsseltes EBS-Volume zu erstellen oder das Volume an eine EC2 Instance anzuhängen, müssen Amazon EBS und die EC2 Amazon-Infrastruktur in der Lage sein, den KMS-Schlüssel zu verwenden, den Sie für die EBS-Volume-Verschlüsselung angegeben haben. Wenn der KMS-Schlüssel nicht verwendbar ist, z. B. wenn sein Schlüsselstatus nicht `Enabled` ist, schlägt die Volume-Erstellung oder das Anfügen des Volumes fehl.

In diesem Fall sendet Amazon EBS ein Ereignis an Amazon EventBridge (früher CloudWatch Events), um Sie über den Fehler zu informieren. In EventBridge können Sie Regeln festlegen, die als Reaktion auf diese Ereignisse automatische Aktionen auslösen. Weitere Informationen finden Sie unter Amazon CloudWatch Events for Amazon EBS im EC2 Amazon-Benutzerhandbuch, insbesondere in den folgenden Abschnitten:

- Ungültiger Verschlüsselungsschlüssel für das Anfügen oder erneute Anfügen eines Volume
- Ungültiger Verschlüsselungsschlüssel für das Erstellen eines Volume

Um dieser Fehler zu beheben, vergewissern Sie sich, dass der KMS-Schlüssel, den Sie für die EBS-Volume-Verschlüsselung angegeben haben, aktiviert ist. Sehen Sie sich dazu zunächst den KMS-Schlüssel an, um seinen aktuellen Schlüsselstatus zu ermitteln (die Spalte Status in AWS Management Console). Sehen Sie sich dann die Informationen unter einem der folgenden Links an:

- Wenn der Schlüsselstatus des KMS-Schlüssels deaktiviert ist, aktivieren Sie ihn.
- Wenn der Schlüsselstatus des KMS-Schlüssels zeigt, dass der Import ausstehend ist, importieren Sie das Schlüsselmaterial.
- Wenn der Schlüsselstatus des KMS-Schlüssels Löschung ausstehend ist, brechen Sie die Schlüssellöschung ab.

Verwenden AWS CloudFormation , um verschlüsselte Amazon EBS-Volumes zu erstellen

Sie können AWS CloudFormation verwenden, um verschlüsselte Amazon-EBS-Volumes zu erstellen. Weitere Informationen finden Sie unter AWS::EC2::Volume im AWS CloudFormation - Benutzerhandbuch.

So verwendet Amazon EMR AWS KMS

Wenn Sie einen Amazon-EMR-Cluster verwenden, können Sie den Cluster zum Verschlüsseln von Daten im Ruhezustand konfigurieren, bevor Sie sie in einem persistenten Speicher ablegen. Sie können ruhende Daten im EMR-Dateisystem (EMRFS) und/oder auf den Speicher-Volumes von Cluster-Knoten verschlüsseln. Sie können Data-at-Rest mit einem AWS KMS key verschlüsseln. In den folgenden Themen wird erläutert, wie ein Amazon-EMR-Cluster einen KMS-Schlüssel zum Verschlüsseln von Daten im Ruhezustand verwendet.

> **Important**
>
> Amazon EMR unterstützt nur symmetrische KMS-Schlüssel. Sie können keinen asymmetrischen KMS-Schlüssel verwenden, um Data-at-Rest in einem Amazon-EMR-Cluster zu verschlüsseln. Informationen zur Feststellung, ob ein KMS-Schlüssel symmetrisch oder asymmetrisch ist, finden Sie unter Identifizieren Sie verschiedene Schlüsseltypen.

Amazon-EMR-Cluster verschlüsseln auch Daten während der Übertragung, was bedeutet, dass der Cluster Daten vor der Übertragung über das Netzwerk verschlüsselt. Sie können einen KMS-

Schlüssel verwenden, um Daten während der Übertragung zu verschlüsseln. Weitere Information erhalten Sie unter Verschlüsselung von Daten während der Übertragung im Amazon-EMR-Management-Handbuch.

Weitere Informationen über alle in Amazon EMR verfügbaren Verschlüsselungsoptionen finden Sie unter Verschlüsselungsoptionen im Amazon-EMR-Management-Handbuch.

Themen

- Verschlüsseln von Daten auf dem EMR-Dateisystem (EMRFS)
- Verschlüsseln von Daten auf den Speicher-Volumes von Cluster-Knoten
- Verschlüsselungskontext

Verschlüsseln von Daten auf dem EMR-Dateisystem (EMRFS)

Amazon-EMR-Cluster verwenden zwei verteilte Dateisysteme:

- Das Hadoop Distributed File System (HDFS). Die HDFS-Verschlüsselung verwendet keinen KMS-Schlüssel in AWS KMS.
- Das EMR File System (EMRFS). Das EMRFS ist eine Implementierung von HDFS, das es Amazon-EMR-Clustern ermöglicht, Daten in Amazon Simple Storage Service (Amazon S3) zu speichern. EMRFS unterstützt vier Verschlüsselungsoptionen, wovon zwei einen KMS-Schlüssel in AWS KMS verwenden. Weitere Informationen über alle in Amazon EMR verfügbaren Verschlüsselungsoptionen finden Sie unter Verschlüsselungsoptionen im Amazon-EMR-Management-Handbuch.

Die zwei EMRFS-Verschlüsselungsoptionen, die einen KMS-Schlüssel verwenden, um die folgenden von Amazon S3 angebotenen Verschlüsselungsfunktionen zu nutzen:

- Schutz von Daten mithilfe serverseitiger Verschlüsselung mit AWS Key Management Service (SSE-KMS). Der Amazon EMR-Cluster sendet Daten an Amazon S3. Amazon S3 verwendet einen KMS-Schlüssel, um die Daten vor dem Speichern in einem S3-Bucket zu verschlüsseln. Weitere Information dazu finden Sie unter Prozess für das Verschlüsseln von Daten auf EMRFS mit SSE-KMS.
- Schützen von Daten mithilfe clientseitiger Verschlüsselung (CSE-KMS) Daten in einem Amazon EMR werden unter einem AWS KMS key bevor es zur Speicherung an Amazon S3 gesendet wird.

Weitere Information dazu finden Sie unter Prozess für das Verschlüsseln von Daten auf EMRFS mit CSE-KMS.

Wenn Sie einen Amazon-EMR-Cluster konfigurieren, um Daten auf EMRFS mit SSE-KMS oder CSE-KMS zu verschlüsseln, wählen Sie den KMS-Schlüssel in , den Amazon S3 oder der Amazon-EMR-Cluster verwenden soll. Für SSE-KMS können Sie den Von AWS verwalteter Schlüssel für Amazon S3 mit dem Alias aws/s3 oder einen von Ihnen erstellten symmetrischen kundenverwalteten Schlüssel auswählen. Für CSE-KMS müssen Sie einen von Ihnen erstellten symmetrischen kundenverwalteten Schlüssel auswählen. Wenn Sie einen kundenverwalteten Schlüssel auswählen, müssen Sie sicherstellen, dass der Amazon-EMR-Cluster über die Berechtigung zur Nutzung des KMS-Schlüssels verfügt. Weitere Informationen finden Sie unter Verschlüsselung verwenden AWS KMS keys im Amazon EMR Management Guide.

Für beide, SSE-KMS und CSE-KMS, ist der von Ihnen gewählte KMS-Schlüssel der Stammschlüssel in einem Envelope-Verschlüsselungs-Workflow. Die Daten werden mit einem eindeutigen Datenschlüssel verschlüsselt, der unter dem KMS-Schlüssel in AWS KMS verschlüsselt wird. Die verschlüsselten Daten und eine verschlüsselte Kopie des Datenschlüssels werden zusammen als einzelnes verschlüsseltes Objekt in einem S3-Bucket verschlüsselt. Weitere Informationen dazu finden Sie in den folgenden Themen.

Themen

- Prozess für das Verschlüsseln von Daten auf EMRFS mit SSE-KMS
- Prozess für das Verschlüsseln von Daten auf EMRFS mit CSE-KMS

Prozess für das Verschlüsseln von Daten auf EMRFS mit SSE-KMS

Wenn Sie für einen Amazon-EMR-Cluster für die Verwendung von SSE-KMS konfigurieren, läuft der Verschlüsselungsprozess folgendermaßen ab:

1. Der Cluster sendet Daten zu Amazon S3 zur Speicherung in einem S3-Bucket.
2. Amazon S3 sendet eine GenerateDataKeyAnfrage an AWS KMS, in der die Schlüssel-ID des KMS-Schlüssels angegeben wird, den Sie bei der Konfiguration des Clusters für die Verwendung von SSE-KMS ausgewählt haben. Die Anforderung enthält Verschlüsselungskontext; weitere Informationen finden Sie unter Verschlüsselungskontext.

3. AWS KMS generiert einen eindeutigen Datenverschlüsselungsschlüssel (Datenschlüssel) und sendet dann zwei Kopien dieses Datenschlüssels an Amazon S3. Eine Kopie ist unverschlüsselt (Klartext) und die andere Kopie ist mit dem KMS-Schlüssel verschlüsselt.
4. Amazon S3 verschlüsselt die in Schritt 1 erhaltenen Daten mit dem Klartext-Datenschlüssel und entfernt diesen Klartext-Datenschlüssel anschließend schnellstmöglich aus dem Arbeitsspeicher.
5. Amazon S3 speichert die verschlüsselten Daten und eine verschlüsselte Kopie des Datenschlüssels zusammen als einzelnes verschlüsseltes Objekt in einem S3-Bucket.

Die Entschlüsselung funktioniert wie folgt:

1. Der Cluster fordert ein verschlüsseltes Datenobjekt von einem S3-Bucket an.
2. Amazon S3 extrahiert den verschlüsselten Datenschlüssel aus dem S3-Objekt und sendet den verschlüsselten Datenschlüssel dann AWS KMS mit einer Decrypt-Anfrage an. Die Anforderung enthält einen Verschlüsselungskontext.
3. AWS KMS entschlüsselt den verschlüsselten Datenschlüssel mit demselben KMS-Schlüssel, der zur Verschlüsselung verwendet wurde, und sendet dann den entschlüsselten (Klartext-) Datenschlüssel an Amazon S3.
4. Amazon S3 verwendet den Klartext-Datenschlüssel, um die verschlüsselten Daten zu entschlüsseln, und entfernt den Klartext-Datenschlüssel anschließend schnellstmöglich aus dem Arbeitsspeicher.
5. Amazon S3 sendet die entschlüsselten Daten an den Cluster.

Prozess für das Verschlüsseln von Daten auf EMRFS mit CSE-KMS

Wenn Sie für einen Amazon-EMR-Cluster die Verwendung von CSE-KMS konfigurieren, läuft der Verschlüsselungsprozess folgendermaßen ab:

1. Wenn der Cluster bereit ist, Daten in Amazon S3 zu speichern, sendet er eine GenerateDataKeyAnfrage an AWS KMS, in der er die Schlüssel-ID des KMS-Schlüssels angibt, den Sie bei der Konfiguration des Clusters für die Verwendung von CSE-KMS ausgewählt haben. Die Anforderung enthält Verschlüsselungskontext; weitere Informationen finden Sie unter Verschlüsselungskontext.
2. AWS KMS generiert einen eindeutigen Datenverschlüsselungsschlüssel (Datenschlüssel) und sendet dann zwei Kopien dieses Datenschlüssels an den Cluster. Eine Kopie ist unverschlüsselt (Klartext) und die andere Kopie ist mit dem KMS-Schlüssel verschlüsselt.

3. Der Cluster verwendet den Klartext-Datenschlüssel, um die Daten zu verschlüsseln, und entfernt den Klartext-Datenschlüssel anschließend schnellstmöglich aus dem Arbeitsspeicher.
4. Der Cluster kombiniert die verschlüsselten Daten und eine verschlüsselte Kopie des Datenschlüssels in einem einzelnen verschlüsselten Objekt.
5. Der Cluster sendet das verschlüsselte Objekt zum Speichern an Amazon S3.

Die Entschlüsselung funktioniert wie folgt:

1. Der Cluster fordert das verschlüsselte Datenobjekt von einem S3-Bucket an.
2. Amazon S3 sendet das verschlüsselte Objekt an den Cluster.
3. Der Cluster extrahiert den verschlüsselten Datenschlüssel aus dem verschlüsselten Objekt und sendet den verschlüsselten Datenschlüssel dann AWS KMS mit einer Decrypt-Anforderung an. Die Anforderung enthält einen Verschlüsselungskontext.
4. AWS KMS entschlüsselt den verschlüsselten Datenschlüssel mit demselben KMS-Schlüssel, der für die Verschlüsselung verwendet wurde, und sendet dann den entschlüsselten (Klartext-) Datenschlüssel an den Cluster.
5. Der Cluster verwendet den Klartext-Datenschlüssel, um die verschlüsselten Daten zu entschlüsseln, und entfernt den Klartext-Datenschlüssel anschließend schnellstmöglich aus dem Arbeitsspeicher.

Verschlüsseln von Daten auf den Speicher-Volumes von Cluster-Knoten

Ein Amazon EMR-Cluster ist eine Sammlung von Amazon Elastic Compute Cloud (Amazon EC2) -Instances. Jede Instance in einem Cluster wird als Cluster-Knoten oder Knoten bezeichnet. Jeder Knoten kann zwei Arten von Speicher-Volumes besitzen: Instance-Speicher-Volumes und Amazon Elastic Block Store (Amazon EBS)-Volumes. Sie können den Cluster auf die Verwendung von Linux Unified Key Setup (LUKS) einstellen, um beide Arten von Speicher-Volumen auf dem Knoten zu verschlüsseln (aber nicht das Start-Volume jedes Knotens). Dies wird als lokale Laufwerksverschlüsselung bezeichnet.

Wenn Sie die lokale Laufwerksverschlüsselung für einen Cluster aktivieren, können Sie den LUKS-Schlüssel mit einem KMS-Schlüssel in AWS KMS verschlüsseln. Sie müssen einen kundenverwalteten Schlüssel auswählen, den Sie erstellen. Sie können keinen Von AWS verwalteter Schlüssel verwenden. Wenn Sie einen kundenverwalteten Schlüssel auswählen, müssen Sie sicherstellen, dass der Amazon-EMR-Cluster über die Berechtigung zur Nutzung des KMS-

Schlüssels verfügt. Weitere Informationen finden Sie unter Verschlüsselung verwenden AWS KMS keys im Amazon EMR Management Guide.

Wenn Sie die lokale Laufwerksverschlüsselung mit einem KMS-Schlüssel aktivieren, läuft der Verschlüsselungsprozess folgendermaßen ab:

1. Wenn jeder Clusterknoten gestartet wird, sendet er eine GenerateDataKeyAnfrage an AWS KMS, in der die Schlüssel-ID des KMS-Schlüssels angegeben wird, den Sie bei der Aktivierung der lokalen Festplattenverschlüsselung für den Cluster ausgewählt haben.
2. AWS KMS generiert einen eindeutigen Datenverschlüsselungsschlüssel (Datenschlüssel) und sendet dann zwei Kopien dieses Datenschlüssels an den Knoten. Eine Kopie ist unverschlüsselt (Klartext) und die andere Kopie ist unter dem KMS-Schlüssel verschlüsselt.
3. Der Knoten verwendet eine base64-verschlüsselte Version des Klartext-Datenschlüssels als Passwort, um den LUKS-Schlüssel zu schützen. Der Knoten speichert die verschlüsselte Kopie des Datenschlüssel in seinem Start-Volume.
4. Wenn der Knoten neu gestartet wird, sendet der neu gestartete Knoten den verschlüsselten Datenschlüssel AWS KMS mit einer Decrypt-Anfrage an.
5. AWS KMS entschlüsselt den verschlüsselten Datenschlüssel mit demselben KMS-Schlüssel, mit dem er verschlüsselt wurde, und sendet dann den entschlüsselten (Klartext-) Datenschlüssel an den Knoten.
6. Der Knoten verwendet die base64-verschlüsselte Version des Klartext-Datenschlüssels als Passwort, um den LUKS-Schlüssel freizuschalten.

Verschlüsselungskontext

Jeder integrierte AWS Dienst AWS KMS kann einen Verschlüsselungskontext angeben, den der Dienst AWS KMS zum Generieren von Datenschlüsseln oder zum Verschlüsseln oder Entschlüsseln von Daten verwendet. Bei dem Verschlüsselungskontext handelt es sich um zusätzliche authentifizierte Informationen, die zur Überprüfung der Datenintegrität AWS KMS verwendet werden. Wenn ein Service für Verschlüsselungsoperation ein Verschlüsselungskontext angibt, muss der Service denselben Verschlüsselungskontext auch für die entsprechende Entschlüsselungsoperation angeben. Andernfalls schlägt die Entschlüsselung fehl. Der Verschlüsselungskontext wird auch in AWS CloudTrail Protokolldateien geschrieben, sodass Sie leichter nachvollziehen können, warum ein bestimmter KMS-Schlüssel verwendet wurde.

Der folgende Abschnitt erläutert den Verschlüsselungskontext, der in jedem Amazon-EMR-Verschlüsselungsszenario verwendet wird, das einen KMS-Schlüssel nutzt.

Verschlüsselungskontext für die EMRFS-Verschlüsselung mit SSE-KMS

Bei SSE-KMS sendet der Amazon-EMR-Cluster Daten an Amazon S3 und Amazon S3 verwendet dann einen KMS-Schlüssel, um die Daten vor dem Speichern in einem S3-Bucket zu verschlüsseln. In diesem Fall verwendet Amazon S3 den Amazon-Ressourcennamen (ARN) des S3-Objekts als Verschlüsselungskontext für jede GenerateDataKeyDecrypt-Anfrage, an AWS KMS die es sendet. Das folgende Beispiel zeigt eine JSON-Darstellung des von Amazon S3 verwendeten Verschlüsselungskontextes:

```
{ "aws:s3:arn" : "arn:aws:s3:::S3_bucket_name/S3_object_key" }
```

Verschlüsselungskontext für die EMRFS-Verschlüsselung mit CSE-KMS

Bei CSE-KMS verwendet der Amazon-EMR-Cluster einen KMS-Schlüssel, um die Daten vor dem Senden an Amazon S3 für die Speicherung zu verschlüsseln. In diesem Fall verwendet der Cluster den Amazon-Ressourcennamen (ARN) des KMS-Schlüssels als Verschlüsselungskontext für jede GenerateDataKeyDecrypt-Anfrage, an AWS KMS die er sendet. Das folgende Beispiel zeigt eine JSON-Darstellung des vom Cluster verwendeten Verschlüsselungskontextes.

```
{ "kms_cmk_id" : "arn:aws:kms:us-
east-2:111122223333:key/0987ab65-43cd-21ef-09ab-87654321cdef" }
```

Verschlüsselungskontext für die Laufwerksverschlüsselung mit LUKS

Wenn ein Amazon EMR-Cluster die lokale Festplattenverschlüsselung mit LUKS verwendet, geben die Clusterknoten keinen Verschlüsselungskontext mit den GenerateDataKeyund Decrypt-Anfragen an, an die sie senden. AWS KMS

So verwendet Amazon Redshift AWS KMS

In diesem Thema wird erläutert, wie Amazon AWS KMS Redshift Daten verschlüsselt.

Themen

- Amazon-Redshift-Verschlüsselung
- Verschlüsselungskontext

Amazon-Redshift-Verschlüsselung

Ein Amazon Redshift Data Warehouse ist eine Sammlung von Computing-Ressourcen, den sogenannten Knoten, die zu Gruppen, den sogenannten Clustern, zusammengefasst werden. In jedem Cluster wird eine Amazon-Redshift-Engine ausgeführt, und er enthält mindestens eine Datenbank.

Amazon Redshift verwendet zur Verschlüsselung eine schlüsselbasierte Architektur mit vier Ebenen. Diese Architektur besteht aus Datenverschlüsselungsschlüsseln, einem Datenbankschlüssel, einem Clusterschlüssel und einem Stammschlüssel. Sie können einen AWS KMS key als Root-Schlüssel verwenden.

Datenverschlüsselungsschlüssel verschlüsseln Datenblöcke im Cluster. Jedem Datenblock wird ein zufällig generierter AES-256-Schlüssel zugewiesen. Diese Schlüssel werden mithilfe des Datenbankschlüssels des Clusters verschlüsselt.

Der Datenbankschlüssel verschlüsselt Datenverschlüsselungsschlüssel im Cluster. Bei ihm handelt es sich um einen zufällig generierten AES-256-Schlüssel. Er wird auf einem Datenträger in einem separaten Netzwerk außerhalb des Amazon-Redshift-Clusters gespeichert und über einen sicheren Kanal an den Cluster übergeben.

Der Clusterschlüssel verschlüsselt den Datenbankschlüssel des Amazon Redshift-Clusters. Sie können AWS KMS, AWS CloudHSM, oder ein externes Hardware-Sicherheitsmodul (HSM) verwenden, um den Clusterschlüssel zu verwalten. Weitere Details finden Sie im Dokumentationsthema Amazon Redshift-Datenbankverschlüsselung.

Anfordern können Sie die Verschlüsselung durch die Aktivierung des entsprechenden Kontrollkästchens in der Amazon-Redshift-Konsole. In der Liste unter dem Kontrollkästchen für die Verschlüsselung können Sie einen kundenverwalteten Schlüssel auswählen, der verwendet werden soll. Wenn Sie keinen kundenverwalteten Schlüssel angeben, verwendet Amazon Redshift den Von AWS verwalteter Schlüssel für Amazon Redshift unter Ihrem Konto.

> **Important**
>
> Amazon Redshift unterstützt nur KMS-Schlüssel mit symmetrischer Verschlüsselung. Sie können einen asymmetrischen KMS-Schlüssel nicht als in einem Amazon-Redshift-Verschlüsselungs-Workflow verwenden. Informationen zur Feststellung, ob ein KMS-Schlüssel symmetrisch oder asymmetrisch ist, finden Sie unter Identifizieren Sie verschiedene Schlüsseltypen.

Verschlüsselungskontext

Jeder Dienst, der in integriert ist, AWS KMS gibt einen Verschlüsselungskontext an, wenn Datenschlüssel angefordert und verschlüsselt und entschlüsselt werden. Der Verschlüsselungskontext besteht aus zusätzlichen authentifizierten Daten (AAD), die zur Überprüfung der AWS KMS Datenintegrität verwendet werden. Wenn für eine Verschlüsselungsoperation ein Verschlüsselungskontext angegeben wird, gibt der Service denselben Verschlüsselungskontext auch für die Entschlüsselungsoperation an. Andernfalls schlägt die Entschlüsselung fehl. Amazon Redshift gibt die Cluster-ID und den Erstellungszeitpunkt als Verschlüsselungskontext an. Im `requestParameters` Feld einer CloudTrail Protokolldatei sieht der Verschlüsselungskontext in etwa so aus.

```
"encryptionContext": {
    "aws:redshift:arn": "arn:aws:redshift:region:account_ID:cluster:cluster_name",
    "aws:redshift:createtime": "20150206T1832Z"
},
```

Sie können in Ihren CloudTrail Protokollen nach dem Clusternamen suchen, um zu verstehen, welche Operationen mithilfe eines AWS KMS key (KMS-Schlüssels) ausgeführt wurden. Zu den möglichen Operationen gehören die Cluster-Verschlüsselung, die Cluster-Entschlüsselung und die Generierung von Datenschlüsseln.

AWS KMS Referenz

Das folgende Referenzmaterial enthält nützliche Informationen zur Verwendung und Verwaltung von KMS-Schlüsseln.

- Schlüsseltypreferenz. Führt den Typ des KMS-Schlüssels auf, der die jeweilige AWS KMS -API-Operation unterstützt.

  Suche: Kann ich einen RSA-Signatur-KMS-Schlüssel aktivieren und deaktivieren?
- Schlüsselstatus-Tabelle. Zeigt, wie der Schlüsselstatus eines KMS-Schlüssels dessen Verwendung in AWS KMS -API-Operationen beeinflusst.

  Suche: Kann ich den Alias eines KMS-Schlüssels ändern, dessen Löschung aussteht?
- AWS KMSReferenz zu API-Berechtigungen. Enthält Informationen zu den Berechtigungen, die für jeden AWS KMS API-Vorgang erforderlich sind.

  So finden Sie heraus: Kann GetKeyPolicyich mit einem Schlüssel in einem anderen AWS Konto arbeiten? Kann ich die Berechtigung `kms:Decrypt` in einer IAM-Richtlinie erlauben?
  - ViaService Referenz. Führt die AWS -Services auf, die den Bedingungsschlüssel `kms:ViaService` unterstützen.

    Um herauszufinden: Kann ich den `kms:ViaService` Bedingungsschlüssel verwenden, um eine Genehmigung nur dann zu gewähren, wenn sie von Amazon kommt ElastiCache? Und wie ist das bei Amazon Neptune?
- AWS KMS Preisgestaltung. Führt die Preise für KMS-Schlüssel auf und erklärt sie.

  Suche: Wie viel kostet die Verwendung meiner asymmetrischen Schlüssel?
- AWS KMS Kontingente anfordern. Listet die Kontingente pro Sekunde für AWS KMS API-Anfragen in jedem Konto und jeder Region auf.

  Suche: Wie viele Decrypt-Anforderungen kann ich pro Sekunde ausführen? Wie viele Decrypt-Anforderungen kann ich auf KMS-Schlüsseln in meinem benutzerdefinierten Schlüsselspeicher ausführen?
- AWS KMS Ressourcenkontingente. Führt die Kontingente für AWS KMS -Ressourcen auf.

  Suche: Wie viele KMS-Schlüssel kann ich in den einzelnen Regionen meines Kontos haben? Wie viele Aliasse kann ich für jeden KMS-Schlüssel haben?

- AWS integrierte Dienste AWS KMS. Führt die AWS Dienste auf, die KMS-Schlüssel verwenden, um die Ressourcen zu schützen, die sie erstellen, speichern und verwalten.

  Suche: Verwendet Amazon Connect KMS-Schlüssel, um meine Connect-Ressourcen zu schützen?

Wichtige Zustände von AWS KMS Schlüsseln

An hat AWS KMS key immer einen Schlüsselstatus. Operationen mit dem KMS-Schlüssel und seiner Umgebung können den Schlüsselstatus ändern. Der Schlüsselstatus kann sich entweder vorübergehend ändern oder bis ein anderer Vorgang seinen Schlüsselstatus ändert. Diese Operationen werden entweder asynchron oder durch einen API-Aufruf ausgeführt.

Die Tabelle in diesem Abschnitt zeigt, wie sich wichtige Status auf Aufrufe von AWS KMS API-Vorgängen auswirken. Aufgrund des Schlüsselstatus wird erwartet, dass eine Operation für einen KMS-Schlüssel erfolgreich ist (#), fehlschlägt (X) oder nur unter bestimmten Bedingungen erfolgreich ist (?). Bei KMS-Schlüssel mit importiertem Schlüsselmaterial kommt es häufig zu abweichenden Ergebnissen.

Diese Tabelle enthält nur die API-Operationen, die einen vorhandenen KMS-Schlüssel verwenden. Andere Operationen, wie CreateKeyund ListKeys, werden weggelassen.

Themen

- Schlüsselstatus und KMS-Schlüsseltypen
- Schlüsselstatus-Tabelle

Schlüsselstatus und KMS-Schlüsseltypen

Der Typ des KMS-Schlüssels bestimmt den Schlüsselstatus, den er haben kann.

- Alle KMS-Schlüssel können im Status `Enabled`, `Disabled`, und `PendingDeletion` sein.
- Die meisten KMS-Schlüssel werden im Status `Enabled` erstellt. Schlüssel mit importiertem Schlüsselmaterial werden im Status `PendingImport` erstellt.
- Der Status `PendingImport` gilt nur für KMS-Schlüssel mit importiertem Schlüsselmaterial. Wenn Schlüsselmaterial für einen importierten Schlüssel gelöscht wird oder der Schlüssel abläuft, ändert sich der Status von `Enabled` zu`PendingImport`.
- Der Status `Unavailable` gilt nur für KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher. Ein KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher liegt

vor`Unavailable`, wenn der benutzerdefinierte Schlüsselspeicher absichtlich von seinem AWS CloudHSM Cluster getrennt wird. Ein KMS-Schlüssel in einem externen Schlüsselspeicher ist `Unavailable`, wenn der benutzerdefinierte Schlüsselspeicher von seinem externen Schlüsselspeicher-Proxy absichtlich getrennt wird. Sie können diese nicht-verfügbaren KMS-Schlüssel anzeigen und verwalten, Sie können sie jedoch nicht in kryptografischen Operationen verwenden.

Der Schlüsselstatus eines KMS-Schlüssels in einem benutzerdefinierten Schlüsselspeicher wird durch Änderungen an seinem Unterstützungsschlüssel nicht beeinflusst. Ein KMS-Schlüssel in einem AWS CloudHSM Schlüsselspeicher wird durch Änderungen an seinem zugehörigen Schlüsselmaterial im AWS CloudHSM Cluster nicht beeinträchtigt. Ein KMS-Schlüssel in einem externen Schlüsselspeicher wird durch Änderungen an seinem externen Schlüssel in einem externen Schlüsselmanager nicht beeinflusst. Wenn der Unterstützungsschlüssel deaktiviert oder gelöscht wird, ändert sich der Status des KMS-Schlüssels nicht. Kryptografische Vorgänge, die den KMS-Schlüssel verwenden, schlagen jedoch fehl.

- Der `Creating`-, `Updating`-, und `PendingReplicaDeletion`-Schlüsselstatus gilt nur für multiregionale Schlüssel.
  - Ein multiregionaler Replikatschlüssel befindet sich während der Erstellung im vorübergehenden Schlüsselstatus `Creating`. Dieser Vorgang ist möglicherweise noch im Gange, wenn der ReplicateKeyVorgang abgeschlossen ist. Wenn der Replikationsprozess abgeschlossen ist, befindet sich der Replikatschlüssel im Status `Enabled` oder `PendingImport`.
  - Multiregionale Schlüssel befinden sich im vorübergehenden Schlüsselstatus `Updating`, während die primäre Region aktualisiert wird. Dieser Vorgang ist möglicherweise noch im Gange, wenn der UpdatePrimaryRegionVorgang abgeschlossen ist. Wenn der Aktualisierungsvorgang abgeschlossen ist, setzen die Primär- und Replikatschlüssel den `Enabled`-Schlüsselstatus fort.
  - Wenn Sie das Löschen eines multiregionalen Primärschlüssels planen, der Replikatschlüssel besitzt, befindet sich der Primärschlüssel im Status `PendingReplicaDeletion`, bis alle seine Replikatschlüssel gelöscht werden. Danach wechselt der Schlüsselstatus zu `PendingDeletion`. Details hierzu finden Sie unter Deleting multi-Region keys.

Schlüsselstatus-Tabelle

Die folgende Tabelle zeigt, wie sich der Schlüsselstatus eines KMS-Schlüssels auf AWS KMS - Operationen auswirkt.

Die Beschreibungen der nummerierten Fußnoten ([n] sind am Ende dieses Themas.

> **Note**
>
> Möglicherweise müssen Sie horizontal oder vertikal Scrollen, um alle Daten in dieser Tabelle anzuzeigen.

API	Enabled	Disabled	Löschen ausstehend Löschen des Replikats ausstehend	Import ausstehend	Nicht verfügbar	Erstellen	Aktualisieren
CancelKeyDeletion	[4]	[4]		[4]	[4], [13]	[4]	[4]
CreateAlias			[3]				
CreateGrant		[1]	[2] oder [3]	[5]		[14]	
Decrypt		[1]	[2] oder [3]	[5]	[11]	[14]	
DeleteAlias							

API	Enabled	Disabled	Löschen ausstehend Löschen des Replikats ausstehend	Import ausstehend	Nicht verfügbar	Erstellen	Aktualisieren
DeleteImportedKeyMaterial	[9]	[9]	[9]		N/A	[14]	[15]
DeriveSharedSecret		[1]	[2] oder [3]	[5]	N/A	[14]	
DescribeKey							
DisableKey			[3]	[5]	[12]	[14]	[15]
DisableKeyRotation	[7]	[1] oder [7]	[3] oder [7]	[6]	[7]	[14]	[7]
EnableKey			[3]	[5]	[12]	[14]	[15]
EnableKeyRotation	[7]	[1] oder [7]	[3] oder [7]	[6]	[7]	[14]	[7]

API	Enabled	Disabled	Löschen ausstehend Löschen des Replikats ausstehend	Import ausstehend	Nicht verfügbar	Erstellen	Aktualisieren
Encrypt		[1]	[2] oder [3]	[5]	[11]	[14]	
GenerateDataKey		[1]	[2] oder [3]	[5]	[11]	[14]	
GenerateDataKeyPair		[1]	[2] oder [3]	[5]	[7]	[14]	
GenerateDataKeyPairWithoutPlaintext		[1]	[2] oder [3]	[5]	[7]	[14]	
GenerateDataKeyWithoutPlaintext		[1]	[2] oder [3]	[5]	[11]	[14]	
GenerateMac		[1]	[2] oder [3]	[5]	N/A	[14]	

API	Enabled	Disabled	Löschen ausstehend Löschen des Replikats ausstehend	Import ausstehend	Nicht verfügbar	Erstellen	Aktualisieren
GetKeyPolicy							
GetKeyRotationStatus	[7]	[7]	[7]	[6]	[7]	[7]	[7]
GetParametersForImport	[9]	[9]	[8] oder [9]		[9]	[14]	[15]
GetPublicKey			[2] oder [3]		N/A	[14]	
ImportKeyMaterial	[9]	[9]	[9]		[9]	[14]	
ListAliases							
ListGrants							
ListKeyPolicies							

API	Enabled	Disabled	Löschen ausstehend Löschen des Replikats ausstehend	Import ausstehend	Nicht verfügbar	Erstellen	Aktualisieren
ListKeyRotations	[7]	[7]	[7]	[6]	[7]	[7]	[7]
ListResourceTags							
PutKeyPolicy							
ReEncrypt		[1]	[2] oder [3]	[5]	[11]	[14]	
ReplicateKey		[1]	[2] oder [3]	[5]	N/A	[14]	[15]
RetireGrant							
RevokeGrant							
RotateKeyOnDemand	[7]	[1] oder [7]	[3] oder [7]	[5]	[7]	[14]	[7]

API	Enabled	Disabled	Löschen ausstehend Löschen des Replikats ausstehend	Import ausstehend	Nicht verfügbar	Erstellen	Aktualisieren
ScheduleKeyDeletion			[3]				[15]
Sign		[1]	[2] oder [3]	[5]	N/A	[14]	
TagResource			[3]				
UntagResource			[3]				
UpdateAlias			[10]				
UpdateKeyDescription			[3]				
UpdatePrimaryRegion		[1]	[2] oder [3]	[5]	N/A	[14]	

API	Enabled	Disabled	Löschen ausstehend Löschen des Replikats ausstehend	Import ausstehend	Nicht verfügbar	Erstellen	Aktualisieren
Verify		[1]	[2] oder [3]	[5]	N/A	[14]	
VerifyMac		[1]	[2] oder [3]	[5]	N/A	[14]	

Tabellendetails

- [1] `DisabledException: <key ARN> is disabled.`
- [2] `DisabledException: <key ARN> is pending deletion (or pending replica deletion).`
- [3] `KMSInvalidStateException: <key ARN> is pending deletion (or pending replica deletion).`
- [4] `KMSInvalidStateException: <key ARN> is not pending deletion (or pending replica deletion).`
- [5] `KMSInvalidStateException: <key ARN> is pending import because no key material has ever been imported or one of the imported key materials is deleted or expired.`
- [6] `UnsupportedOperationException: <key ARN> origin is EXTERNAL which is not valid for this operation.`
- [7] Wenn sich der KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher befindet:`UnsupportedOperationException`.

- [8] Wenn der KMS-Schlüssel importiertes Schlüsselmaterial enthält: `KMSInvalidStateException`
- [9] Wenn der KMS-Schlüssel kein Schlüsselmaterial importiert haben kann:`UnsupportedOperationException`.
- [10] Wenn für den Quell-KMS-Schlüssel die Löschung aussteht, wird der Befehl erfolgreich ausgeführt. Wenn für den Ziel-KMS-Schlüssel die Löschung aussteht, schlägt der Befehl mit diesem Fehler fehl: `KMSInvalidStateException : <key ARN> is pending deletion.`
- [11] `KMSInvalidStateException: <key ARN> is unavailable.` Sie können diese Operation nicht auf einem KMS-Schlüssel durchführen, der nicht verfügbar ist.
- [12] Die Operation ist erfolgreich, aber der Schlüsselstatus des KMS-Schlüssels ändert sich erst, wenn er tatsächlich verfügbar ist.
- [13] Wenn ein KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zur Löschung aussteht, bleibt der Schlüsselstatus auf `PendingDeletion`, auch wenn der KMS-Schlüssel nicht mehr verfügbar ist. Dies erlaubt es Ihnen, die Löschung des KMS-Schlüssels vor Ablauf der Wartefrist jederzeit abzubrechen.
- [14] `KMSInvalidStateException: <key ARN> is creating.` AWS KMS löst diese Ausnahme aus, während ein Schlüssel für mehrere Regionen repliziert wird (). `ReplicateKey`
- [15] `KMSInvalidStateException: <key ARN> is updating.` AWS KMS löst diese Ausnahme aus, während die primäre Region eines Schlüssels mit mehreren Regionen aktualisiert wird (). `UpdatePrimaryRegion`

Schlüsseltypreferenz

AWS KMS unterstützt verschiedene Funktionen für verschiedene Typen von KMS-Schlüsseln. Sie können beispielsweise mit KMS-Schlüsseln zur symmetrischen Verschlüsselung nur symmetrische Datenschlüssel und asymmetrische Datenschlüsselpaare generieren. Außerdem werden das Importieren von Schlüsselmaterial und die automatische Schlüsseldrehung nur für KMS-Schlüssel zur symmetrischen Verschlüsselung unterstützt, und Sie können in einem benutzerdefinierten Schlüsselspeicher nur KMS-Schlüssel zur symmetrischen Verschlüsselung erstellen.

Diese Referenz enthält zwei Tabellen.

- In der Tabelle mit den Schlüsseltypen sind die AWS KMS Operationen aufgeführt, die für KMS-Schlüssel mit symmetrischer Verschlüsselung, asymmetrische KMS-Schlüssel und HMAC-KMS-Schlüssel gültig sind.

- In der Tabelle „Spezielle Funktionen“ sind die AWS KMS Operationen aufgeführt, die für mehrere Regionen geltende KMS-Schlüssel, KMS-Schlüssel mit importiertem Schlüsselmaterial und KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern gültig sind.

Schlüsseltyp-Tabelle

Möglicherweise müssen Sie horizontal oder vertikal scrollen, um alle Daten in dieser Tabelle anzuzeigen.

AWS KMS API-Betrieb	KMS-Schlüssel zur symmetrischen Verschlüsselung	HMAC-KMS-Schlüssel	Asymmetrische KMS-Schlüssel (ENCRYPT DECRYPT	Asymmetrische KMS-Schlüssel (SIGN_VERIFY)	Asymmetrische KMS-Schlüssel (KEY_AGREEMENT)
CancelKeyDeletion	Ja	Ja	Ja	Ja	Ja
CreateAlias	Ja	Ja	Ja	Ja	Ja
CreateGrant	Ja	Ja	Ja	Ja	Ja
CreateKey	Ja	Ja	Ja	Ja	Ja
Decrypt	Ja	Nein	Ja	Nein	Nein
DeleteAlias	Ja	Ja	Ja	Ja	Ja
DeleteImportedKeyMaterial Gilt nur für KMS-Schlüssel mit importiertem Schlüsselmaterial (`Origin` ist `EXTERNAL`).	Ja	Ja	Ja	Ja	Ja

AWS KMS API-Betrieb	KMS-Schlüssel zur symmetrischen Verschlüsselung	HMAC-KMS-Schlüssel	Asymmetrische KMS-Schlüssel (ENCRYPT DECRYPT	Asymmetrische KMS-Schlüssel (SIGN_VERIFY)	Asymmetrische KMS-Schlüssel (KEY_AGREEMENT)
DeriveSharedSecret	Nein	Nein	Nein	Nein	Ja
DescribeKey	Ja	Ja	Ja	Ja	Ja
DisableKey	Ja	Ja	Ja	Ja	Ja
DisableKeyRotation	Ja Gilt nur für KMS-Schlüssel mit AWS KMS Schlüsselmaterial (`Origin` `AWS_KMS`	Nein	Nein	Nein	Nein
EnableKey	Ja	Ja	Ja	Ja	Ja

AWS KMS API-Betrieb	KMS-Schlüssel zur symmetrischen Verschlüsselung	HMAC-KMS-Schlüssel	Asymmetrische KMS-Schlüssel (ENCRYPT DECRYPT	Asymmetrische KMS-Schlüssel (SIGN_VERIFY)	Asymmetrische KMS-Schlüssel (KEY_AGREEMENT)
EnableKeyRotation	Ja Gilt nur für KMS-Schlüssel mit AWS KMS Schlüsselmaterial (`Originis`	Nein	Nein	Nein	Nein
Encrypt	Ja	Nein	Ja	Nein	Nein
GenerateDataKey	Ja	Nein	Nein	Nein	Nein
GenerateDataKeyPair Generierung eines asymmetrischen Datenschlüsselpaars, das durch einen KMS-Schlüssel mit symmetrischer Verschlüsselung geschützt ist.	Ja Gilt nicht für KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern.	Nein	Nein	Nein	Nein

AWS KMS API-Betrieb	KMS-Schlüssel zur symmetrischen Verschlüsselung	HMAC-KMS-Schlüssel	Asymmetrische KMS-Schlüssel (ENCRYPT DECRYPT	Asymmetrische KMS-Schlüssel (SIGN_VERIFY)	Asymmetrische KMS-Schlüssel (KEY_AGREEMENT)
GenerateDataKeyPairWithoutPlaintext Generierung eines asymmetrischen Datenschlüsselpaars, das durch einen KMS-Schlüssel mit symmetrischer Verschlüsselung geschützt ist.	Ja Gilt nicht für KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern.	Nein	Nein	Nein	Nein
GenerateDataKeyWithoutPlaintext	Ja	Nein	Nein	Nein	Nein
GenerateMac	Nein	Ja	Nein	Nein	Nein
GetKeyPolicy	Ja	Ja	Ja	Ja	Ja
GetKeyRotationStatus	Ja	Ja (`KeyRotationEnabled` ist immer `false`.)	Ja (`KeyRotationEnabled` ist immer `false`.)	Ja (`KeyRotationEnabled` ist immer `false`.)	Ja (`KeyRotationEnabled` ist immer `false`.)

AWS KMS API-Betrieb	KMS-Schlüssel zur symmetrischen Verschlüsselung	HMAC-KMS-Schlüssel	Asymmetrische KMS-Schlüssel (ENCRYPT DECRYPT	Asymmetrische KMS-Schlüssel (SIGN_VERIFY)	Asymmetrische KMS-Schlüssel (KEY_AGREEMENT)
GetParametersForImport Gilt nur für KMS-Schlüssel mit importiertem Schlüsselmaterial (`Origin` ist `EXTERNAL`).	Ja	Ja	Ja	Ja	Ja
GetPublicKey	Nein	Nein	Ja	Ja	Ja
ImportKeyMaterial Gilt nur für KMS-Schlüssel mit importiertem Schlüsselmaterial (`Origin` ist `EXTERNAL`).	Ja	Ja	Ja	Ja	Ja
ListAliases	Ja	Ja	Ja	Ja	Ja
ListGrants	Ja	Ja	Ja	Ja	Ja
ListKeyPolicies	Ja	Ja	Ja	Ja	Ja

AWS KMS API-Betrieb	KMS-Schlüssel zur symmetrischen Verschlüsselung	HMAC-KMS-Schlüssel	Asymmetrische KMS-Schlüssel (ENCRYPT DECRYPT	Asymmetrische KMS-Schlüssel (SIGN_VERIFY)	Asymmetrische KMS-Schlüssel (KEY_AGREEMENT)
ListKeyRotations	Ja	Ja (Das `Rotation` Feld wird immer Null oder leer sein.)	Ja (Das `Rotation` Feld wird immer Null oder leer sein.)	Ja (Das `Rotations` Feld wird immer Null oder leer sein.)	Ja (Das `Rotations` Feld wird immer Null oder leer sein.)
ListResourceTags	Ja	Ja	Ja	Ja	Ja
ListRetirableGrants	Ja	Ja	Ja	Ja	Ja
PutKeyPolicy	Ja	Ja	Ja	Ja	Ja
ReEncrypt	Ja	Nein	Ja	Nein	Nein
ReplicateKey - gilt nur für multiregionale Schlüssel	Ja	Ja	Ja	Ja	Ja
RetireGrant	Ja	Ja	Ja	Ja	Ja
RevokeGrant	Ja	Ja	Ja	Ja	Ja

AWS KMS API-Betrieb	KMS-Schlüssel zur symmetrischen Verschlüsselung	HMAC-KMS-Schlüssel	Asymmetrische KMS-Schlüssel (ENCRYP DECRYPT	Asymmetrische KMS-Schlüssel (SIGN_VERIFY)	Asymmetrische KMS-Schlüssel (KEY_AGREEMENT)
RotateKeyOnDemand	Ja Gilt nur für vom Kunden verwaltete KMS-Schlüssel mit symmetrischer Verschlüsselung `AWS_KMS` ohne `EXTERNAL` Herkunft.	Nein	Nein	Nein	Nein
ScheduleKeyDeletion	Ja	Ja	Ja	Ja	Ja
Sign	Nein	Nein	Nein	Ja	Nein
TagResource	Ja	Ja	Ja	Ja	Ja
UntagResource	Ja	Ja	Ja	Ja	Ja

AWS KMS API-Betrieb	KMS-Schlüssel zur symmetrischen Verschlüsselung	HMAC-KMS-Schlüssel	Asymmetrische KMS-Schlüssel (ENCRYPT DECRYPT	Asymmetrische KMS-Schlüssel (SIGN_VERIFY)	Asymmetrische KMS-Schlüssel (KEY_AGREEMENT)
UpdateAlias Der aktuelle KMS-Schlüssel und der neue KMS-Schlüssel müssen vom selben Typ sein (beide symmetrisch oder beide asymmetrisch oder beide HMAC) und die gleiche Schlüsselnutzung aufweisen.	Ja	Ja	Ja	Ja	Ja
UpdateKeyDescription	Ja	Ja	Ja	Ja	Ja
UpdateReplicaRegion - gilt nur für multiregionale Schlüssel	Ja	Ja	Ja	Ja	Ja
Verify	Nein	Nein	Nein	Ja	Nein
VerifyMac	Nein	Ja	Nein	Nein	Nein

Tabelle „Spezielle Funktionen“

In dieser Tabelle sind die AWS KMS API-Operationen aufgeführt, die für jeden Typ von Spezialschlüsseln unterstützt werden.

Achten Sie beim Lesen dieser Tabelle auf die folgenden Interaktionen:

- Multiregionale Schlüssel:
  - Bei multiregionalen Schlüsseln kann es sich um KMS-Schlüssel mit symmetrischer Verschlüsselung, KMS-Schlüssel mit asymmetrischer Verschlüsselung, HMAC-KMS-Schlüssel und KMS-Schlüssel mit importiertem Schlüsselmaterial handeln.
  - Sie können keine multiregionale Schlüssel in einem benutzerdefinierten Schlüsselspeicher verwenden.
- Importiertes Schlüsselmaterial
  - Sie können Schlüsselmaterial für symmetrisch verschlüsselte KMS-Schlüssel, asymmetrische KMS-Schlüssel und HMAC-KMS-Schlüssel importieren.
  - Sie können multiregionale Schlüssel mit importiertem Schlüsselmaterial erstellen.
  - Sie können keine Schlüssel mit importiertem Schlüsselmaterial oder in einem benutzerdefinierten Schlüsselspeicher erstellen.
  - Automatische Schlüsseldrehung (`EnableKeyRotation`, `DisableKeyRotation`) wird für KMS-Schlüssel oder KMS-Schlüssel mit importiertem Schlüsselmaterial nicht unterstützt.
  - Die On-Demand-Schlüsselrotation (`RotateKeyOnDemand`) wird für KMS-Schlüssel mit symmetrischer Verschlüsselung für einzelne Regionen mit importiertem Schlüsselmaterial unterstützt.
- Benutzerdefinierte Schlüsselspeicher
  - Benutzerdefinierte Schlüsselspeicher unterstützen nur KMS-Schlüssel mit symmetrischer Verschlüsselung.
  - Symmetrische Operationen mit asymmetrischen Schlüsselpaaren (`GenerateDataKeyPair`, `GenerateDataKeyPairWithoutPlaintext`) werden für KMS-Schlüssel in benutzerdefinierten Schlüsselspeichern nicht unterstützt.
  - Automatische Schlüsselrotation (`EnableKeyRotation`, `DisableKeyRotation`) wird bei KMS-Schlüsseln in benutzerdefinierten Schlüsselspeichern nicht unterstützt.
  - Sie können keine multiregionalen Schlüssel in benutzerdefinierten Schlüsselspeichern erstellen.

Möglicherweise müssen Sie horizontal oder vertikal scrollen, um alle Daten in dieser Tabelle anzuzeigen.

AWS KMS API-Betrieb	Multiregionale Schlüssel	Importiertes Schlüssel material	KMS-Schlüssel in einem benutzerdefinierten Schlüssel speicher
CancelKeyDeletion			
CreateAlias			
CreateGrant			
CreateKey Sie können `CreateKey` verwenden, um einen multiregionalen Primärschlüssel, einen KMS-Schlüssel mit importiertem Schlüsselmaterial oder einen KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher zu erstellen. Verwenden Sie `ReplicateKey` zum Erstellen eines multiregionalen Replikatschlüssels.			
Decrypt	Gilt nur, wenn `KeyUsage` `ENCRYPT_DECRYPT` ist.		
DeleteAlias			
DeleteImportedKeyMaterial			

AWS KMS API-Betrieb	Multiregionale Schlüssel	Importiertes Schlüssel material	KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher
	Gilt nur für Schlüssel mit importiertem Schlüsselmaterial (`Origin` ist `EXTERNAL`)		
DeriveSharedSecret	Gültig (nur wenn `KeyUsage` ist `KEY_AGREEMENT`)	Gültig (nur wenn `KeyUsage` ist `KEY_AGREEMENT`)	
DescribeKey			
DisableKey			
DisableKeyRotation	Gilt nur für symmetrische Verschlüsselungsschlüssel mit AWS KMS Schlüsselmaterial (`Origin` ist `AWS_KM`		

AWS KMS API-Betrieb	Multiregionale Schlüssel	Importiertes Schlüsselmaterial	KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher
EnableKey	Gilt nur für KMS-Schlüssel mit symmetrischer Verschlüsselung		
EnableKeyRotation	Gilt nur für symmetrische Verschlüsselungsschlüssel mit AWS KMS Schlüsselmaterial (`Origin`ist`AWS_KM`		
Encrypt	Gilt nur, wenn `KeyUsage` `ENCRYPT_DECRYPT` ist.		
GenerateDataKey	Gilt nur für KMS-Schlüssel mit symmetrischer Verschlüsselung		

AWS KMS API-Betrieb	Multiregionale Schlüssel	Importiertes Schlüssel material	KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher
GenerateDataKeyPair	Gilt nur für KMS-Schlüssel mit symmetrischer Verschlüsselung		
GenerateDataKeyPairWithoutPlaintext	Gilt nur für KMS-Schlüssel mit symmetrischer Verschlüsselung		
GenerateDataKeyWithoutPlaintext	Gilt nur für KMS-Schlüssel mit symmetrischer Verschlüsselung		
GenerateMac Gilt nur für HMAC-KMS-Schlüssel			
GetKeyPolicy			

AWS KMS API-Betrieb	Multiregionale Schlüssel	Importiertes Schlüsselmaterial	KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher
GetKeyRotationStatus		(`KeyRotationEnabled` ist immer `false`.)	
GetParametersForImport	Gilt nur für Schlüssel mit importiertem Schlüsselmaterial (`Origin` ist `EXTERNAL`).		
GetPublicKey Gilt nur für asymmetrische KMS-Schlüssel.			
ImportKeyMaterial	Gilt nur für Schlüssel mit importiertem Schlüsselmaterial (`Origin` ist `EXTERNAL`).		

AWS KMS API-Betrieb	Multiregionale Schlüssel	Importiertes Schlüssel material	KMS-Schlüssel in einem benutzerdefinierten Schlüssel speicher
ListAliases			
ListGrants			
ListKeyPolicies			
ListKeyRotations	Gilt nur für symmetrische Verschlüsselungsschlüssel mit `AWS_KMS` Ursprung.	Gilt nur für symmetrische Verschlüsselungsschlüssel mit einer Region.	
ListResourceTags			
ListRetirableGrants			
PutKeyPolicy			
ReEncrypt	Gilt nur, wenn `KeyUsage` `ENCRYPT_DECRYPT` ist.		

AWS KMS API-Betrieb	Multiregionale Schlüssel	Importiertes Schlüssel material	KMS-Schlüssel in einem benutzerdefinierten Schlüsselspeicher
ReplicateKey	Gilt nur für multiregionale Primärschlüssel.	Gilt nur für multiregionale Primärschlüssel.	
RetireGrant			
RevokeGrant			
RotateKeyOnDemand	Gilt nur für symmetrische Verschlüsselungsschlüssel mit Herkunft. `AWS_KMS`	Gilt nur für symmetrische Verschlüsselungsschlüssel mit einer Region.	
ScheduleKeyDeletion			
Sign Gilt nur, wenn `KeyUsage` `SIGN_VERIFY` ist.			
TagResource			
UntagResource			

AWS KMS API-Betrieb	Multiregionale Schlüssel	Importiertes Schlüssel material	KMS-Schlüssel in einem benutzerdefinierten Schlüssel speicher
UpdateAlias - Der aktuelle KMS-Schlüssel und der neue KMS-Schlüssel müssen vom selben Typ sein (beide symmetrisch oder beide asymmetrisch oder beide HMAC) und die gleiche Schlüssel nutzung aufweisen.			
UpdateKeyDescription			
UpdateReplicaRegion		Gilt nur für multiregionale Schlüssel.	
Verify Gilt nur, wenn `KeyUsage` `SIGN_VERIFY` ist.			
VerifyMac Gilt nur für HMAC-KMS-Schlüssel			

Referenz zu wichtigen Spezifikationen

Wenn Sie einen asymmetrischen KMS-Schlüssel oder HMAC-KMS-Schlüssel erstellen, wählen Sie dessen Schlüsselspezifikation aus. Die Schlüsselspezifikation, die eine Eigenschaft von Every ist AWS KMS key, stellt die kryptografische Konfiguration Ihres KMS-Schlüssels dar. Sie wählen die

Schlüsselspezifikation, wenn Sie den KMS-Schlüssel erstellen. Sie kann danach nicht mehr geändert werden. Wenn Sie die falsche Schlüsselspezifikation ausgewählt haben, löschen Sie den KMS-Schlüssel und erstellen Sie einen neuen.

> **Note**
>
> Die Schlüsselspezifikation für einen KMS-Schlüssel wurde als „Kunden-Hauptschlüssel-Spezifikation“ bezeichnet. Der `CustomerMasterKeySpec` Parameter des CreateKeyVorgangs ist veraltet. Verwenden Sie stattdessen den `KeySpec`-Parameter. Die Antwort der DescribeKeyOperationen `CreateKey` und enthält ein `KeySpec` `CustomerMasterKeySpec` Und-Element mit demselben Wert.

Die Schlüsselspezifikation bestimmt, ob der KMS-Schlüssel symmetrisch oder asymmetrisch ist, welche Art von Schlüsselmaterial im KMS-Schlüssel enthalten ist und welche Verschlüsselungs-, Signaturalgorithmen oder MAC-Algorithmen (Message Authentication Code) den KMS-Schlüssel AWS KMS unterstützen. Die von Ihnen gewählte Schlüsselspezifikation wird in der Regel durch Ihren Anwendungsfall und gesetzliche Anforderungen bestimmt. Allerdings haben kryptografische Operationen zu KMS-Schlüsseln mit unterschiedlichen Schlüsselspezifikationen unterschiedliche Preise und unterliegen unterschiedlichen Kontingenten. Details zu den Preisen finden Sie unter AWS Key Management Service -Preise. Weitere Informationen zu Anforderungskontingenten finden Sie unter Anforderungskontingente.

Um die Schlüsselspezifikationen einzuschränken, die Prinzipale bei der Erstellung von KMS-Schlüsseln verwenden können, verwenden Sie den Bedingungsschlüssel kms:. KeySpec Sie können den `kms:KeySpec` Bedingungsschlüssel auch verwenden, um es Prinzipalen zu ermöglichen, AWS KMS Operationen nur für KMS-Schlüssel mit einer bestimmten Schlüsselspezifikation aufzurufen. Beispielsweise können Sie die Berechtigung zum Planen des Löschens eines KMS-Schlüssels mit einer `RSA_4096`-Schlüsselspezifikation verweigern.

AWS KMS unterstützt die folgenden Schlüsselspezifikationen für KMS-Schlüssel:

Symmetrische Verschlüsselungsschlüsselspezifikation(Standard)

- SYMMETRIC_DEFAULT

RSA-Schlüsselspezifikationen (Verschlüsselung und Entschlüsselung -oder- Signatur und Verifizierung)

- RSA_2048

- RSA_3072
- RSA_4096

Elliptic Curve(EC)-Schlüsselspezifikationen

- Von NIST empfohlene asymmetrische Schlüsselpaare mit elliptischen Kurven (Signierung und Verifizierung oder Ableitung gemeinsam genutzter Geheimnisse)
  - ECC_NIST_P256 (secp256r1)
  - ECC_NIST_P384 (secp384r1)
  - ECC_NIST_P521 (secp521r1)
- Andere asymmetrische Elliptic Curve-Schlüsselpaare (Signatur und Verifizierung)
  - ECC_SECG_P256K1 (secp256k1), häufig für Kryptowährung verwendet.

SM2 Schlüsselspezifikation (Verschlüsselung und Entschlüsselung — oder — Signierung und Verifizierung — oder Ableitung gemeinsamer Geheimnisse)

- SM2 (nur Regionen China)

HMAC-Schlüsselspezifikationen

- HMAC_224
- HMAC_256
- HMAC_384
- HMAC_512

ML-DSA — wichtige Spezifikationen

- ML_DSA_44
- ML_DSA_65
- ML_DSA_87

Schlüsselspezifikation SYMMMETRIC_DEFAULT

Die Standard-Schlüsselspezifikation SYMMMETRIC_DEFAULT ist die Schlüsselspezifikation für KMS-Schlüssel mit symmetrischer Verschlüsselung. Wenn Sie in der Konsole den Schlüsseltyp Symmetrisch und die Schlüsselverwendung verschlüsseln und entschlüsseln auswählen, wird die AWS KMS Schlüsselspezifikation ausgewählt. `SYMMETRIC_DEFAULT` Wenn Sie bei der CreateKeyOperation keinen `KeySpec` Wert angeben, wird SYMMETRIC_DEFAULT ausgewählt. Wenn Sie keinen Grund haben, eine andere Schlüsselspezifikation zu verwenden, ist SYMMETRIC_DEFAULT eine gute Wahl.

SYMMETRIC_DEFAULT steht für AES-256-GCM, einen symmetrischen Algorithmus, der auf dem Advanced Encryption Standard (AES) im Galois Counter Mode (GCM) mit 256-Bit-Schlüsseln basiert, einem Industriestandard für sichere Verschlüsselung. Der Chiffretext, den dieser Algorithmus generiert, unterstützt zusätzliche authentifizierte Daten (AAD), z. B. einen Verschlüsselungskontext, und GCM bietet eine zusätzliche Integritätsprüfung für den Chiffretext.

Die unter AES-256-GCM verschlüsselten Daten sind gegenwärtig und zukünftig geschützt. Kryptographen betrachten diesen Algorithmus als quantenresistent. Theoretische zukünftige, groß angelegte Quantenrechnungsangriffe auf Verschlüsselungstexte, die unter 256-Bit-AES-GCM-Schlüsseln erstellt wurden, reduzieren die effektive Sicherheit des Schlüssels auf 128-Bits. Diese Sicherheitsstufe reicht jedoch aus, um Brute-Force-Angriffe auf Chiffretexte unmöglich zu machen. AWS KMS

Die einzige Ausnahme in China Regionen, wo SYMMETRIC_DEFAULT für einen symmetrischen 128-Bit-Schlüssel steht, der Verschlüsselung verwendet. SM4 Sie können einen SM4 128-Bit-Schlüssel nur in Regionen China erstellen. Sie können keinen 256-Bit-AES-GCM-KMS-Schlüssel in China erstellen.

Sie können einen KMS-Schlüssel mit symmetrischer Verschlüsselung verwenden, um Daten AWS KMS zu verschlüsseln, zu entschlüsseln und erneut zu verschlüsseln und um generierte Datenschlüssel und Datenschlüsselpaare zu schützen. AWS Dienste, die in integriert sind, AWS KMS verwenden KMS-Schlüssel mit symmetrischer Verschlüsselung, um Ihre Daten im Ruhezustand zu verschlüsseln. Sie können in einen KMS-Schlüssel mit symmetrischer Verschlüsselung Ihr eigenes Schlüsselmaterial importieren und KMS-Schlüssel mit symmetrischer Verschlüsselung in benutzerdefinierten Schlüsselspeichern erstellen. Eine Tabelle mit den Operationen, die Sie für symmetrische und asymmetrische KMS-Schlüssel ausführen können, finden Sie unter Vergleich symmetrischer und asymmetrischer KMS-Schlüssel.

Sie können einen KMS-Schlüssel mit symmetrischer Verschlüsselung verwenden, um Daten AWS KMS zu verschlüsseln, zu entschlüsseln und erneut zu verschlüsseln sowie Datenschlüssel und Datenschlüsselpaare zu generieren. Sie können multiregionale KMS-Schlüssel mit symmetrischer Verschlüsselung erstellen, ihr eigenes Schlüsselmaterial in einen KMS-Schlüssel mit symmetrischer Verschlüsselung importieren und KMS-Schlüssel mit symmetrischer Verschlüsselung in benutzerdefinierten Schlüsselspeichern erstellen. Eine Tabelle mit den Operationen, die Sie auf unterschiedlichen KMS-Schlüsseltypen durchführen können, finden Sie unter Schlüsseltypreferenz.

RSA-Schlüsselspezifikationen

Wenn Sie eine RSA-Schlüsselspezifikation verwenden, AWS KMS wird ein asymmetrischer KMS-Schlüssel mit einem RSA-Schlüsselpaar erstellt. Der private Schlüssel wird niemals unverschlüsselt verlassen. AWS KMS Sie können den öffentlichen Schlüssel innerhalb von verwenden oder den öffentlichen Schlüssel herunterladen AWS KMS, um ihn außerhalb von AWS KMS zu verwenden.

> **Warning**
>
> Wenn Sie Daten außerhalb von verschlüsseln, stellen Sie sicher AWS KMS, dass Sie Ihren Chiffretext entschlüsseln können. Wenn Sie den öffentlichen Schlüssel aus einem KMS-Schlüssel, der aus AWS KMS gelöscht wurde, den öffentlichen Schlüssel aus einem für Signatur und Überprüfung konfigurierten KMS-Schlüssel oder einen vom KMS-Schlüssel nicht unterstützten Verschlüsselungsalgorithmus verwenden, können die Daten nicht wiederhergestellt werden.

In AWS KMS können Sie asymmetrische KMS-Schlüssel mit RSA-Schlüsselpaaren für die Verschlüsselung und Entschlüsselung oder für das Signieren und Überprüfen verwenden, aber nicht beides. Diese Eigenschaft, auch bekannt als Key usage, wird unabhängig von der Schlüsselspezifikation bestimmt. Sie sollten diese Entscheidung jedoch treffen, bevor Sie eine Schlüsselspezifikation auswählen.

AWS KMS unterstützt die folgenden RSA-Schlüsselspezifikationen für Verschlüsselung und Entschlüsselung oder Signierung und Überprüfung:

- RSA_2048
- RSA_3072
- RSA_4096

Die RSA-Schlüsselspezifikationen unterscheiden sich in der Länge des RSA-Schlüssels in Bits. Die von Ihnen gewählte RSA-Schlüsselspezifikation hängt möglicherweise von Ihren Sicherheitsstandards oder den Anforderungen Ihrer Aufgabe ab. Verwenden Sie im Allgemeinen den größten Schlüssel, der für Ihre Aufgabe praktisch und erschwinglich ist. Kryptografische Operationen zu KMS-Schlüsseln mit unterschiedlichen RSA-Schlüsselspezifikationen haben unterschiedliche Preise. AWS KMS Preisinformationen finden Sie unter Preise für den AWS Key Management Service. Weitere Hinweise zu Anforderungskontingenten finden Sie unter Anforderungskontingente.

RSA-Schlüsselspezifikationen für Verschlüsselung und Entschlüsselung

Wenn ein asymmetrischer RSA-KMS-Schlüssel für die Verschlüsselung und Entschlüsselung verwendet wird, verschlüsseln Sie mit dem öffentlichen Schlüssel und entschlüsseln mit dem privaten Schlüssel. Wenn Sie den `Encrypt` Vorgang AWS KMS für einen RSA-KMS-Schlüssel aufrufen, AWS KMS verwendet den öffentlichen Schlüssel im RSA-Schlüsselpaar und den von Ihnen angegebenen Verschlüsselungsalgorithmus, um Ihre Daten zu verschlüsseln. Um den Chiffretext zu entschlüsseln, rufen Sie den `Decrypt` Vorgang auf und geben Sie denselben KMS-Schlüssel und denselben Verschlüsselungsalgorithmus an. AWS KMS verwendet dann den privaten Schlüssel im RSA-Schlüsselpaar, um Ihre Daten zu entschlüsseln.

Sie können den öffentlichen Schlüssel auch herunterladen und damit Daten außerhalb von verschlüsseln. AWS KMS Stellen Sie sicher, dass Sie einen Verschlüsselungsalgorithmus verwenden, der RSA-KMS-Schlüssel AWS KMS unterstützt. Um den Chiffretext zu entschlüsseln, rufen Sie die `Decrypt`-Funktion mit demselben KMS-Schlüssel und Verschlüsselungsalgorithmus auf.

AWS KMS unterstützt zwei Verschlüsselungsalgorithmen für KMS-Schlüssel mit RSA-Schlüsselspezifikationen. Diese Algorithmen, die in PKCS #1 v2.2, definiert sind, unterscheiden sich in der Hashfunktion, die sie intern verwenden. In AWS KMS verwenden die RSAES_OAEP-Algorithmen immer dieselbe Hash-Funktion sowohl für Hashing-Zwecke als auch für die Maskengenerierungsfunktion (). MGF1 Sie müssen beim Aufruf der Operationen Encrypt und Decrypt einen Verschlüsselungsalgorithmus angeben. Sie können für jede Anforderung einen anderen Algorithmus auswählen.

Unterstützte Verschlüsselungsalgorithmen für RSA-Schlüsselspezifikationen

Verschlüsselungsalgorithmus	Beschreibung des Algorithmus
RSAES_OAEP_SHA_1	PKCS #1 v2.2, Abschnitt 7.1. RSA-Versc hlüsselung mit OAEP-Padding, wobei SHA-1 sowohl für den Hash als auch für die Maskengenerierungsfunktion zusammen mit einem leeren Label verwendet wird. MGF1
RSAES_OAEP_SHA_256	PKCS #1, Abschnitt 7.1. RSA-Verschlüsselun g mit OAEP-Padding unter Verwendung von SHA-256 sowohl für den Hash als auch für die

Verschlüsselungsalgorithmus	Beschreibung des Algorithmus
	Maskengenerierungsfunktion zusammen mit einem leeren Label. MGF1

Sie können einen KMS-Schlüssel nicht so konfigurieren, dass er einen bestimmten Verschlüsselungsalgorithmus verwendet. Sie können jedoch die EncryptionAlgorithm Richtlinienbedingung kms: verwenden, um die Verschlüsselungsalgorithmen anzugeben, die Prinzipale mit dem KMS-Schlüssel verwenden dürfen.

Um die Verschlüsselungsalgorithmen für einen KMS-Schlüssel abzurufen, sehen Sie sich die kryptografische Konfiguration des KMS-Schlüssels in der AWS KMS Konsole an oder verwenden Sie den DescribeKeyVorgang. AWS KMS stellt außerdem die Schlüsselspezifikation und die Verschlüsselungsalgorithmen bereit, wenn Sie Ihren öffentlichen Schlüssel entweder in der AWS KMS Konsole oder mithilfe des GetPublicKeyVorgangs herunterladen.

Sie können eine RSA-Schlüsselspezifikation basierend auf der Länge der Klartextdaten auswählen, die Sie in jeder Anforderung verschlüsseln können. Die folgende Tabelle zeigt die maximale Größe (in Byte) des Klartextes, den Sie bei einem einzelnen Aufruf der Produktion Encrypt verschlüsseln können. Die Werte unterscheiden sich je nach Schlüsselspezifikation und Verschlüsselungsalgorithmus. Zum Vergleich können Sie einen KMS-Schlüssel mit symmetrischer Verschlüsselung verwenden, um bis zu 4096 Bytes gleichzeitig zu verschlüsseln.

Verwenden Sie die folgende Formel, um die maximale Klartextlänge in Byte für diese Algorithmen zu berechnen: (*key_size_in_bits*/8) - (2 * *hash_length_in_bits* /8) - 2. Für RSA_2048 mit SHA-256 beträgt beispielsweise die maximale Klartextgröße in Bytes (2048/8) - (2 * 256/8) -2 = 190.

Maximale Klartextgröße (in Bytes) in einer Verschlüsselungsoperation

	Verschlüsselungsalgorithmus	
Schlüsselspezifikation	RSAES_OAEP_SHA_1	RSAES_OAEP_SHA_256
RSA_2048	214	190
RSA_3072	342	318
RSA_4096	470	446

RSA-Schlüsselspezifikationen für Signatur und Verifizierung

Wenn ein asymmetrischer RSA-KMS-Schlüssel für Signatur und Verifizierung verwendet wird, generieren Sie die Signatur für eine Nachricht mit dem privaten Schlüssel und überprüfen die Signatur mit dem öffentlichen Schlüssel.

Wenn Sie den `Sign` Vorgang AWS KMS für einen asymmetrischen KMS-Schlüssel aufrufen, AWS KMS verwendet den privaten Schlüssel im RSA-Schlüsselpaar, die Nachricht und den von Ihnen angegebenen Signaturalgorithmus, um eine Signatur zu generieren. Um die Signatur zu überprüfen, rufen Sie die Produktion Verify auf. Geben Sie die Signatur sowie denselben KMS-Schlüssel, dieselbe Nachricht und denselben Signaturalgorithmus an. AWS KMS verwendet dann den öffentlichen Schlüssel im RSA-Schlüsselpaar, um die Signatur zu überprüfen. Sie können den öffentlichen Schlüssel auch herunterladen und ihn verwenden, um die Signatur außerhalb von AWS KMS zu überprüfen.

AWS KMS unterstützt die folgenden Signaturalgorithmen für alle KMS-Schlüssel mit einer RSA-Schlüsselspezifikation. Sie müssen einen Signaturalgorithmus angeben, wenn Sie die Operationen Sign (Signieren) und Verify (Überprüfen) aufrufen. Sie können für jede Anforderung einen anderen Algorithmus auswählen. Beim Signieren mit RSA-Schlüsselpaaren werden RSASSA-PSS-Algorithmen bevorzugt. Aus Gründen der Kompatibilität mit bestehenden Anwendungen verwenden wir RSASSA- PKCS1 -v1_5-Algorithmen.

Unterstützte Signaturalgorithmen für RSA-Schlüsselspezifikationen

Signaturalgorithmus	Beschreibung des Algorithmus
RSASSA_PSS_SHA_256	PKCS #1 v2.2, Abschnitt 8.1, RSA-Signatur mit PSS-Padding unter Verwendung von SHA-256 sowohl für den Nachrichten-Digest als auch für die Maskengenerierungsfunktion zusammen mit einem 256-Bit-Salt MGF1
RSASSA_PSS_SHA_384	PKCS #1 v2.2, Abschnitt 8.1, RSA-Signatur mit PSS-Padding unter Verwendung von SHA-384 sowohl für den Nachrichten-Digest als auch für die Maskengenerierungsfunktion zusammen mit einem 384-Bit-Salt MGF1

Signaturalgorithmus	Beschreibung des Algorithmus
RSASSA_PSS_SHA_512	PKCS #1 v2.2, Abschnitt 8.1, RSA-Signatur mit PSS-Padding unter Verwendung von SHA-512 sowohl für den Nachrichten-Digest als auch für die Maskengenerierungsfunktion zusammen mit einem 512-Bit-Salt MGF1
RSASSA_ _V1_5_SHA_256 PKCS1	PKCS #1 v2.2, Abschnitt 8.2, RSA-Signatur mit PKCS #1v1.5 Auffüllung und SHA-256
RSASSA_ PKCS1 _V1_5_SHA_384	PKCS #1 v2.2, Abschnitt 8.2, RSA-Signatur mit PKCS #1v1.5 Auffüllung und SHA-384
RSASSA_ PKCS1 _V1_5_SHA_512	PKCS #1 v2.2, Abschnitt 8.2, RSA-Signatur mit PKCS #1v1.5 Auffüllung und SHA-512

Sie können einen KMS-Schlüssel nicht so konfigurieren, dass bestimmte Signaturalgorithmen verwendet werden. Sie können jedoch die SigningAlgorithm Richtlinienbedingung kms: verwenden, um die Signaturalgorithmen anzugeben, die Prinzipale mit dem KMS-Schlüssel verwenden dürfen.

Um die Signaturalgorithmen für einen KMS-Schlüssel abzurufen, sehen Sie sich die kryptografische Konfiguration des KMS-Schlüssels in der AWS KMS Konsole an oder verwenden Sie den DescribeKeyVorgang. AWS KMS stellt außerdem die Schlüsselspezifikation und die Signaturalgorithmen bereit, wenn Sie Ihren öffentlichen Schlüssel entweder in der AWS KMS Konsole oder mithilfe des GetPublicKeyVorgangs herunterladen.

Elliptic Curve(EC)-Schlüsselspezifikationen

Wenn Sie eine ECC-Schlüsselspezifikation (Elliptic Curve) verwenden, AWS KMS wird ein asymmetrischer KMS-Schlüssel mit einem ECC-Schlüsselpaar zum Signieren und Verifizieren oder zum Ableiten von gemeinsamen Geheimnissen (aber nicht beides) erstellt. Der private Schlüssel, der Signaturen generiert oder gemeinsame Geheimnisse ableitet, bleibt niemals unverschlüsselt. AWS KMS Sie können den öffentlichen Schlüssel verwenden, um Signaturen innerhalb von zu überprüfen AWS KMS, oder den öffentlichen Schlüssel herunterladen, um ihn außerhalb von zu verwenden. AWS KMS

AWS KMS unterstützt die folgenden ECC-Schlüsselspezifikationen für asymmetrische KMS-Schlüssel.

- Von NIST empfohlene asymmetrische Schlüsselpaare mit elliptischen Kurven (Signierung und Verifizierung — oder Ableitung gemeinsam genutzter Geheimnisse)
  - ECC_NIST_P256 (secp256r1)
  - ECC_NIST_P384 (secp384r1)
  - ECC_NIST_P521 (secp521r1)
- Andere asymmetrische Elliptic Curve-Schlüsselpaare (Signatur und Verifizierung)
  - ECC_SECG_P256K1 (secp256k1), häufig für Kryptowährung verwendet.

Die von Ihnen gewählte ECC-Schlüsselspezifikation hängt möglicherweise von Ihren Sicherheitsstandards oder den Anforderungen Ihrer Aufgabe ab. Verwenden Sie im Allgemeinen die Kurve mit den meisten Punkten, die für Ihre Aufgabe praktisch und erschwinglich ist.

Wenn Sie einen asymmetrischen KMS-Schlüssel erstellen, um gemeinsame geheime Schlüssel abzuleiten, verwenden Sie eine der von NIST empfohlenen Spezifikationen für elliptische Kurvenschlüssel. Der einzige unterstützte Schlüsselvereinbarungsalgorithmus zur Ableitung gemeinsamer Geheimnisse ist der Elliptic Curve Cryptography Cofactor Diffie-Hellman Primitive (ECDH). Ein Beispiel dafür, wie gemeinsame Geheimnisse offline abgeleitet werden können, finden Sie unter. the section called “Offline-Ableitung gemeinsam genutzter Geheimnisse”

Wenn Sie einen asymmetrischen KMS-Schlüssel für die Verwendung mit Kryptowährungen erstellen, verwenden Sie die Schlüsselspezifikation ECC_SECG_P256K1. Sie können diese Schlüsselspezifikation auch für andere Zwecke verwenden, aber sie ist für Bitcoin und andere Kryptowährungen erforderlich.

KMS-Schlüssel mit unterschiedlichen ECC-Schlüsselspezifikationen haben unterschiedliche Preise und unterliegen unterschiedlichen Anforderungkontingenten. Informationen zur Preisgestaltung finden Sie unter AWS KMS AWS Key Management Service Preisgestaltung. Weitere Informationen zu Anforderungskontingenten finden Sie unter Anforderungskontingente.

In der folgenden Tabelle sind die Signaturalgorithmen aufgeführt, die für die einzelnen ECC-Schlüsselspezifikationen AWS KMS unterstützt werden. Sie können einen KMS-Schlüssel nicht so konfigurieren, dass bestimmte Signaturalgorithmen verwendet werden. Sie können jedoch die SigningAlgorithm Richtlinienbedingung kms: verwenden, um die Signaturalgorithmen anzugeben, die Prinzipale mit dem KMS-Schlüssel verwenden dürfen.

Unterstützte Signaturalgorithmen für ECC-Schlüsselspezifikationen

Schlüsselspezifikation	Signaturalgorithmus	Beschreibung des Algorithmus
ECC_NIST_P256	ECDSA_SHA_256	NIST FIPS 186-4, Section 6.4, ECDSA signature using the curve specified by the key and SHA-256 für den Message Digest.
ECC_NIST_P384	ECDSA_SHA_384	NIST FIPS 186-4, Section 6.4, ECDSA signature using the curve specified by the key and SHA-384 für den Nachrichten-Digest.
ECC_NIST_P521	ECDSA_SHA_512	NIST FIPS 186-4, Section 6.4, ECDSA signature using the curve specified by the key and SHA-512 für den Nachrichten-Digest.
ECC_SECG_P256K1	ECDSA_SHA_256	NIST FIPS 186-4, Section 6.4, ECDSA signature using the curve specified by the key and SHA-256 für den Nachrichten-Digest.

Schlüsselspezifikationen für HMAC-KMS-Schlüssel

AWS KMS unterstützt symmetrische HMAC-Schlüssel in unterschiedlichen Längen. Häufig wird die von Ihnen ausgewählte Schlüsselspezifikation durch Ihre Sicherheits-, gesetzliche, oder geschäftliche Anforderungen bestimmt. Die Länge des Schlüssels bestimmt den MAC-Algorithmus, der in GenerateMacAND-Operationen verwendet wird. VerifyMac Im Allgemeinen sind längere Schlüssel sicherer. Verwenden Sie den längsten Schlüssel, der für Ihren Anwendungsfall praktisch ist.

HMAC-Schlüsselspezifikation	MAC-Algorithmen
HMAC_224	HMAC_SHA_224
HMAC_256	HMAC_SHA_256
HMAC_384	HMAC_SHA_384
HMAC_512	HMAC_SHA_512

ML-DSA — wichtige Spezifikationen

Ein ML-DSA-Schlüssel ist ein kryptografischer Schlüssel, der im Module-Lattice-Based Digital Signature Algorithm (ML-DSA) verwendet wird, der für die Post-Quanten-Kryptografie konzipiert ist. Dieser Algorithmus ist Teil der Standardisierungsbemühungen des NIST (National Institute of Standards and Technology), die speziell in den Federal Information Processing Standards (FIPS) 204 beschrieben sind.

ML-DSA-Schlüssel werden in einem öffentlich-privaten Schlüsselpaarsystem verwendet. Der private Schlüssel wird zum Signieren von Daten verwendet, während der öffentliche Schlüssel zur Überprüfung der Signatur verwendet wird. Dieses System gewährleistet die Authentizität, Integrität und Nichtabstreitbarkeit digitaler Nachrichten oder Dokumente, selbst angesichts potenzieller Bedrohungen durch Quantencomputer.

Wenn Sie einen Schlüssel mit der ML-DSA-Schlüsselspezifikation erstellen, AWS KMS wird ein asymmetrischer KMS-Schlüssel mit einem ML-DSA-Schlüsselpaar zum Signieren und Überprüfen erstellt. Der private Schlüssel, der Signaturen generiert, bleibt niemals unverschlüsselt. AWS KMS Sie können den öffentlichen Schlüssel verwenden, um Signaturen innerhalb von zu verifizieren AWS KMS, oder den öffentlichen Schlüssel herunterladen, um ihn außerhalb von AWS KMS zu verwenden.

AWS KMS unterstützt die folgenden ML-DSA-Schlüsselspezifikationen für asymmetrische KMS-Schlüssel:

- ML_DSA_44
- ML_DSA_65
- ML_DSA_87

AWS KMS unterstützt den ML_DSA_SHAKE_256-Signaturalgorithmus für alle ML-DSA-Schlüsselspezifikationen.

SM2 wichtige Spezifikation (nur Regionen China)

Die SM2 wichtigste Spezifikation ist eine Schlüsselspezifikation mit elliptischer Kurve, die in der vom chinesischen Office of State Commercial Cryptography Administration (OSCCA) veröffentlichten GM/T Spezifikationsreihe definiert ist. Die SM2 Schlüsselspezifikation ist nur in China Regionen verfügbar. Wenn Sie die SM2 Schlüsselspezifikation verwenden, AWS KMS wird ein asymmetrischer KMS-Schlüssel mit einem SM2 key pair erstellt. Sie können Ihr SM2 key pair innerhalb von AWS KMS verwenden oder den öffentlichen Schlüssel für die Verwendung außerhalb von herunterladen AWS KMS. Weitere Informationen finden Sie unter the section called “Offline-Überprüfung mit SM2 Schlüsselpaaren (nur Regionen China)”.

Jeder KMS-Schlüssel kann nur einen habenKey usage. Sie können einen SM2 KMS-Schlüssel zum Signieren und Überprüfen, Verschlüsseln und Entschlüsseln oder zum Ableiten von gemeinsamen Geheimnissen verwenden. Sie müssen die Schlüsselverwendung bestimmen, wenn Sie den KMS-Schlüssel erstellen. Sie kann nach der Schlüsselerstellung nicht mehr geändert werden.

Wenn Sie einen asymmetrischen KMS-Schlüssel erstellen, um gemeinsame geheime Schlüssel abzuleiten, verwenden Sie die Schlüsselspezifikation. SM2 Der einzige unterstützte Schlüsselvereinbarungsalgorithmus zur Ableitung gemeinsamer Geheimnisse ist der Elliptic Curve Cryptography Cofactor Diffie-Hellman Primitive (ECDH).

AWS KMS unterstützt die folgenden Verschlüsselungs- und Signierungsalgorithmen: SM2

- SM2PKE-Verschlüsselungsalgorithmus

  SM2PKE ist ein auf elliptischen Kurven basierender Verschlüsselungsalgorithmus, der von OSCCA in 0003.4-2012 definiert wurde. GM/T

- SM2DSA-Signaturalgorithmus

  SM2DSA ist ein auf elliptischen Kurven basierender Signaturalgorithmus, der von OSCCA in 0003.2-2012 definiert wurde. GM/T SM2DSA benötigt eine identifizierende ID, die mit dem Hash-Algorithmus gehasht und dann mit der Nachricht SM3 oder dem Message Digest kombiniert wird, an die Sie übergeben haben. AWS KMS Dieser verkettete Wert wird dann gehasht und von signiert. AWS KMS

AWS KMS Berechtigungen

Diese Tabelle soll Ihnen helfen, die AWS KMS Berechtigungen zu verstehen, sodass Sie den Zugriff auf Ihre AWS KMS Ressourcen kontrollieren können. Definitionen der Spaltenüberschriften werden unter der Tabelle angezeigt.

Informationen zu AWS KMS Berechtigungen finden Sie auch in den Abschnitten Aktionen, Ressourcen und Bedingungsschlüssel zum AWS Key Management Service Thema Service Authorization Reference. In diesem Thema werden jedoch nicht alle Bedingungsschlüssel aufgeführt, die Sie zum Verfeinern jeder Berechtigung verwenden können.

Weitere Informationen darüber, welche AWS KMS Operationen für KMS-Schlüssel mit symmetrischer Verschlüsselung, asymmetrische KMS-Schlüssel und HMAC-KMS-Schlüssel gültig sind, finden Sie unter. Schlüsseltypreferenz

> **Note**
>
> Möglicherweise müssen Sie horizontal oder vertikal scrollen, um alle Daten in der Tabelle anzuzeigen.

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
CancelKeyDeletion `kms:CancelKeyDeletion`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService
ConnectCustomKeyStore `kms:ConnectCustomKeyStore`	IAM-Richtlinie	Nein	*	km: CallerAccount

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
CreateAlias `kms:CreateAlias`	IAM-Richtlinie (für den Alias)	Nein	Alias	Keine (bei der Steuerung des Zugriffs auf den Alias)
Um diese Produktion verwenden zu können, benötigt der Aufrufer die Berechtigung `kms:CreateAlias` für zwei Ressourcen: • Alias (in einer IAM-Richtlinie) • Der KMS-Schlüssel (in einer Schlüsselrichtlinie) Details hierzu finden Sie unter Steuern des Zugriffs auf Aliasse.	Schlüsselrichtlinie (für den KMS-Schlüssel)	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService
CreateCustomKeyStore `kms:CreateCustomKeyStore`	IAM-Richtlinie	Nein	*	km: CallerAccount

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
CreateGrant `kms:CreateGrant`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für den Verschlüsselungskontext: kms:EncryptionContext: Kontextschlüssel km: EncryptionContextKeys Bedingungen für Erteilungen: km: GrantConstraintType km: GranteePrincipal km: GrantIsFor AWSResource km: GrantOperations km: RetiringPrincipal Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
CreateKey `kms:CreateKey`	IAM-Richtlinie	Nein	*	km: BypassPolicyLockoutSafetyCheck km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ViaService aws:RequestTag/tag-key (AWS globaler Bedingungsschlüssel) aws:ResourceTag/tag-key (globaler Bedingungsschlüssel)AWS aws: TagKeys (AWS globaler Bedingungsschlüssel)

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
Decrypt `kms:Decrypt`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für kryptografische Operationen km: EncryptionAlgorithm km: RequestAlias Bedingungen für den Verschlüsselungskontext: kms:EncryptionContext: Kontextschlüssel km: EncryptionContextKeys Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService
DeleteAlias `kms:DeleteAlias`	IAM-Richtlinie (für den Alias)	Nein	Alias	Keine (bei der Steuerung des Zugriffs auf den Alias)
Um diese Produktion verwenden zu können, benötigt der Aufrufer die Berechtigung `kms:DeleteAlias` für zwei Ressourcen: • Alias (in einer IAM-Richtlinie) • Der KMS-Schlüssel (in einer Schlüsselrichtlinie) Details hierzu finden Sie unter Steuern des Zugriffs auf Aliasse.	Schlüsselrichtlinie (für den KMS-Schlüssel)	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
DeleteCustomKeyStore `kms:DeleteCustomKeyStore`	IAM-Richtlinie	Nein	*	km: CallerAccount
DeleteImportedKeyMaterial `kms:DeleteImportedKeyMaterial`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
DeriveSharedSecret `kms:DeriveSharedSecret`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Bedingungen für kryptografische Operationen km: KeyAgreementAlgorithm
DescribeCustomKeyStores `kms:DescribeCustomKeyStores`	IAM-Richtlinie	Nein	*	km: CallerAccount

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
DescribeKey `kms:DescribeKey`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen: km: RequestAlias

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
DisableKey `kms:DisableKey`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
DisableKeyRotation `kms:DisableKeyRotation`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService
DisconnectCustomKeyStore `kms:DisconnectCustomKeyStore`	IAM-Richtlinie	Nein	*	km: CallerAccount

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
EnableKey `kms:EnableKey`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
EnableKeyRotation `kms:EnableKeyRotation`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Bedingungen für die automatische Schlüsselrotation: km: RotationPeriodInDays

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
Encrypt `kms:Encrypt`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für kryptografische Operationen km: EncryptionAlgorithm km: RequestAlias Bedingungen für den Verschlüsselungskontext: kms:EncryptionContext: Kontextschlüssel km: EncryptionContextKeys Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
GenerateDataKey `kms:GenerateDataKey`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für kryptografische Operationen km: EncryptionAlgorithm km: RequestAlias Bedingungen für den Verschlüsselungskontext: kms:EncryptionContext: Kontextschlüssel km: EncryptionContextKeys Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
GenerateDataKeyPair `kms:GenerateDataKeyPair`	Schlüsselrichtlinie	Ja	KMS-Schlüssel Generierung eines asymmetrischen Datenschlüsselpaars, das durch einen KMS-Schlüssel mit symmetrischer Verschlüsselung geschützt ist.	Bedingungen für Datenschlüsselpaare: km: DataKeyPairSpec Bedingungen für kryptografische Operationen km: EncryptionAlgorithm km: RequestAlias Bedingungen für den Verschlüsselungskontext: kms:EncryptionContext: Kontextschlüssel km: EncryptionContextKeys Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
GenerateDataKeyPairWithoutPlaintext `kms:GenerateDataKeyPairWithoutPlaintext`	Schlüsselrichtlinie	Ja	KMS-Schlüssel Generierung eines asymmetrischen Datenschlüsselpaars, das durch einen KMS-Schlüssel mit symmetrischer Verschlüsselung geschützt ist.	Bedingungen für Datenschlüsselpaare: km: DataKeyPairSpec Bedingungen für kryptografische Operationen km: EncryptionAlgorithm km: RequestAlias Bedingungen für den Verschlüsselungskontext: kms:EncryptionContext: Kontextschlüssel km: EncryptionContextKeys Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
GenerateDataKeyWithoutPlaintext `kms:GenerateDataKeyWithoutPlaintext`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für kryptografische Operationen km: EncryptionAlgorithm km: RequestAlias Bedingungen für den Verschlüsselungskontext: kms:EncryptionContext: Kontextschlüssel km: EncryptionContextKeys Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService
GenerateMac `kms:GenerateMac`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Bedingungen für kryptografische Operationen km: MacAlgorithm km: RequestAlias

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
GenerateRandom `kms:GenerateRandom`	IAM-Richtlinie	N/A	*	Keine
GetKeyPolicy `kms:GetKeyPolicy`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
GetKeyRotationStatus `kms:GetKeyRotationStatus`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
GetParametersForImport `kms:GetParametersForImport`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	km: WrappingAlgorithm km: WrappingKeySpec Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
GetPublicKey `kms:GetPublicKey`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen: km: RequestAlias

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
ImportKeyMaterial `kms:ImportKeyMaterial`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen: km: ExpirationModel km: ValidTo
ListAliases `kms:ListAliases`	IAM-Richtlinie	Nein	*	Keine

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
ListGrants `kms:ListGrants`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen: km: GrantIsForAWSResource

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
ListKeyPolicies `kms:ListKeyPolicies`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
ListKeyRotations `kms:ListKeyRotations`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService
ListKeys `kms:ListKeys`	IAM-Richtlinie	Nein	*	Keine

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
ListResourceTags `kms:ListResourceTags`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
ListRetirableGrants `kms:ListRetirableGrants`	IAM-Richtlinie	Der angegebene Prinzipal muss sich im lokalen Konto befinden, aber die Produktion gibt Erteilungen in allen Konten zurück.	*	Keine

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
PutKeyPolicy `kms:PutKeyPolicy`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen: km: BypassPolicyLockoutSafetyCheck

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel

ReEncrypt
 	
`kms:ReEncryptFrom`
 	
`kms:ReEncryptTo`
 	
Um diese Produktion verwenden zu können, benötigt der Anrufer die Berechtigung für zwei KMS-Schlüssel:
 	
• `kms:ReEncryptFrom` auf dem KMS-Schlüssel zum Entschlüsseln
• `kms:ReEncryptTo` auf dem KMS-Schlüssel zum Verschlüsseln
	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für kryptografische Operationen
				
				km: EncryptionAlgorithm
				
				km: RequestAlias
				
				Bedingungen für den Verschlüsselungskontext:
				
				kms:EncryptionContext: Kontextschlüssel
				
				km: EncryptionContextKeys
				
				Bedingungen für KMS-Schlüssel-Operationen:
				
				km: CallerAccount
				
				km: KeySpec
				
				km: KeyUsage
				
				km: KeyOrigin
				
				km: MultiRegion
				
				km: MultiRegionKeyType
				
				km: ResourceAliases

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
				aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen: km: ReEncryptOnSameKey

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
ReplicateKey `kms:ReplicateKey` Um diese Produktion verwenden zu können, benötigt der Anrufer die folgenden Berechtigungen: • `kms:ReplicateKey` auf dem multiregionalen Primärschlüssel • `kms:CreateKey` in einer IAM-Richtlinie in der Replikatregion	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen: km: ReplicaRegion

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
RetireGrant `kms:RetireGrant` Die Berechtigung zur Außerbetriebnahme einer Erteilung wird hauptsächlich durch die Erteilung bestimmt. Eine Richtlinie allein kann den Zugriff auf diese Produktion nicht erlauben. Weitere Informationen finden Sie unter Außerbetriebnahme und Widerruf von Erteilungen.	IAM-Richtlinie (Diese Berechtigung ist in einer Schlüsselrichtlinie nicht wirksam.)	Ja	KMS-Schlüssel	Bedingungen für den Verschlüsselungskontext: kms:EncryptionContext: Kontextschlüssel km: EncryptionContextKeys Bedingungen für Erteilungen: km: GrantConstraintType Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
RevokeGrant `kms:RevokeGrant`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen: km: GrantIsFor AWSResource

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
RotateKeyOnDemand `kms:RotateKeyOnDemand`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
ScheduleKeyDeletion `kms:ScheduleKeyDeletion`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
Sign `kms:Sign`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für Signatur und Verifizierung: km: MessageType km: RequestAlias km: SigningAlgorithm Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
TagResource `kms:TagResource`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Bedingungen für Markierung: aws:RequestTag/tag-key (AWS globaler Bedingungsschlüssel) aws: TagKeys (AWS globaler Bedingungsschlüssel)

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
UntagResource `kms:UntagResource`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Bedingungen für Markierung: aws:RequestTag/tag-key (AWS globaler Bedingungsschlüssel) aws: TagKeys (AWS globaler Bedingungsschlüssel)

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
UpdateAlias `kms:UpdateAlias` Um diese Produktion verwenden zu können, benötigt der Aufrufer die Berechtigung `kms:UpdateAlias` für drei Ressourcen: • Alias • Der aktuell zugeordnete KMS-Schlüssel • Der neu zugeordnete KMS-Schlüssel Details hierzu finden Sie unter Steuern des Zugriffs auf Aliasse.	IAM-Richtlinie (für den Alias)	Nein	Alias	Keine (bei der Steuerung des Zugriffs auf den Alias)
	Schlüsselrichtlinie (für die KMS-Schlüssel)	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService
UpdateCustomKeyStore `kms:UpdateCustomKeyStore`	IAM-Richtlinie	Nein	*	km: CallerAccount

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
UpdateKeyDescription `kms:UpdateKeyDescription`	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
UpdatePrimaryRegion `kms:UpdatePrimaryRegion` Um diese Produktion verwenden zu können, benötigt der Aufrufer die `kms:UpdatePrimaryRegion` -Berechtigung sowohl für den multiregionalen Primärschlüssel, der zu einem Replikatschlüssel wird, und den multiregionalen Replikatschlüssel, der zum Primärschlüssel wird.	Schlüsselrichtlinie	Nein	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Andere Bedingungen km: PrimaryRegion

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
Verify `kms:Verify`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für Signatur und Verifizierung: km: MessageType km: RequestAlias km: SigningAlgorithm Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService

Aktionen und Berechtigungen	Richtlinientyp	Kontoübergreifende Verwendung	Ressourcen (für IAM-Richtlinien)	AWS KMS Bedingungsschlüssel
VerifyMac `kms:VerifyMac`	Schlüsselrichtlinie	Ja	KMS-Schlüssel	Bedingungen für KMS-Schlüssel-Operationen: km: CallerAccount km: KeySpec km: KeyUsage km: KeyOrigin km: MultiRegion km: MultiRegionKeyType km: ResourceAliases aws:ResourceTag/tag-key (AWS globaler Bedingungsschlüssel) km: ViaService Bedingungen für kryptografische Operationen km: MacAlgorithm km: RequestAlias

Beschreibungen der Spalten

Die Spalten in dieser Tabelle enthalten folgende Informationen:

- Unter Aktionen und Berechtigungen werden alle AWS KMS API-Operationen und die Berechtigungen aufgeführt, die den Vorgang zulassen. Sie geben die Produktion im `Action`-Element einer Richtlinienanweisung an.
- Policy type (Richtlinientyp) gibt an, ob die Berechtigung in einer Schlüsselrichtlinie oder einer IAM-Richtlinie verwendet werden kann.

  Key policy (Schlüsselrichtlinie) bedeutet, dass Sie die Berechtigung in der Schlüsselrichtlinie angeben können. Wenn die Schlüsselrichtlinie die Richtlinienanweisung zur Aktivierung von IAM-Richtlinien enthält, können Sie die Berechtigung in einer IAM-Richtlinie angeben.

  IAM policy (IAM-Richtlinie) bedeutet, dass Sie die Berechtigung nur in einer IAM-Richtlinie angeben können.
- Kontenübergreifende Verwendung zeigt die Operationen an, die autorisierte Benutzer für Ressourcen in einem anderen AWS-Konto ausführen können.

  Ein Wert von Ja bedeutet, dass Prinzipale die Produktion für Ressourcen in einem anderen AWS-Konto ausführen können.

  Ein Wert von No (Nein) bedeutet, dass Prinzipale die Produktion nur auf Ressourcen in ihrem eigenen AWS-Konto ausführen können.

  Wenn Sie einem Prinzipal in einem anderen Konto eine Berechtigung erteilen, die nicht für eine kontoübergreifende Ressource verwendet werden kann, ist die Berechtigung nicht wirksam. Wenn Sie beispielsweise einem Prinzipal in einem anderen Konto die TagResource Berechtigung kms: für einen KMS-Schlüssel in Ihrem Konto erteilen, schlagen seine Versuche fehl, den KMS-Schlüssel in Ihrem Konto zu kennzeichnen.
- Resources listet die AWS KMS Ressourcen auf, für die die Berechtigungen gelten. AWS KMS unterstützt zwei Ressourcentypen: einen KMS-Schlüssel und einen Alias. In einer Schlüsselrichtlinie ist der Wert des `Resource`-Elements stets `*`. Dies gibt den KMS-Schlüssel an, dem die Schlüsselrichtlinie angefügt ist.

  Verwenden Sie die folgenden Werte, um eine AWS KMS Ressource in einer IAM-Richtlinie darzustellen.

  KMS-Schlüssel

  Wenn es sich bei der Ressource um einen KMS-Schlüssel handelt, verwenden Sie dessen Schlüssel-ARN. Weitere Informationen dazu finden Sie unter the section called “Finden Sie die Schlüssel-ID und den Schlüssel-ARN”.

`arn:AWS_partition_name:kms:AWS_Region:AWS_account_ID:key/key_ID`

Zum Beispiel:

arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab

Alias

Wenn es sich bei der Ressource um einen Alias handelt, verwenden Sie den Alias-ARN. Weitere Informationen dazu finden Sie unter the section called "Suchen Sie den Aliasnamen und den Alias-ARN".

`arn:AWS_partition_name:kms:AWS_region:AWS_account_ID:alias/alias_name`

Zum Beispiel:

arn:aws:kms:us-west- 2:111122223333:alias/ ExampleAlias

***** (Sternchen)

Wenn die Berechtigung nicht für eine bestimmte Ressource (KMS-Schlüssel oder Alias) gilt, verwenden Sie ein Sternchen (`*`).

In einer IAM-Richtlinie für eine Berechtigung steht ein Sternchen im Element für alle Ressourcen (KMS-Schlüssel und Aliase). AWS KMS `Resource` AWS KMS Sie können in dem `Resource` Element auch ein Sternchen verwenden, wenn die AWS KMS Berechtigung nicht für bestimmte KMS-Schlüssel oder -Aliase gilt. Wenn Sie beispielsweise eine `kms:ListKeys` Erlaubnis zulassen `kms:CreateKey` oder verweigern, müssen Sie das `Resource` Element auf setzen. `*`

- AWS KMS Bedingungstasten listet die AWS KMS Bedingungsschlüssel auf, mit denen Sie den Zugriff auf den Vorgang steuern können. Sie geben Bedingungen im `Condition`-Element einer Richtlinie an. Weitere Informationen finden Sie unter AWS KMS Bedingungsschlüssel. Diese Spalte enthält auch AWS globale Bedingungsschlüssel, die von AWS KMS, aber nicht von allen AWS Diensten unterstützt werden.

AWS KMS interne Operationen

AWS Key Management Service (AWS KMS) bietet kryptografische Schlüssel und Operationen, die durch FIPS 140-3 Security Level 3-validierte Hardware-Sicherheitsmodule (HSM) gesichert sind, die

für die Cloud skaliert sind. AWS KMS Schlüssel und Funktionen werden von mehreren AWS Cloud-Diensten verwendet, und Sie können sie verwenden, um Daten in Ihren Anwendungen zu schützen. Dieser technische Leitfaden enthält Einzelheiten zu den kryptografischen Vorgängen, die AWS bei der Verwendung AWS KMS ausgeführt werden.

AWS KMS HSMs Für die Skalierung und Sicherheit eines weltweit verteilten Schlüsselverwaltungsdienstes sind interne Komponenten erforderlich.

Themen

- Domänen und Domänenstatus
- Interne Kommunikationssicherheit
- Replikationsprozess für Schlüssel mit mehreren Regionen
- Beständigkeitsschutz

Domänen und Domänenstatus

Eine kooperative Sammlung vertrauenswürdiger interner AWS KMS Entitäten innerhalb einer AWS-Region wird als Domäne bezeichnet. Eine Domäne umfasst einen Satz vertrauenswürdiger Entitäten, einen Satz von Regeln und einen Satz geheimer Schlüssel, die als Domänenschlüssel bezeichnet werden. Die Domänenschlüssel werden von HSMs allen Mitgliedern der Domäne gemeinsam genutzt. Ein Domänenstatus besteht aus den folgenden Feldern.

Name

Ein Domänenname zur Identifizierung dieser Domäne.

Mitglieder

Eine Liste HSMs dieser Mitglieder der Domäne, einschließlich ihrer öffentlichen Signaturschlüssel und der Schlüssel für öffentliche Vereinbarungen.

Operatoren

Eine Liste von Entitäten, öffentlichen Signaturschlüsseln und einer Rolle (AWS KMS Betreiber oder Service-Host), die die Betreiber dieses Dienstes repräsentiert.

Regeln

Eine Liste der Quorumregeln für jeden Befehl, der erfüllt sein muss, um einen Befehl auf dem HSM auszuführen.

Domänenschlüssel

Eine Liste der Domänenschlüssel (symmetrische Schlüssel), die derzeit in der Domäne verwendet werden.

Der vollständige Domänenstatus ist nur im HSM verfügbar. Der Domänenstatus wird zwischen HSM-Domänenmitgliedern als exportiertes Domänen-Token synchronisiert.

Domänenschlüssel

Alle Mitglieder HSMs einer Domain teilen sich einen Satz von Domänenschlüsseln, $\{DK_r\}$. Diese Schlüssel werden über eine Exportroutine für Domänenstatus freigegeben. Der exportierte Domänenstatus kann in jedes HSM importiert werden, das Mitglied der Domäne ist.

Der Satz von Domänenschlüsseln $\{DK_r\}$enthält immer einen aktiven Domänenschlüssel und mehrere deaktivierte Domänenschlüssel. Die Domainschlüssel werden täglich gewechselt, um sicherzustellen, dass sie der Empfehlung für die Schlüsselverwaltung — Teil 1 AWS entsprechen. Während der Domänenschlüsseldrehung werden alle vorhandenen KMS-Schlüssel, die unter dem ausgehenden Domänenschlüssel verschlüsselt wurden, unter dem neuen aktiven Domänenschlüssel neu verschlüsselt. Der aktive Domainschlüssel wird verwendet, um alle neuen EKTs zu verschlüsseln. Die abgelaufenen Domänenschlüssel können nur verwendet werden, um zuvor verschlüsselte Domänenschlüssel EKTs für eine Anzahl von Tagen zu entschlüsseln, die der Anzahl der kürzlich rotierten Domänenschlüssel entspricht.

Exportierte Domänen-Token

Es besteht eine regelmäßige Notwendigkeit, den Status zwischen Domänenteilnehmern zu synchronisieren. Dies wird durch den Export des Domänenstatus erreicht, wenn eine Änderung an der Domäne vorgenommen wird. Der Domänenstaus wird als exportiertes Domänen-Token exportiert.

Name

Ein Domänenname zur Identifizierung dieser Domäne.

Mitglieder

Eine Liste dieser Mitglieder der Domain HSMs , einschließlich der öffentlichen Schlüssel, die sie unterzeichnet haben und denen sie zugestimmt haben.

Operatoren

Eine Liste von Entitäten, öffentlichen Signaturschlüsseln und einer Rolle, die die Operatoren dieses Services darstellt.

Regeln

Eine Liste der Quorumregeln für jeden Befehl, der erfüllt sein muss, um einen Befehl auf einem HSM-Domänenmitglied auszuführen.

Verschlüsselte Domänenschlüssel

Envelope-verschlüsselte Domänenschlüssel. Die Domänenschlüssel werden vom unterzeichnenden Mitglied für jedes der oben aufgeführten Mitglieder verschlüsselt und in seinen öffentlichen Vereinbarungsschlüssel Envelope-verschlüsselt.

Signatur

Eine Signatur im Domänenstatus, die von einem HSM erstellt wurde, notwendigerweise ein Mitglied der Domäne, die den Domänenstatus exportiert hat.

Das exportierte Domänen-Token bildet die grundlegende Vertrauensquelle für Entitäten, die innerhalb der Domäne arbeiten.

Verwalten von Domänenstatus

Der Domänenstatus wird über quorumauthentifizierte Befehle verwaltet. Zu diesen Änderungen gehören das Ändern der Liste der vertrauenswürdigen Teilnehmer in der Domäne, das Ändern der Quorumregeln zum Ausführen von HSM-Befehlen und das regelmäßige Drehen der Domänenschlüssel. Diese Befehle werden pro Befehl authentifiziert, im Gegensatz zu authentifizierten Sitzungsvorgängen, wie im folgenden Bild gezeigt.

Ein HSM enthält in seinem initialisierten und betriebsbereiten Status einen Satz selbsterzeugter asymmetrischer Identitätsschlüssel, ein Signaturschlüsselpaar und ein Schlüsselpaar zur Schlüsseleinrichtung. Durch einen manuellen Prozess kann ein AWS KMS Betreiber eine erste Domain einrichten, die auf einem ersten HSM in einer Region erstellt werden soll. Diese anfängliche Domäne besteht aus einem vollständigen Domänenstatus, wie zuvor in diesem Thema definiert. Es wird über einen Join-Befehl zu jedem der definierten HSM-Mitglieder in der Domäne installiert.

Nachdem ein HSM einer anfänglichen Domäne beigetreten ist, ist es an die Regeln gebunden, die in dieser Domäne definiert sind. Diese Regeln regeln die Befehle, die Kryptographieschlüssel

des Kunden verwenden oder Änderungen am Host- oder Domänenstatus vornehmen. Die authentifizierten Sitzungs-API-Operationen, die Ihre kryptografischen Schlüssel verwenden, wurden früher definiert.

Das obige Bild zeigt, wie ein Domänenstatus geändert wird. Der Prozess besteht aus vier Schritten:

1. Ein quorumbasierter Befehl wird an ein HSM gesendet, um die Domäne zu ändern.
2. Ein neuer Domänenstatus wird generiert und als neues exportiertes Domänentoken exportiert. Der Status auf dem HSM wird nicht geändert, was bedeutet, dass die Änderung nicht auf dem HSM übernommen wird.
3. Ein zweiter Befehl wird an jedes der HSMs neu exportierten Domain-Token gesendet, um ihren Domain-Status mit dem neuen Domain-Token zu aktualisieren.
4. Das im neu exportierten Domain-Token HSMs aufgelistete Token kann den Befehl und das Domain-Token authentifizieren. Sie können auch die Domänenschlüssel entpacken, um den Domänenstatus für alle HSMs in der Domäne zu aktualisieren.

HSMs kommunizieren nicht direkt miteinander. Stattdessen fordert ein Quorum von Operatoren eine Änderung des Domänenstatus an, die zu einem neuen exportierten Domänen-Token führt. Ein Servicehostmitglied der Domäne wird verwendet, um den neuen Domänenstatus an jedes HSM in der Domäne zu verteilen.

Das Verlassen und Verbinden einer Domäne erfolgt über die HSM-Verwaltungsfunktionen. Die Änderung des Domänenstatus erfolgt über die Domänenverwaltungsfunktionen.

Verlassen der Domäne

Bewirkt, dass ein HSM eine Domäne verlässt und alle Reste und Schlüssel dieser Domäne aus dem Speicher löscht.

Registrieren der Domäne

Bewirkt, dass ein HSM einer neuen Domäne beitritt oder seinen aktuellen Domänenstatus auf den neuen Domänenstatus aktualisiert. Die vorhandene Domäne wird als Quelle des anfänglichen Regelsatzes verwendet, um diese Nachricht zu authentifizieren.

Domäne erstellen

Bewirkt, dass eine neue Domäne auf einem HSM erstellt wird. Gibt ein erstes Domain-Token zurück, das an ein Mitglied HSMs der Domain verteilt werden kann.

Operatoren ändern

Fügt Operatoren und deren Rollen in der Domäne hinzu oder entfernt sie aus der Liste der autorisierten Operatoren.

Ändern von Mitgliedern

Fügt der Liste der HSMs in der Domäne autorisierten HSM ein HSM hinzu oder entfernt es daraus.

Regeln ändern

Ändert den Satz von Quorumregeln, die zum Ausführen von Befehlen auf einem HSM erforderlich sind.

Drehen der Domänenschlüssel

Bewirkt, dass ein neuer Domänenschlüssel erstellt und als aktiver Domänenschlüssel markiert wird. Dadurch wird der vorhandene aktive Schlüssel in einen deaktivierten Schlüssel verschoben und der älteste deaktivierte Schlüssel aus dem Domänenstatus entfernt.

Interne Kommunikationssicherheit

Befehle zwischen den Dienst-Hosts oder AWS KMS -Operatoren und den HSMs werden durch zwei Mechanismen gesichert, die unter beschrieben sindAuthentifizierte Sitzungen: eine quorumsignierte Anforderungsmethode und eine authentifizierte Sitzung unter Verwendung eines HSM-Service-Hostprotokolls.

Die quorumsignierten Befehle sind so konzipiert, dass kein einzelner Bediener die kritischen Sicherheitsvorkehrungen, die sie bieten, ändern kann. HSMs Die Befehle, die über die authentifizierten Sitzungen ausgeführt werden, tragen dazu bei, dass nur autorisierte Service-Operatoren Operationen mit KMS-Schlüsseln ausführen können. Alle an Kunden gebundenen geheimen Informationen sind in der gesamten Infrastruktur gesichert. AWS

Schlüssel-Einrichtung

AWS KMS Verwendet zur Sicherung der internen Kommunikation zwei verschiedene wichtige Methoden zur Einrichtung. Die erste Methode ist als C (1, 2, ECC DH) in der Empfehlung für Verfahren zur paarweisen Schlüsselerstellung unter Verwendung diskreter Logarithmus-Kryptografie definiert. Dieses Schema hat einen Initiator mit einem statischen Signaturschlüssel. Der Initiator generiert und signiert einen ephemeren elliptischen Kurven-Diffie-Hellman-Schlüssel (ECDH), der

für einen Empfänger mit einem statischen ECDH-Übereinstimmungsschlüssel bestimmt ist. Diese Methode verwendet einen ephemeren Schlüssel und zwei statische Schlüssel mit ECDH. Das ist die Ableitung der Markierung C (1, 2, ECC DH). Diese Methode wird manchmal als One-Pass-ECDH bezeichnet.

Die zweite Methode zur Schlüsseleinrichtung ist C(2, 2, ECC, DH). In diesem Schema haben beide Parteien einen statischen Signaturschlüssel, und sie generieren, signieren und tauschen einen ephemeren ECDH-Schlüssel aus. Diese Methode verwendet zwei statische Schlüssel und zwei ephemere Schlüssel, die jeweils ECDH verwenden. Das ist die Ableitung der Markierung C (2, 2, ECC DH). Diese Methode wird manchmal als ephemere ECDH oder ECDHE bezeichnet. Alle ECDH-Schlüssel werden auf der Kurve secp384r1 (NIST-P384) generiert.

HSM-Sicherheitsgrenze

Die innere Sicherheitsgrenze von ist das HSM. AWS KMS Das HSM verfügt über eine proprietäre Schnittstelle und keine anderen aktiven physikalischen Schnittstellen im Betriebszustand. Ein operatives HSM wird während der Initialisierung mit den erforderlichen kryptografischen Schlüsseln bereitgestellt, um seine Rolle in der Domäne festzulegen. Sensible kryptografische Materialien des HSM werden nur im flüchtigen Speicher gespeichert und gelöscht, wenn das HSM den Betriebszustand verlässt, einschließlich beabsichtigter oder unbeabsichtigter Abschaltungen oder Rücksetzungen.

Die HSM-API-Operationen werden entweder durch einzelne Befehle oder über eine gegenseitig authentifizierte vertrauliche Sitzung authentifiziert, die von einem Service-Host eingerichtet wird.

Quorumsignierte Befehle

Befehle mit Quorumsignierung werden von Operatoren an ausgegeben. HSMs In diesem Abschnitt wird beschrieben, wie quorumbasierte Befehle erstellt, signiert und authentifiziert werden. Diese Regeln sind ziemlich einfach. Beispielsweise erfordert der Befehl Foo, dass zwei Mitglieder der Rolle Bar authentifiziert werden. Es gibt drei Schritte bei der Erstellung und Überprüfung eines quorumbasierten Befehls. Der erste Schritt ist die anfängliche Befehlserstellung; die zweite ist die Vorlage an zusätzliche Operatoren zur Unterzeichnung; und die dritte ist die Überprüfung und Ausführung.

Gehen Sie zur Einführung der Konzepte davon aus, dass es einen authentischen Satz von öffentlichen Schlüsseln und Rollen des Operators $\{QOS_s\}$ und einen Satz von Quorum-Regeln QR = {Command, $Rule_{\{i, t\}}$} $gibt_i$, wobei jede Regel aus einer Gruppe von Rollen und der Mindestanzahl N {Role, N} besteht. $_t$ $_t$ Damit ein Befehl die Quorumregel erfüllt, muss die Befehlsdatei von einer in

{QOS_s} aufgelisteten Gruppe von Operatoren signiert sein, sodass sie eine der für diesen Befehl aufgeführten Regeln erfüllen. Wie bereits erwähnt, werden die Quorumregeln und Operatoren im Domänenstatus und im exportierten Domänen-Token gespeichert.

In der Praxis signiert ein Initialsignierer den Befehl Sig_1 = Sign(dO_{p1}, Befehl) . Ein zweiter Operator signiert auch den Befehl Sig_2 = Sign(dO_{p2}, Befehl) . Die doppelt signierte Nachricht wird zur Ausführung an ein HSM gesendet. Die HMS führt Folgendes aus:

1. Für jede Signatur wird der öffentliche Schlüssel des Unterzeichners aus dem Domainstatus extrahiert und die Signatur des Befehls überprüft.
2. Es überprüft, ob die Gruppe von Unterzeichnern eine Regel für den Befehl erfüllt.

Authentifizierte Sitzungen

Ihre wichtigsten Operationen laufen zwischen den nach außen gerichteten AWS KMS Hosts und dem. HSMs Diese Befehle beziehen sich auf die Erstellung und Verwendung von kryptografischen Schlüsseln und die sichere Zufallszahlengenerierung. Die Befehle werden über einen sitzungsauthentifizierten Kanal zwischen den Service-Hosts und dem ausgeführt. HSMs Neben der Notwendigkeit der Authentizität erfordern diese Sitzungen Vertraulichkeit. Befehle, die über diese Sitzungen ausgeführt werden, umfassen die Rückgabe von Klartext-Datenschlüsseln und entschlüsselten Nachrichten, die für Sie bestimmt sind. Um sicherzustellen, dass diese Sitzungen nicht durch man-in-the-middle Angriffe unterlaufen werden können, werden Sitzungen authentifiziert.

Dieses Protokoll führt eine gegenseitig authentifizierte ECDHE-Schlüsselvereinbarung zwischen dem HSM und dem Service-Host durch. Der Austausch wird vom Service-Host initiiert und vom HSM abgeschlossen. Der HSM gibt auch einen durch den ausgehandelten Schlüssel verschlüsselten Sitzungsschlüssel (SK) und einen exportierten Schlüssel-Token zurück, der den Sitzungsschlüssel enthält. Das exportierte Schlüssel-Token enthält einen Gültigkeitszeitraum, nach dem der Service-Host einen Sitzungsschlüssel neu aushandeln muss.

Ein Service-Host ist Mitglied der Domain und verfügt über ein Identity-Signing-Schlüsselpaar ($DHOS_i$, $QHOS_i$) und eine authentische Kopie der öffentlichen Identitätsschlüssel HSMs. Es verwendet seinen Satz von Identitätssignierungsschlüsseln, um sicher einen Sitzungsschlüssel auszuhandeln, der zwischen dem Service-Host und jedem HSM in der Domäne verwendet werden kann. Den exportierten Schlüssel-Token ist ein Gültigkeitszeitraum zugeordnet, nach dem ein neuer Schlüssel ausgehandelt werden muss.

Der Prozess beginnt damit, dass der Service-Host erkennt, dass er einen Sitzungsschlüssel benötigt, um vertrauliche Kommunikationsflüsse zwischen ihm und einem HSM-Mitglied der Domäne zu senden und zu empfangen.

1. Ein Service-Host generiert ein ephemeres ECDH-Schlüsselpaar (d_1, Q_1) und signiert es mit seinem Identitätsschlüssel Sig1 = Sign(dOS,Q_1).
2. Das HSM überprüft die Signatur auf dem empfangenen öffentlichen Schlüssel mit seinem aktuellen Domänen-Token und erstellt ein ephemeres ECDH-Schlüsselpaar (d_2, Q_2). Anschließend vervollständigt er das Verfahren ECDH-key-exchange gemäß der Empfehlung für paarweise Verfahren zur Einrichtung von Schlüsseln unter Verwendung diskreter Logarithmus-Kryptografie, um einen ausgehandelten 256-Bit-AES-GCM-Schlüssel zu bilden. Das HSM generiert einen neuen 256-Bit-AES-GCM-Sitzungsschlüssel. Es verschlüsselt den Sitzungsschlüssel mit dem ausgehandelten Schlüssel, um den verschlüsselten Sitzungsschlüssel (ESK) zu bilden. Es verschlüsselt auch den Sitzungsschlüssel unter dem Domänenschlüssel als exportiertes Schlüssel-Token EKT. Schließlich signiert es einen Rückgabewert mit seinem Identitätsschlüsselpaar Sig_2 = Sign(dHSK, (Q_2, ESK, EKT)).
3. Der Service-Host überprüft die Signatur für die empfangenen Schlüssel mit seinem aktuellen Domänen-Token. Der Service-Host schließt dann den ECDH-Schlüsselaustausch gemäß der Empfehlung für paarweise Schlüsselerstellungsschemata unter Verwendung diskreter Logarithmus-Kryptografie ab. Es entschlüsselt als nächstes den ESK, um den Sitzungsschlüssel SK zu erhalten.

Während der Gültigkeitsdauer im EKT kann der Service-Host den ausgehandelten Sitzungsschlüssel SK verwenden, um mit einen Envelope-verschlüsselten Befehl an das HSM zu senden. Jeder service-host-initiated Befehl über diese authentifizierte Sitzung beinhaltet das EKT. Der HSM antwortet mit demselben ausgehandelten Sitzungsschlüssel SK.

Replikationsprozess für Schlüssel mit mehreren Regionen

AWS KMS verwendet einen regionsübergreifenden Replikationsmechanismus, um das Schlüsselmaterial in einem KMS-Schlüssel von einem HSM in einem in ein HSM in einem anderen AWS-Region zu kopieren. AWS-Region Damit dieser Mechanismus funktioniert, muss der zu replizierende KMS-Schlüssel ein Multiregions-Schlüssel sein. Bei der Replikation eines KMS-Schlüssels von einer Region HSMs in eine andere können die Regionen nicht direkt kommunizieren, da sie sich in isolierten Netzwerken befinden. Stattdessen werden die während der regionsübergreifenden Replikation ausgetauschten Nachrichten von einem Proxyservice übermittelt.

Bei der regionsübergreifenden Replikation wird jede von einem AWS KMS HSM generierte Nachricht mithilfe eines Replikationssignaturschlüssels kryptografisch signiert. Replikationssignaturschlüssel (RSKs) sind ECDSA-Schlüssel auf der NIST-P-384-Kurve. Jede Region besitzt mindestens eine RSK, und die öffentliche Komponente jeder RSK wird von allen anderen Regionen in derselben Partition gemeinsam genutzt. AWS

Der regionsübergreifende Replikationsprozess zum Kopieren von Schlüsselmaterial von Region A nach Region B funktioniert folgendermassen:

1. Das HSM in Region B erzeugt einen flüchtigen ECDH-Schlüssel auf der NIST-P-384-Kurve, Replikationsvereinbarungs-Schlüssel B (RAKB). Die öffentliche Komponente von RAKB wird vom Proxy-Dienst an ein HSM in Region A gesendet.
2. Das HSM in Region A erhält die öffentliche Komponente von RAKB und generiert dann einen weiteren flüchtigen ECDH-Schlüssel auf der NIST-P-384-Kurve, Replikationsvereinbarungs-Schlüssel A (RAKA). Dazu führt das HSM das ECDH-Schlüsselherstellungsschema für RAKA und die öffentliche Komponente von RAKB aus und leitet aus der Ausgabe einen symmetrischen Schlüssel ab, den Replication Wrapping Key (RWK). Die RWK wird verwendet, um das Schlüsselmaterial des multiregionalen KMS-Schlüssels zu verschlüsseln, der repliziert wird.
3. Die öffentliche Komponente von RAKA und das mit der RWK verschlüsselte Schlüsselmaterial werden über den Proxy-Service an das HSM in Region B gesendet.
4. Das HSM in Region B erhält die öffentliche Komponente von RAKA und das Schlüsselmaterial, das mit der RWK verschlüsselt wurde. Das HSM wird von RWK abgeleitet, indem das ECDH-Schlüsselherstellungsschema auf RAKB und der öffentlichen Komponente von RAKA ausgeführt wird.
5. Das HSM in Region B verwendet die RWK, um das Schlüsselmaterial aus Region A zu entschlüsseln.

Beständigkeitsschutz

Eine zusätzliche Lebensdauer der durch den Dienst generierten Schlüssel wird durch die Verwendung von HSMs Offline-Speichern mehrerer nichtflüchtiger exportierter Domain-Tokens und die redundante Speicherung verschlüsselter KMS-Schlüssel gewährleistet. Die HSMs Offline-Benutzer sind Mitglieder der vorhandenen Domänen. Mit der Ausnahme, dass sie nicht online sind und an den regulären Domänenoperationen teilnehmen, werden die HSMs Offline-Benutzer im Domänenstatus genauso angezeigt wie die vorhandenen HSM-Mitglieder.

Das langlebige Design soll alle KMS-Schlüssel in einer Region schützen, falls es zu einem AWS umfassenden Verlust der Online-Schlüssel HSMs oder der KMS-Schlüsselsätze kommen sollte, die in unserem primären Speichersystem gespeichert sind. AWS KMS keys bei importiertem Schlüsselmaterial fallen nicht unter die Haltbarkeitsschutzmaßnahmen, die andere KMS-Schlüssel bieten. Im Falle eines regionsweiten Fehlers muss AWS KMS importiertes Schlüsselmaterial möglicherweise erneut in einen KMS-Schlüssel importiert werden.

Die HSMs Offline-Daten und die Zugangsdaten für den Zugriff werden in Safes in überwachten Sicherheitsräumen an mehreren unabhängigen geografischen Standorten gespeichert. Für jeden Tresor sind mindestens ein AWS Sicherheitsbeauftragter und ein AWS KMS Bediener aus zwei unabhängigen Teams erforderlich AWS, um diese Materialien zu beschaffen. Die Verwendung dieser Materialien wird durch interne Richtlinien geregelt, nach denen ein Quorum von AWS KMS Bedienungspersonal anwesend sein muss.

Dokumentverlauf

In diesem Thema werden wichtige Aktualisierungen im AWS Key Management Service - Entwicklerhandbuch beschrieben.

Themen

- Neueste Aktualisierungen
- Frühere Aktualisierungen

Neueste Aktualisierungen

Die folgende Tabelle beschreibt signifikante Änderungen an dieser Dokumentation seit Januar 2018. Neben den hier aufgelisteten größeren Änderungen aktualisieren wir die Dokumentation regelmäßig überarbeitet, um Beschreibungen und Beispiele zu verbessern und Ihre Rückmeldungen zu berücksichtigen. Wenn Sie über wichtige Änderungen benachrichtigt werden möchten, abonnieren Sie den RSS-Feed.

Möglicherweise müssen Sie horizontal oder vertikal scrollen, um alle Daten in dieser Tabelle anzuzeigen.

Änderung	Beschreibung	Datum
Dual-Stack-Unterstützung	AWS KMS unterstützt Dual-Stack.	18. Juni 2025
Feature-Update	Fügt Unterstützung für den Module-Lattice Digital Signature Algorithm (ML-DSA) für kryptografische Signature n nach dem Quantenprinzip hinzu.	13. Juni 2025
Importierte Schlüsselrotation	Sie können bei Bedarf eine Rotation von KMS-Schlüsseln mit symmetris cher Verschlüsselung mit	5. Juni 2025

	importiertem Schlüssel material (`EXTERNAL`Ursprung) durchführen.	
Feature-Update	Unterstützung für KMS-Schlüssel mit mehreren Regionen in China Regionen hinzugefügt.	21. November 2024
AWS verwaltetes Richtlinienupdate	Die AWSKeyManagementServiceMultiRegionKeysServiceRolePolicydienstbezogene Rolle wurde aktualisiert, indem der verwalteten Richtlinie mit der Richtlinienversion v2 eine Anweisungs-ID (`Sid`) hinzugefügt wurde.	21. November 2024
Kontingentänderung	Die Standardanforderate für ImportKeyMaterialDeleteImportedKeyMaterialUND-Anfragen wurde erhöht.	23. Juli 2024
Kontingentänderung	Die Anforderungsrate für standardmäßige kryptografische Operationen für KMS-Schlüssel, RSA-KMS-Schlüssel sowie ECC- und KMS-Schlüssel mit symmetrischer Verschlüsselung wurde erhöht. SM2	8. Juli 2024

Neues Feature	Es wurde ein neuer `KeyUsage` Typ `KEY_AGREEMENT` für von NIST empfohlene KMS-Schlüssel mit elliptischen Kurven (ECC) und SM2 (nur Regionen China) hinzugefügt und Unterstützung für die Ableitung von gemeinsamen Geheimnissen hinzugefügt.	13. Juni 2024
Aktualisierungen der Schlüsselrotation	Unterstützung für benutzerdefinierte Rotationsperioden für automatische Schlüsselrotationen, On-Demand-Schlüsselrotationen und Einblick in Ihre wichtigsten Materialrotationen hinzugefügt.	12. April 2024
Aktualisierungen der verwalteten Richtlinie	Es wurden neue Berechtigungen hinzugefügt`AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy`, die es AWS KMS ermöglichen, Änderungen in der VPC, die Ihren AWS CloudHSM Cluster enthält, zu überwachen, sodass bei Ausfällen klare Fehlermeldungen ausgegeben werden AWS KMS können.	10. November 2023
Feature-Update	Unterstützung für den `DryRun`-API-Parameter hinzugefügt.	5. Juli 2023

Feature-Update	Unterstützung für den Import von Schlüsselmaterial für alle Arten von AWS KMS Schlüsseln hinzugefügt, mit Ausnahme von benutzerd efinierten Schlüsselspeichern.	5. Juni 2023
Feature-Update	Aktualisierungen AWS KMS APIs für Nitro Enclaves	10. März 2023
Feature-Update	Der `RSAES_PKCS1_V1_5` Wrapping-Algorithmus ist veraltet. AWS KMS wird gemäß den Richtlinien des National Institute of Standards and Technology (NIST) zur Verwaltung kryptografischer Schlüssel bis zum 1. Oktober 2023 jegliche Unterstüt zung einstellen. `RSAES_PKC S1_V1_5` Wir empfehlen , sofort mit der Verwendun g eines anderen Wrapping-Algorithmus zu beginnen.	28. Februar 2023
Feature-Update	Zusätzliche Unterstützung für externe Schlüsselspeicher, eine Funktion, mit der Sie Ihre AWS Ressourcen schützen können, indem Sie kryptogra fische Schlüssel außerhalb von verwenden. AWS	29. November 2022

Kontingentänderung	Das AWS KMS keys Ressourcenkontingent wurde für jedes Konto und jede Region auf 100.000 KMS-Schlüssel erhöht.	8. Juli 2022
Funktionsupdate	Unterstützung für HMAC-KMS-Schlüssel wurde in weiteren Bereichen hinzugefügt AWS-Regionen	8. Juli 2022
Neues Thema	Das AWS Key Management Service Thema Resilienz wurde dem Kapitel Sicherheit des AWS KMS Entwicklerhandbuchs hinzugefügt.	14. Juni 2022
Neues Feature	Unterstützung für AWS KMS Schlüssel und API-Operationen, die HMAC-Codes generieren und verifizieren, wurde hinzugefügt.	19. April 2022
Änderung der Dokumentation	Ersetzen des Begriffes Kundenhauptschlüssel (Customer Master Key, CMK) mit AWS KMS key und KMS-Schlüssel.	30. August 2021

Neue Funktion	Zusätzliche Unterstützung für multiregionale Schlüssel, ein Satz interoperabler KMS-Schlüssel in verschiedenen Regionen, die die gleiche Schlüssel-ID und das gleiche Schlüsselmaterial haben. Sie können multiregionale Schlüssel verwenden, um Daten in einer Region zu verschlüsseln und in einer beliebigen anderen Region zu entschlüsseln.	8. Juni 2021
Neue Funktion	Zusätzliche Unterstützung für attributbasierte Zugriffss teuerung (ABAC). Sie können Tags und Aliase verwenden , um den Zugriff auf Ihre zu kontrollieren. AWS KMS keys	17. Dezember 2020
Neue Funktion	Zusätzliche Unterstützung für VPC-Endpunktrichtlinien.	9. Juli 2020
Neuer Inhalt	Erläutert die Sicherhei tseigenschaften von. AWS KMS	18. Juni 2020
Neue Funktion	Unterstützung für asymmetri sche AWS KMS keys und asymmetrische Datenschl üssel wurde hinzugefügt.	25. November 2019

Aktualisiertes Feature	Sie können die Schlüssel richtlinie von Von AWS verwaltete Schlüssel in der AWS KMS Konsole einsehen. Diese Funktion war früher auf kundenverwaltete Schlüssel beschränkt.	15. November 2019
Neue Funktion	Erläutert, wie Hybrid-Post-Quantum-Schlüsselaustauschalgorithmen in TLS für Ihre Aufrufe an AWS KMS verwendet werden.	4. November 2019
Kontingentänderung	Die Ressourcenkontingente für einige APIs , die KMS-Schlüssel verwalten, wurden erhöht.	18. September 2019
Kontingentänderung	Die Ressourcenkontingente für KMS-Schlüssel, Aliasse und Erteilungen pro KMS-Schlüssel wurden geändert.	27. März 2019
Kontingentänderung	Das gemeinsame pro Sekunde Anforderungskontingent für kryptografische Operationen, die AWS KMS keys in einem benutzerdefinierten Schlüssel speicher verwenden, wurde geändert.	7. März 2019

Neue Funktion	Erläutert, wie AWS KMS benutzerdefinierte Schlüssel speicher erstellt und verwaltet werden. Jeder Schlüssel speicher wird von einem AWS CloudHSM Cluster unterstützt, den Sie besitzen und kontrolli eren.	26. November 2018
New console	Erläutert, wie die neue AWS KMS Konsole verwendet wird, die unabhängig von der IAM-Konsole ist. Die ursprüngl iche Version der Konsole wird zusammen mit der Anleitung zu ihrer Verwendung noch eine kurze Zeit lang zur Verfügung stehen, um Ihnen Zeit zu geben, sich mit der neuen Konsole vertraut zu machen.	7. November 2018
Kontingentänderung	Das Kontingent für gemeinsam e Anfragen zur Verwendung von AWS KMS keys wurde geändert.	21. August 2018
Neuer Inhalt	Erläutert, wie AWS KMS Schlüssel AWS Secrets Manager verwendet werden, um den geheimen Wert in einem Geheimnis zu verschlüs seln.	13. Juli 2018

Neuer Inhalt	Erläutert, wie DynamoDB seine serverseitige AWS KMS AWS KMS keys Verschlüs selungsoption verwendet, um es zu unterstützen.	23. Mai 2018
Neue Funktion	Erläutert, wie Sie einen privaten Endpunkt in Ihrer VPC verwenden, um eine direkte Verbindung herzustel len AWS KMS, anstatt eine Verbindung über das Internet herzustellen.	22. Januar 2018

Frühere Aktualisierungen

In der folgenden Tabelle werden die wichtigen Änderungen am AWS Key Management Service Entwicklerhandbuch vor 2018 beschrieben.

Möglicherweise müssen Sie horizontal oder vertikal scrollen, um alle Daten in dieser Tabelle anzuzeigen.

Änderung	Beschreibung	Datum
Neuer Inhalt	Dokumentation zu Schlagwor te in AWS KMS hinzugefügt.	15. Februar 2017
Neuer Inhalt	Dokumentation zu Überwache n AWS KMS keys und Überwachen Sie KMS-Schlü ssel mit Amazon CloudWatch hinzugefügt.	31. August 2016
Neuer Inhalt	Dokumentation zu Importiertes Schlüsselmaterial hinzugefügt.	11. August 2016

Änderung	Beschreibung	Datum
Neuer Inhalt	Die folgende Dokumentation wurde hinzugefügt: IAM-Richtlinien, Berechtigungsreferenz und Bedingungsschlüssel.	5. Juli 2016
Aktualisierung	Aktualisierung von Teilen der Dokumentation im Kapitel KMS-Schlüsselzugriff und Berechtigungen.	5. Juli 2016
Aktualisierung	Die Seite Kontingente wurde aktualisiert, um die neuen Standardkontingente zu berücksichtigen.	31. Mai 2016
Aktualisierung	Die Seite Kontingente wurde aktualisiert, um die neuen Standardkontingente zu berücksichtigen. Außerdem wurde die Erteilungs-Token-Dokumentation aktualisiert, um sie zu verdeutlichen und genauer zu gestalten.	11. April 2016
Neuer Inhalt	Dokumentation zu Mehreren IAM-Prinzipalen Zugriff auf einen KMS-Schlüssel gewähren und Verwenden der IP-Adressbedingung hinzugefügt.	17. Februar 2016

Änderung	Beschreibung	Datum
Aktualisierung	Aktualisierung der Seiten Die wichtigsten Richtlinien in AWS KMS und Ändern Sie eine wichtige Richtlinie zur Verbesserung der Klarheit und Genauigkeit.	17. Februar 2016
Aktualisierung	Die Themenseiten zum Verwalten von KMS-Schlüsseln wurden aus Gründen der Übersichtlichkeit aktualisiert.	5. Januar 2016
Neuer Inhalt	Dokumentation zu hinzugefügt CloudTrail.	18. November 2015
Neuer Inhalt	Ergänzung von Anweisungen für Ändern Sie eine wichtige Richtlinie.	18. November 2015
Aktualisierung	Die Dokumentation zur Verwendung AWS KMS von Amazon Relational Database Service wurde aktualisiert.	18. November 2015
Neuer Inhalt	Dokumentation über Amazon hinzugefügt WorkSpaces.	6. November 2015
Aktualisierung	Aktualisierung der Seite Die wichtigsten Richtlinien in AWS KMS zur Verbesserung der Klarheit.	22. Oktober 2015

Änderung	Beschreibung	Datum
Neuer Inhalt	Ergänzung von Dokumentation zu Lösche eine AWS KMS key, einschließlich Begleitdokumentation zu Alarm erstellen und Ermitteln Sie die frühere Verwendung eines KMS-Schlüssels.	15. Oktober 2015
Neuer Inhalt	Dokumentation zu Bestimmung des Zugriffs auf AWS KMS keys hinzugefügt.	15. Oktober 2015
Neuer Inhalt	Dokumentation zu Wichtige Zustände von AWS KMS Schlüsseln hinzugefügt.	15. Oktober 2015
Neuer Inhalt	Dokumentation zu Amazon Simple Email Service hinzugefügt.	1. Oktober 2015
Aktualisierung	Die Seite Kontingente wurde aktualisiert, um die neuen Anforderungskontingente zu erläutern.	31. August 2015
Neuer Inhalt	Es wurden Informationen zu den Nutzungsgebühren hinzugefügt AWS KMS. Siehe AWS KMS -Preise	14. August 2015
Neuer Inhalt	Anforderungskontingente wurden dem hinzugefügt AWS KMS Kontingente.	11. Juni 2015

Änderung	Beschreibung	Datum
Neuer Inhalt	Neues Java-Code-Beispiel zur Demonstration der Verwendung der Produktion `UpdateAlias` hinzugefügt.	1. Juni 2015
Aktualisierung	AWS Key Management Service -Regionentabelle zur Allgemeine AWS-Referenz verschoben.	29. Mai 2015
Neuer Inhalt	Dokumentation zu So verwendet Amazon EMR AWS KMS hinzugefügt.	28. Januar 2015
Neuer Inhalt	Dokumentation über Amazon hinzugefügt WorkMail.	28. Januar 2015
Neuer Inhalt	Es wurde eine Dokumentation zur Verwendung AWS KMS von Amazon Relational Database Service hinzugefügt.	6. Januar 2015
Neuer Inhalt	Dokumentation zu Amazon Elastic Transcoder hinzugefügt.	24. November 2014
Neues Handbuch	Einführung des AWS Key Management Service - Entwicklerhandbuchs.	12. November 2014

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.