aws

User Guide

AWS CloudHSM

AWS CloudHSM: User Guide

Table of Contents

Was ist AWS CloudHSM?

AWS CloudHSM kombiniert die Vorteile der AWS-Cloud mit der Sicherheit von Hardwaresicherheitsmodulen (HSMs). Ein Hardwaresicherheitsmodul (HSM) ist eine Hardwarekomponente bzw. Appliance, die kryptografische Operationen durchführt und sicheren Speicher für kryptographische Schlüssel bietet. Mit AWS CloudHSM haben Sie die vollständige Kontrolle über hochverfügbare HSMs in der AWS-Cloud, haben Zugriff mit niedriger Latenz und eine sichere Vertrauensbasis, die das HSM-Management (einschließlich Sicherungen, Bereitstellung, Konfiguration und Wartung) automatisiert.

AWS CloudHSM bietet Kunden eine Vielzahl von Vorteilen:

HSMs sind nach FIPS 140-2 Level-3 validiert

AWS CloudHSM verwendet Standard-HSMs für allgemeine Zwecke, die standardkonform, Einzelmandanten-fähig und nach FIPS 140-2 Level-3 validiert sind. Sie bieten mehr Flexibilität im Vergleich zu vollständig verwalteten AWS-Services, die über vordefinierte Algorithmen und Schlüssellängen für Ihre Anwendung verfügen.

Die E2E-Verschlüsselung ist für AWS nicht sichtbar

Da Ihre Datenebene end-to-end (E2E) verschlüsselt und für AWS nicht sichtbar ist, kontrollieren Sie Ihre eigene Benutzerverwaltung (außerhalb von IAM-Rollen). Der Nachteil dieser Kontrolle besteht darin, dass Sie mehr Verantwortung tragen, als wenn Sie einen verwalteten AWS-Service verwenden würden.

Vollständige Kontrolle über Ihre Schlüssel, Algorithmen und Anwendungsentwicklung

AWS CloudHSM gibt Ihnen die volle Kontrolle über die von Ihnen verwendeten Algorithmen und Schlüssel. Sie können kryptografische Schlüssel (einschließlich Sitzungsschlüssel, Token-Schlüssel, symmetrische Schlüssel und asymmetrische Schlüsselpaare) erzeugen, speichern, importieren, exportieren, verwalten und verwenden. Darüber hinaus geben Ihnen AWS CloudHSM-SDKs die volle Kontrolle über die Anwendungsentwicklung, die Anwendungssprache, das Threading und darüber, wo Ihre Anwendungen physisch existieren.

Migrieren Sie Ihre kryptografischen Workloads in die Cloud

Kunden, die eine Infrastruktur mit öffentlichen Schlüsseln migrieren, die Public Key Cryptography Standards #11 (PKCS #11), Java Cryptographic Extension (JCE), Cryptography API: Next Generation (CNG) oder Key Storage Provider (KSP) verwenden, können mit weniger Änderungen an ihrer Anwendung zu migrieren. AWS CloudHSM

Zugriff auf FIPS- und Nicht-FIPS-Cluster

Weitere Informationen zu den Verwendungsmöglichkeiten des AWS CloudHSM finden Sie in den folgenden Themen. Wenn Sie bereit für erste Schritte mit AWS CloudHSM sind, sehen Sie sich Erste Schritte an.

> **Note**
>
> Wenn Sie einen verwalteten Service für das Erstellen und Kontrollieren der Verschlüsselungsschlüssel möchten, jedoch kein eigenes HSM betreiben wollen, sollten Sie stattdessen AWS Key Management Service in Betracht ziehen.
> Wenn Sie nach einem elastischen Service suchen, der Zahlungs-HSMs und Schlüssel für Zahlungsabwicklungsanwendungen in der Cloud verwaltet, sollten Sie die Verwendung von AWS Payment Cryptography in Betracht ziehen.

Inhalt

Anwendungsfälle für AWS CloudHSM

AWS CloudHSM kann für eine Vielzahl von Zielen eingesetzt werden. Der Inhalt dieses Themas bietet einen Überblick darüber, was Sie mit AWS CloudHSM machen können.

Einhaltung gesetzlicher Vorschriften

Unternehmen, die sich an Unternehmenssicherheitsstandards halten müssen, können mit AWS CloudHSM private Schlüssel zum Schutz hochvertraulicher Daten verwalten. Die von AWS CloudHSM angebotenen HSMs sind FIPS 140-2 Level 3 zertifiziert und entsprechen dem PCI DSS. AWS CloudHSM ist außerdem PCI-PIN-konform und PCI-3DS-konform. Weitere Informationen finden Sie unter -Compliance.

Verschlüsseln und Entschlüsseln von Daten

Wird AWS CloudHSM zur Verwaltung von privaten Schlüsseln zum Schutz hochvertraulicher Daten, zur Verschlüsselung bei der Übertragung und zur Verschlüsselung im Ruhezustand verwendet. Darüber hinaus bietet AWS CloudHSM eine standardkonforme Integration mit mehreren kryptografischen SDKs.

Signieren und verifizieren der Dokumente mit privaten und öffentlichen Schlüsseln

In der Kryptografie ermöglicht die Verwendung eines privaten Schlüssels zum Signieren eines Dokuments den Empfängern, mithilfe eines öffentlichen Schlüssels zu überprüfen, ob Sie (und nicht jemand anderes) das Dokument tatsächlich gesendet haben. Verwenden Sie AWS CloudHSM, um asymmetrische öffentliche und private Schlüsselpaare zu erstellen, die speziell für diesen Zweck entwickelt wurden.

Authentifizieren von Nachrichten mithilfe von HMACs und CMACs

In der Kryptographie werden Cipher-Nachrichtenauthentifizierungscodes (CMACs) und Hash-basierte Nachrichtenauthentifizierungscodes (HMAC) (HMACs) verwendet, um die Integrität von Nachrichten, die über unsichere Netzwerke gesendet werden, zu authentifizieren und zu gewährleisten. Mit AWS CloudHSM können Sie sicher symmetrische Schlüssel erstellen und verwalten, die HMACs und CMACs unterstützen.

Nutzen Sie die Vorteile von AWS CloudHSM und AWS Key Management Service

Kunden können wichtige AWS CloudHSM und AWS KMS kombinieren, um Schlüsselmaterial in einer Einzelmandantenumgebung zu speichern, die nach FIPS 140-2 Level 3 zertifiziert ist, und gleichzeitig die wichtigsten Vorteile von Management, Skalierung und Cloud-Integration von AWS KMS nutzen. Einzelheiten dazu finden Sie im Entwicklerleitfaden zu AWS Key Management Service unter AWS CloudHSMKey Stores.

Auslagerung der SSL/TLS-Verarbeitung für Webserver

Um Daten sicher über das Internet zu senden, verwenden Webserver öffentlich-private Schlüsselpaare und öffentliche SSL/TLS-Schlüsselzertifikate, um HTTPS-Sitzungen einzurichten. Dieser Prozess erfordert eine Menge Rechenleistung für Webserver, aber Sie können den Rechenaufwand reduzieren und gleichzeitig zusätzliche Sicherheit bieten, indem Sie einen Teil

davon auf Ihren AWS CloudHSM-Cluster auslagern. Informationen zum Einrichten der SSL/TLS-Auslagerung mit AWS CloudHSM finden Sie unter SSL/TLS-Auslagerung.

Aktivieren Sie die transparente Datenverschlüsselung (TDE)

Transparent Data Encryption (TDE) wird für die Verschlüsselung von Datenbankdateien eingesetzt. Mit TDE verschlüsselt die Datenbanksoftware die Daten, bevor sie auf der Festplatte gespeichert werden. Sie können mehr Sicherheit erreichen, indem Sie den TDE-Master-Verschlüsselungsschlüssel in HSMs in Ihrem AWS CloudHSM speichern. Informationen zum Einrichten von Oracle TDE mit AWS CloudHSM finden Sie unter Oracle Database-Verschlüsselung.

Verwalten Sie die privaten Schlüssel einer ausstellenden Zertifizierungsstelle (CA)

Eine Zertifizierungsstelle (CA) ist eine vertrauenswürdige Entität, die digitale Zertifikate ausstellt, die einen öffentlichen Schlüssel an eine Identität (eine Person oder Organisation) binden. Um eine CA zu betreiben, müssen Sie das Vertrauen aufrechterhalten, indem Sie den privaten Schlüssel schützen, der die von Ihrer CA ausgestellten Zertifikate signiert. Sie können solche privaten Schlüssel in Ihrem AWS CloudHSM-Cluster speichern und dann Ihre HSMs verwenden, um kryptografische Signaturvorgänge durchzuführen.

Generieren von Zufallszahlen

Die Generierung von Zufallszahlen zur Erstellung von Verschlüsselungsschlüsseln ist ein zentraler Bestandteil der Online-Sicherheit. AWS CloudHSM kann zur sicheren Generierung von Zufallszahlen in HSMs verwendet werden, die Sie kontrollieren und die nur für Sie sichtbar sind.

Funktionsweise von AWS CloudHSM

Dieses Thema bietet einen Überblick über die grundlegenden Konzepte und die Architektur, die Sie verwenden, um Daten sicher zu verschlüsseln und kryptografische Operationen in HSMs durchzuführen. AWS CloudHSM arbeitet in Ihrer eigenen Amazon Virtual Private Cloud (VPC). Bevor Sie AWS CloudHSM verwenden können, erstellen Sie zunächst einen Cluster, fügen ihm HSMs hinzu, erstellen Benutzer und Schlüssel und verwenden dann Client-SDKs, um Ihre HSMs in Ihre Anwendung zu integrieren. Sobald dies erledigt ist, verwenden Sie Client-SDK-Protokolle, AWS CloudTrail, Auditprotokolle und Amazon CloudWatch zur Überwachung von AWS CloudHSM.

Lernen Sie die grundlegenden Konzepte von AWS CloudHSM kennen und wie sie zusammenwirken, um Ihre Daten zu schützen.

Themen

AWS CloudHSM-Clustern

Es kann schwierig sein, einzelne HSMs in einem synchronisierten, redundanten und hochverfügbaren Cluster zusammenarbeiten zu lassen. AWS CloudHSM nimmt Ihnen jedoch die Arbeit ab, da Hardware-Sicherheitsmodule (HSMs) in Clustern bereitgestellt werden. Ein Cluster ist eine Sammlung von einzelnen HSMs, die von AWS CloudHSM synchronisiert werden. Wenn Sie eine Aufgabe oder Operation auf einem HSM in einem Cluster ausführen, werden die anderen HSMs in diesem Cluster automatisch aktualisiert. Um Ihre Ziele in Bezug auf Verfügbarkeit, Beständigkeit und Skalierbarkeit zu erreichen, legen Sie die Anzahl der HSMs in Ihrem Cluster über mehrere Availability Zones hinweg fest.

Sie können einen Cluster mit 1 bis 28 HSMs erstellen (der Standardgrenzwert ist 6 HSMs pro AWS-Konto pro AWS-Region). Sie können die HSMs in verschiedenen Availability Zones in einer AWS-Region. unterbringen. Wenn Sie einem Cluster weitere HSMs hinzufügen, wird der Cluster leistungsfähiger. Wenn Sie Cluster über mehrere Availability Zones hinweg verteilen, bietet dies Redundanz und hohe Verfügbarkeit.

Weitere Informationen zu Clustern finden Sie unter Verwaltung von AWS CloudHSM Clustern.

Informationen zum Erstellen eines Clusters finden Sie unter Erste Schritte.

HSM-Benutzer

Im Gegensatz zu den meisten AWS-Diensten und -Ressourcen verwenden Sie keine AWS Identity and Access Management-(IAM-)Benutzer oder IAM-Richtlinien, um auf Ressourcen innerhalb Ihres

Clusters zuzugreifen. Stattdessen verwenden Sie HSM-Benutzer direkt auf HSMs in Ihrem AWS CloudHSM-Cluster.

HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer mit den richtigen Anmeldeinformationen können HSMs erstellen, indem sie über die AWS-API mit Ressourcen interagieren. Da die E2E-Verschlüsselung für AWS nicht sichtbar ist, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren, da die Anmeldeinformationen direkt auf dem HSM erfolgen. Das HSM authentifiziert jeden HSM-Benutzer anhand von Anmeldeinformationen, die Sie definieren und verwalten. Jeder HSM-Benutzer hat einen Typ, der bestimmt, welche Operationen dieser Benutzer auf dem HSM durchführen kann. Jedes HSM authentifiziert jeden HSM-Benutzer anhand von Anmeldeinformationen, die Sie mit der CloudHSM-CLI definieren.

Wenn Sie die vorherige SDK-Versionsserie verwenden, verwenden Sie CloudHSM Management Utility (CMU).

HSM-Schlüssel

Mit AWS CloudHSM können Sie Ihre Verschlüsselungsschlüssel sicher in Single-Mandanten-HSMs generieren, speichern und verwalten, die sich in Ihrem AWS CloudHSM CloudHSM-Cluster befinden. Schlüssel können symmetrisch oder asymmetrisch sein, können Sitzungsschlüssel (flüchtige Schlüssel) für einzelne Sitzungen und Tokenschlüssel (persistente Schlüssel) für die langfristige Verwendung sein und können aus AWS CloudHSM exportiert und in AWS CloudHSM importiert werden. Schlüssel können auch zur Ausführung gängiger kryptografischer Aufgaben und Funktionen verwendet werden:

- Führen Sie kryptografische Datensignierung und Signaturverifizierung mit symmetrischen und asymmetrischen Verschlüsselungsalgorithmen durch.
- Arbeiten Sie mit Hash-Funktionen, um Message Digests und Hash-basierte Nachrichtenauthentifizierungscodes (HMACs) zu berechnen.
- Wickeln und schützen Sie andere Schlüssel.
- Zugriff auf kryptographisch sichere Zufallsdaten.

Darüber hinaus befolgt AWS CloudHSM einige grundlegende Prinzipien für die Verwendung und Verwaltung von Schlüsseln:

Es stehen viele Schlüsseltypen und Algorithmen zur Auswahl

Damit Sie Ihre eigenen Lösungen anpassen können, bietet AWS CloudHSM viele Schlüsseltypen und Algorithmen zur Auswahl. Algorithmen unterstützen eine Reihe von Schlüsselgrößen. Weitere Informationen finden Sie auf den Seiten zu Attributen und Mechanismen von jedem AWS CloudHSM Kunden-SDKs.

Wie verwalten Sie Schlüssel

AWS CloudHSM-Schlüssel werden über SDKs und Befehlszeilen-Tools verwaltet. Informationen zur Verwendung dieser Tools zur Verwaltung von Schlüsseln finden Sie unter Verwalten von Schlüsseln in AWS CloudHSM und Bewährte Methoden für AWS CloudHSM.

Wem gehören Schlüssel

In AWS CloudHSM befindet sich der Schlüssel im Besitz des Crypto-Benutzers (CU), der diesen erstellt hat. Der Besitzer kann die Befehle key share und key unshare verwenden, um den Schlüssel mit anderen CUs zu teilen oder die gemeinsame Nutzung aufzuheben. Weitere Informationen finden Sie unter Verwenden der CloudHSM-CLI zum Teilen und Aufheben der Freigabe von Schlüsseln.

Zugriff und Nutzung können mit attributebasierter Verschlüsselung gesteuert werden

AWS CloudHSM ermöglicht Ihnen die Verwendung der attributbasierten Verschlüsselung, einer Form der Verschlüsselung, mit der Sie anhand von Schlüsselattributen steuern können, wer Daten auf der Grundlage von Richtlinien entschlüsseln kann.

Client-SDKs

Bei der Verwendung von AWS CloudHSM führen Sie kryptografische Operationen mit AWS CloudHSM Client Software Development Kits (SDKs) durch. AWS CloudHSM Zu den Client-SDKs gehören:

- Public Key Cryptography Standards #11 (PKCS #11)
- JCE-Anbieter
- OpenSSL Dynamic Engine

- Kryptografie-API: CNG (Next Generation) und Key Storage Provider (KSP) für Microsoft Windows

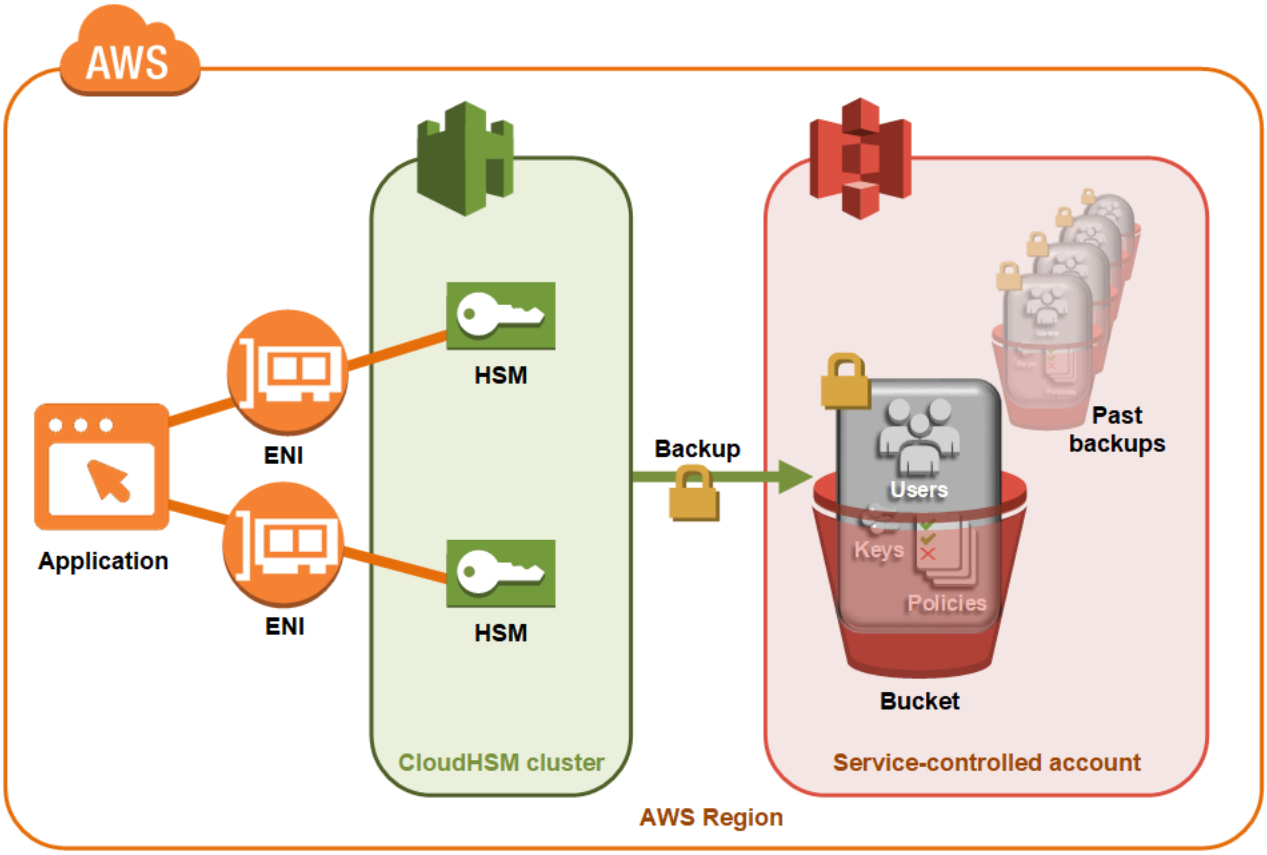
Sie können eines oder alle dieser SDKS in Ihrem AWS CloudHSM-Cluster verwenden. Schreiben Sie Ihren Anwendungscode, um diese SDKs zur Ausführung kryptografischer Operationen in Ihren HSMs zu verwenden.

Hilfsprogramme und Befehlszeilentools werden nicht nur für die Verwendung von SDKs, sondern auch für die Konfiguration der Anmeldeinformationen, Richtlinien und Einstellungen Ihrer Anwendung benötigt. Weitere Informationen finden Sie unter AWS CloudHSM Befehlszeilentools.

Weitere Informationen zur Installation und Verwendung des Client-SDK oder zur Sicherheit der Client-Verbindung finden Sie unter Client-SDKs und nd-to-end E-Verschlüsselung.

Sicherungen von AWS CloudHSM-Clustern

AWS CloudHSM erstellt regelmäßig Sicherungen der Benutzer, Schlüssel und Richtlinien im Cluster. Sicherungen sind sicher, dauerhaft und werden nach einem vorhersehbaren Zeitplan aktualisiert. Die folgende Abbildung zeigt die Beziehung zwischen Ihren Sicherungen und dem Cluster.



Weitere Informationen zur Arbeit mit Sicherungen finden Sie unter Sicherungen verwalten.

Sicherheit

Wenn AWS CloudHSM eine Sicherung des HSM durchführt, verschlüsselt das HSM alle seine Daten, bevor diese an AWS CloudHSM gesendet werden. Die Daten werden zu keinem Zeitpunkt unverschlüsselt aus dem HSM ausgegeben. Außerdem können Sicherungen nicht von AWS entschlüsselt werden, da AWS keinen Zugriff auf den Schlüssel hat, der zum Entschlüsseln der Sicherungen verwendet wurde. Weitere Informationen finden Sie unter Sicherheit von Cluster-Sicherungen.

Haltbarkeit

AWS CloudHSM speichert Sicherungen in einem servicegesteuerten Amazon Simple Storage Service (Amazon S3)-Bucket in derselben Region wie Ihr Cluster. Sicherungen haben eine Beständigkeit von 99,999999999 %, genau wie jedes in Amazon S3 gespeicherte Objekt.

Regionen

Weitere Informationen zu den unterstützten Regionen für AWS CloudHSM finden Sie unter AWS CloudHSM Regions and Endpoints in der Allgemeine AWS-Referenz oder in der Regionentabelle.

AWS CloudHSM ist möglicherweise nicht in allen Availability Zones in einer gegebenen Region verfügbar. Dies sollte sich jedoch nicht auf die Leistung auswirken, da AWS CloudHSM die Last automatisch auf alle HSMs in einem Cluster verteilt.

Wie die meisten AWS-Ressourcen sind Cluster und HSMs regionale Ressourcen. Sie können einen Cluster nicht in mehreren Regionen wiederverwenden oder ihn auf mehrere Regionen erweitern. Sie müssen alle unter Erste Schritte mit AWS CloudHSM aufgelisteten erforderlichen Schritte ausführen, um einen neuen Cluster in einer neuen Region zu erstellen.

AWS CloudHSM ermöglicht es Ihnen, Sicherungen Ihres AWS CloudHSM-Clusters zu Notfallwiederherstellungszwecken von einer Region in eine andere zu kopieren. Weitere Informationen finden Sie unter Sicherungen von AWS CloudHSM-Clustern.

Preisgestaltung

Mit AWS CloudHSM zahlen Sie pro Stunde, ohne langfristige Verpflichtungen oder Vorauszahlungen. Weitere Informationen finden Sie auf der -Website unter AWS CloudHSM-PreiseAWS.

Erste Schritte mit AWS CloudHSM

Die folgenden Themen helfen Ihnen beim Erstellen, Initialisieren und Aktivieren eines AWS CloudHSM-Clusters. Nachdem Sie diese Verfahren abgeschlossen haben, können Sie Benutzer verwalten, Cluster verwalten und die integrierten Software-Bibliotheken zur Durchführung kryptografischer Operationen nutzen.

Inhalt

Erstellen von IAM-Verwaltungsgruppen

Als bewährte Methode wird empfohlen, nicht Ihren Root-Benutzer des AWS-Kontos für die Interaktion mit AWS, einschließlich AWS CloudHSM, zu verwenden. Verwenden Sie stattdessen AWS Identity and Access Management (IAM), um einen IAM-Benutzer, eine IAM-Rolle oder einen Verbundbenutzer zu erstellen. Führen Sie die Schritte im Abschnitt ausErstellen von IAM-Benutzern und -Administratorgruppen, um eine Administratorgruppe zu erstellen und ihr die AdministratorAccess Richtlinie anzufügen. Erstellen Sie dann einen neuen Administratorbenutzer und fügen Sie den Benutzer der Gruppe hinzu. Fügen Sie der Gruppe bei Bedarf weitere Benutzer hinzu. Jeder Benutzer, den Sie hinzufügen, erbt die AdministratorAccess Richtlinie von der Gruppe.

Eine weitere bewährte Methode besteht darin, eine AWS CloudHSM-Administratorgruppe zu erstellen, die nur über die erforderlichen Berechtigungen zum Ausführen von AWS CloudHSM verfügt. Fügen Sie dieser Gruppe bei Bedarf einzelne Benutzer hinzu. Jeder Benutzer übernimmt die eingeschränkten Berechtigungen, die der Gruppe zugeordnet sind, anstatt uneingeschränkten AWS-Zugriff zu erhalten. Der nachfolgende Vom Kunden verwaltete Richtlinien für AWS CloudHSM-Abschnitt enthält die Richtlinie, die Sie Ihrer AWS CloudHSM-Administratorgruppe zuordnen sollten.

AWS CloudHSM definiert eine serviceverknüpfte Rolle für Ihr AWS-Konto. Die serviceverknüpfte Rolle legt derzeit die Berechtigungen fest, die Ihrem Konto die Protokollierung von AWS CloudHSM-Ereignissen gestattet. Die Rolle kann automatisch von AWS CloudHSM oder manuell von Ihnen erstellt werden. Sie können die Rolle zwar nicht bearbeiten, aber Sie können sie löschen. Weitere Informationen finden Sie unter Mit Diensten verknüpfte Rollen für AWS CloudHSM.

Erstellen von IAM-Benutzern und -Administratorgruppen

Beginnen Sie, indem Sie einen IAM-Benutzer sowie eine Administratorgruppe für diesen Benutzer erstellen.

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen eines Administratorbenutzers

Nachdem Sie sich für ein AWS-Kontoangemeldet haben, sichern Sie Ihr Root-Benutzer des AWS-Kontos, aktivieren Sie AWS IAM Identity Centerund erstellen Sie einen administrativen Benutzer, damit Sie nicht den Root-Benutzer für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei der AWS Management Console als Kontobesitzer an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-AnmeldungBenutzerhandbuch zu .
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen dazu finden Sie unter Aktivieren eines virtuellen MFA-Geräts für den Root-Benutzer Ihres AWS-Konto (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Administratorbenutzers

1. Aktivieren von IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Im IAM Identity Center gewähren Sie einem administrativen Benutzer administrativen Zugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit dem standardmäßigen IAM-Identity-Center-Verzeichnis konfigurieren im AWS IAM Identity Center-Benutzerhandbuch.

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM-Identity-Center-Benutzer finden Sie unter Anmelden beim AWS-Zugangsportal im AWS-Anmeldung Benutzerhandbuch zu.

Zu Beispielrichtlinien für AWS CloudHSM, die Sie an eine IAM-Benutzergruppe anfügen können, siehe Identitäts- und Zugriffsmanagement für AWS CloudHSM.

Erstellen einer Virtual Private Cloud (VPC)

Wenn Sie noch keine Virtual Private Cloud (VPC) haben, führen Sie die Schritte in diesem Thema aus, um eine zu erstellen.

> **Note**
>
> Wenn Sie diese Schritte befolgen, werden öffentliche und private Subnetze erstellt.

So erstellen Sie eine VPC

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Verwenden Sie die Regionenauswahl in der Navigationsleiste, um eine der AWS-Regionen auszuwählen, in denen AWS CloudHSM derzeit unterstützt wird.
3. Wählen Sie die Schaltfläche VPC erstellen.
4. Wählen Sie unter Zu erstellende Ressourcen die Option VPC und mehr aus.
5. Geben Sie für Automatische Generierung von Namenstags einen eindeutigen Namen ein, z. B. **`CloudHSM`**.
6. Belassen Sie alle anderen Optionen auf ihren Standardwerten.
7. Wählen Sie VPC erstellen aus.
8. Nachdem die VPC erstellt wurde, wählen Sie VPC anzeigen aus, um die VPC anzuzeigen, die Sie gerade erstellt haben.

Erstellen eines -Clusters

Ein Cluster ist eine Sammlung einzelner HSMs. AWS CloudHSM synchronisiert die HSMs in jedem Cluster, sodass sie als logische Einheit fungieren.

Wenn Sie einen Cluster erstellen, AWS CloudHSM erstellt er in Ihrem Namen eine Sicherheitsgruppe für den Cluster. Diese Sicherheitsgruppe steuert den Netzwerkzugriff auf die HSMs in dem Cluster. Sie erlaubt eingehende Verbindungen nur von Amazon Elastic Compute Cloud (Amazon

EC2) Instances, die sich in der Sicherheitsgruppe befinden. Die Sicherheitsgruppe enthält standardmäßig zunächst keine Instances. Später starten Sie eine Client-Instance und konfigurieren die Sicherheitsgruppe des Clusters,, um die Kommunikation und Verbindungen mit dem HSM zu ermöglichen.

> **Important**
>
> Wenn Sie einen Cluster erstellen, AWS CloudHSM erstellt eine serviceverknüpfte Rolle mit dem Namen AWSServiceRoleForCloudHSM. Wenn die Rolle AWS CloudHSM nicht erstellt werden kann oder die Rolle noch nicht existiert, können Sie möglicherweise keinen Cluster erstellen. Weitere Informationen finden Sie unter Beheben von Cluster-Erstellungsfehlern. Weitere Informationen zu serviceverknüpften Rollen finden Sie unter Mit Diensten verknüpfte Rollen für AWS CloudHSM.

Sie können einen Cluster über die AWS CloudHSM Konsole, die AWS Command Line Interface (CLI) oder die AWS CloudHSM API erstellen.

Einen Cluster erstellen (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Verwenden Sie in der Navigationsleiste die Regionsauswahl, um eine der AWS Regionen auszuwählen, die AWS CloudHSM derzeit unterstützt werden.
3. Wählen Sie Cluster erstellen.
4. Führen Sie im Abschnitt Cluster-Konfiguration Folgendes aus:
   a. Wählen Sie für VPC die VPC, die Sie in Erstellen einer Virtual Private Cloud (VPC) erstellt haben.
   b. Wählen Sie für Availability Zone(s), neben jeder Availability Zone, ein von Ihnen erstelltes privates Subnetz aus.

> **Note**
>
> Auch wenn dies AWS CloudHSM in einer bestimmten Availability Zone nicht unterstützt wird, sollte die Leistung nicht beeinträchtigt werden, da der Lastenausgleich zwischen allen HSMs in einem Cluster AWS CloudHSM automatisch erfolgt. Informationen zur Availability Zone-Unterstützung für finden Sie

unter AWS CloudHSM Regionen und Endpunkte in der Allgemeine AWS-Referenz.
AWS CloudHSM

5. Wählen Sie Weiter aus.
6. Geben Sie an, wie lange der Dienst Sicherungen aufbewahren soll.

> **Note**
>
> Akzeptieren Sie den standardmäßigen Aufbewahrungszeitraum von 90 Tagen oder geben Sie einen neuen Wert zwischen 7 und 379 Tagen ein. Der Dienst löscht automatisch Sicherungen in diesem Cluster, die älter sind als der hier angegebene Wert. Sie können dies später ändern. Weitere Informationen finden Sie unter Konfiguration der Aufbewahrung für Sicherungen.

7. Wählen Sie Weiter.
8. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Tag hinzufügen, wenn Sie mehr als ein Tag zum Cluster hinzufügen möchten.
9. Wählen Sie Überprüfen.
10. Überprüfen Sie Ihre Cluster-Konfiguration und wählen Sie dann Cluster erstellen aus.

So erstellen Sie einen Cluster (CLI)

- Führen Sie über die Eingabeaufforderung den folgenden create-cluster-Befehl aus. Geben Sie den HSM-Instance-Typ, den Aufbewahrungszeitraum für Sicherungen und die Subnetz-IDs der Subnetze an, in denen Sie HSMs erstellen möchten. Verwenden Sie die Subnetz-IDs der von Ihnen erstellten privaten Subnetze. Geben Sie nur ein Subnetz pro Availability Zone an.

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm1.medium \
      --backup-retention-policy Type=DAYS,Value=<number of days> \
      --subnet-ids <subnet ID>

{
    "Cluster": {
        "BackupPolicy": "DEFAULT",
        "BackupRetentionPolicy": {
            "Type": "DAYS",
            "Value": 90
         },
```

```
        "VpcId": "vpc-50ae0636",
        "SubnetMapping": {
            "us-west-2b": "subnet-49a1bc00",
            "us-west-2c": "subnet-6f950334",
            "us-west-2a": "subnet-fd54af9b"
        },
        "SecurityGroup": "sg-6cb2c216",
        "HsmType": "hsm1.medium",
        "Certificates": {},
        "State": "CREATE_IN_PROGRESS",
        "Hsms": [],
        "ClusterId": "cluster-igklspoyj5v",
        "CreateTimestamp": 1502423370.069
    }
}
```

Um einen Cluster (AWS CloudHSM API) zu erstellen

- Senden Sie eine CreateCluster-Anforderung. Geben Sie den HSM-Instance-Typ, die Richtlinie zur Aufbewahrung der Sicherung und die Subnetz-IDs der Subnetze an, in denen Sie HSMs erstellen möchten. Verwenden Sie die Subnetz-IDs der von Ihnen erstellten privaten Subnetze. Geben Sie nur ein Subnetz pro Availability Zone an.

Wenn Ihre Versuche, einen Cluster zu erstellen, fehlschlagen, kann dies auf Probleme mit den servicegebundenen AWS CloudHSM -Rollen zurückzuführen sein. Hilfe beim Beheben des Fehlers finden Sie unter Beheben von Cluster-Erstellungsfehlern.

Überprüfen der Cluster-Sicherheitsgruppe

Wenn Sie einen Cluster erstellen, erstellt AWS CloudHSM eine Sicherheitsgruppe mit dem Namen `cloudhsm-cluster-`*`clusterID`*`-sg`. Diese Sicherheitsgruppe enthält eine vorkonfigurierte TCP-Regel, die eine eingehende und ausgehende Kommunikation innerhalb der Cluster-Sicherheitsgruppe über die Ports 2223-2225 zulässt. Diese SG ermöglicht es Ihren EC2-Instances, Ihre VPC für die Kommunikation mit HSMs in Ihrem Cluster zu verwenden.

⚠ Warning

- Löschen oder ändern Sie nicht die vorkonfigurierte TCP-Regel, die in der Cluster-Sicherheitsgruppe vorhanden ist. Diese Regel kann Verbindungsprobleme und unbefugten Zugriff auf Ihre HSMs verhindern.
- Die Cluster-Sicherheitsgruppe verhindert den unbefugten Zugriff auf Ihre HSMs. Jeder, der Zugriff auf die Instances in der Sicherheitsgruppe hat, hat auch Zugriff auf Ihre HSMs. Die meisten Operationen erfordern, dass sich ein Benutzer am HSM anmeldet. Es ist jedoch möglich, HSMs ohne Authentifizierung nullzusetzen, was das Schlüsselmaterial, die Zertifikate und andere Daten zerstört. Wenn dies geschieht, gehen alle Daten, die seit der letzten Sicherung erstellt oder geändert wurden, verloren und können nicht wiederhergestellt werden. Um den unbefugten Zugriff zu verhindern, stellen Sie sicher, dass nur vertrauenswürdige Administratoren die Instances in der Standardsicherheitsgruppe ändern oder auf sie zugreifen können.

Im nächsten Schritt können Sie eine Amazon-EC2-Instance starten und mit Ihren HSMs verbinden, indem Sie die Cluster-Sicherheitsgruppe anfügen.

Amazon-EC2-Client-Instance starten

Um mit Ihrem AWS CloudHSM Cluster und Ihren HSM-Instances zu interagieren und diese zu verwalten, müssen Sie in der Lage sein, mit den Elastic Network-Schnittstellen Ihrer HSMs zu kommunizieren. Die einfachste Möglichkeit ist die Verwendung einer EC2-Instance in derselben VPC wie Ihr Cluster. Für die Verbindung zu Ihrem Cluster können Sie auch die folgenden AWS -Ressourcen verwenden:

- Amazon-VPC-Peering
- AWS Direct Connect
- VPN-Verbindungen

> **Note**
>
> Dieses Handbuch enthält ein vereinfachtes Beispiel dafür, wie Sie eine EC2-Instance mit Ihrem Cluster verbinden. AWS CloudHSM Bewährte Methoden rund um sichere Netzwerkkonfigurationen finden Sie Sicherer Zugriff auf Ihren Cluster unter.

In der AWS CloudHSM Dokumentation wird normalerweise davon ausgegangen, dass Sie eine EC2-Instance in derselben VPC und Availability Zone (AZ) verwenden, in der Sie Ihren Cluster erstellen.

So erstellen Sie eine EC2-Instance

1. Öffnen Sie das EC2-Dashboard unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie Instance starten aus. Wählen Sie im Drop-down-Menü die Option Instance starten aus.
3. Geben Sie in das Feld Name einen Namen für Ihre EC2-Instance ein.
4. Wählen Sie im Abschnitt Anwendungen und Betriebssystem-Images (Amazon Machine Image) ein Amazon Machine Image (AMI), das einer von CloudHSM unterstützten Plattform entspricht. Weitere Informationen finden Sie unter Client-SDK 5 unterstützte Plattformen.
5. Wählen Sie im Bereich Instance-Typ einen Instance-Typ aus.
6. Verwenden Sie im Abschnitt Schlüsselpaar ein vorhandenes Schlüsselpaar oder wählen Sie Neues Schlüsselpaar erstellen aus und führen Sie die folgenden Schritte aus:
   a. Geben Sie unter Schlüsselpaarname einen Namen für das Schlüsselpaar ein.
   b. Wählen Sie für Schlüsselpaartyp einen Schlüsselpaartyp aus.
   c. Wählen Sie unter Dateiformat des privaten Schlüssels das Dateiformat des privaten Schlüssels.
   d. Wählen Sie Schlüsselpaar erstellen aus.
   e. Laden Sie die private Schlüsseldatei herunter und speichern Sie sie.

> **Important**
>
> Dies ist die einzige Möglichkeit, die private Schlüsseldatei zu speichern. Laden Sie die Datei herunter und speichern Sie sie an einem sicheren Ort. Sie müssen den Namen für Ihr Schlüsselpaar beim Starten einer Instance angeben. Außerdem müssen Sie jedes

Mal, wenn Sie sich mit der Instance verbinden, den entsprechenden privaten Schlüssel angeben und das Schlüsselpaar wählen, das Sie bei der Einrichtung erstellt haben.

7. Wählen Sie in den Netzwerkeinstellungen die Option Bearbeiten aus.
8. Wählen Sie für VPC die zuvor für Ihren Cluster erstellte VPC aus.
9. Wählen Sie für Subnetz das für die VPC erstellte öffentliche Subnetz aus.
10. Wählen Sie für Öffentlche IP automatisch zuweisen Aktivieren aus.
11. Wählen Sie Bestehende Sicherheitsgruppe auswählen aus.
12. Wählen Sie unter Allgemeine Sicherheitsgruppen die Standardsicherheitsgruppe aus dem Drop-down-Menü aus.
13. Verwenden Sie unter Speicher konfigurieren die Drop-down-Menüs, um eine Speicherkonfiguration auszuwählen.
14. Wählen Sie im Übersichtsfenster die Option Instance starten aus.

> **Note**
>
> Wenn Sie diesen Schritt abschließen, wird der Prozess zum Erstellen Ihrer EC2 Instance gestartet.

Weitere Informationen zum Erstellen eines Linux-AmazonEC2-Clients, finden Sie unter Erste Schritte mit Amazon EC2 Linux Instances. Informationen zum Herstellen einer Verbindung mit einem ausgeführten Client finden Sie in den folgenden Themen:

- Herstellen einer Verbindung mit Ihrer Linux-Instance per SSH
- Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY

Das Amazon-EC2-Benutzerhandbuch enthält detaillierte Anweisungen zum Einrichten und Verwenden Ihrer Amazon-EC2-Instances. Die folgende Liste gibt einen Überblick über die verfügbare Dokumentation für Linux- und Windows-Amazon-EC2-Clients:

- Um einen Linux-Amazon-EC2-Client zu erstellen, lesen Sie Erste Schritte mit Amazon-EC2-Linux-Instances.

  Informationen zum Herstellen einer Verbindung mit einem ausgeführten Client finden Sie in den folgenden Themen:

  - Herstellen einer Verbindung mit Ihrer Linux-Instance per SSH
  - Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY
- Um einen Windows-Amazon-EC2-Client zu erstellen, lesen Sie Erste Schritte mit Amazon-EC2-Windows-Instances. Weitere Informationen über das Herstellen einer Verbindung mit dem Windows-Client finden Sie unter Verbinden mit der Windows Instanz.

> **Note**
>
> Ihre EC2-Instance kann alle in diesem Handbuch enthaltenen CLI-Befehle ausführen. Wenn die CLI nicht installiert ist, können Sie sie von herunterladen AWS Command Line Interface. Wenn Sie Windows verwenden, können Sie ein 32- oder 64-Bit-Windows-Installationsprogramm herunterladen. Wenn Sie Linux oder macOS verwenden, können Sie die CLI mit pip installieren.

Konfiguration der Sicherheitsgruppen der Amazon-EC2-Client-Instance

Als Sie eine Amazon-EC2-Instance gestartet haben, haben Sie sie mit einer Standard-Amazon-VPC-Sicherheitsgruppe verknüpft. In diesem Thema wird erläutert, wie Sie die Cluster-Sicherheitsgruppe der EC2-Instance zuordnen. Diese Zuordnung ermöglicht es dem AWS CloudHSM Client, der auf Ihrer EC2-Instance ausgeführt wird, mit Ihren HSMs zu kommunizieren. Um Ihre EC2-Instance mit Ihrem AWS CloudHSM Cluster zu verbinden, müssen Sie die VPC-Standardsicherheitsgruppe ordnungsgemäß konfigurieren und die Cluster-Sicherheitsgruppe der Instance zuordnen.

Ändern der Standardsicherheitsgruppe

Sie müssen die Standardsicherheitsgruppe so ändern, dass diese die SSH- oder RDP-Verbindung zulässt, damit Sie Clientsoftware herunterladen und installieren und mit Ihrem HSM interagieren können.

So ändern Sie die Standardsicherheitsgruppe:

1. Öffnen Sie das EC2-Dashboard unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie Instances (ausgeführt) und aktivieren Sie dann das Kontrollkästchen neben der EC2-Instance, auf der Sie den AWS CloudHSM Client installieren möchten.

3. Wählen Sie auf der Registerkarte Sicherheit die Sicherheitsgruppe namens Standard aus.
4. Wählen Sie oben auf der Seite Aktionen und dann Eingangsregeln bearbeiten aus.
5. Wählen Sie Regel hinzufügen aus.
6. Führen Sie für Typ eine der folgenden Aktionen aus:
   - Für eine Windows Server Amazon-EC2-Instance wählen Sie RDP. Der Port 3389 wird automatisch ausgefüllt.
   - Für eine Linux Amazon-EC2-Instance wählen Sie SSH. Der Portbereich 22 wird automatisch ausgefüllt.
7. Setzen Sie bei einer der beiden Optionen die Quelle auf Meine IP, damit Sie mit Ihrer Amazon-EC2-Instance kommunizieren können.

   > **Important**
   >
   > Geben Sie nicht 0.0.0.0/0 als CIDR-Bereich an, um zu vermeiden, dass jemand auf Ihre Instance zugreifen kann.

8. Wählen Sie Speichern.

Verbinden der Amazon EC2-Instance mit dem AWS CloudHSM Cluster

Sie müssen die Cluster-Sicherheitsgruppe der EC2-Instance zuordnen, damit die EC2-Instance mit den HSMs in Ihrem Cluster kommunizieren kann. Die Cluster-Sicherheitsgruppe enthält eine vorkonfigurierte Regel, die eingehende Kommunikation über die Ports 2223-2225 zulässt.

So verbinden Sie die EC2-Instance mit dem AWS CloudHSM Cluster

1. Öffnen Sie das EC2-Dashboard unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie Instances (ausgeführt) und aktivieren Sie dann das Kontrollkästchen für die EC2-Instance, auf der Sie den AWS CloudHSM Client installieren möchten.
3. Wählen Sie oben auf der Seite Aktionen, Sicherheit und dann Sicherheitsgruppen ändern aus.
4. Wählen Sie die Sicherheitsgruppe mit dem Gruppennamen aus, der Ihrer Cluster-ID entspricht (z. B. `cloudhsm-cluster-`*`clusterID`*`-sg`).
5. Wählen Sie Sicherheitsgruppen hinzufügen aus.
6. Wählen Sie Speichern.

> **Note**
>
> Sie können einer Amazon-EC2-Instance maximal fünf Sicherheitsgruppen zuweisen. Wenn Sie die maximale Grenze erreicht haben, müssen Sie die Standardsicherheitsgruppe der Amazon-EC2-Instance und die Cluster-Sicherheitsgruppe ändern:
>
> Gehen Sie in der Standardsicherheitsgruppe wie folgt vor:
>
> - Fügen Sie eine eingehende Regel hinzu, um den Datenverkehr über das TCP-Protokoll über die Ports 2223-2225 von Cluster-Sicherheitsgruppe zuzulassen.
>
> Gehen Sie in der Cluster-Sicherheitsgruppe wie folgt vor:
>
> - Fügen Sie eine eingehende Regel hinzu, um den Datenverkehr über das TCP-Protokoll über die Ports 2223-2225 der Standardsicherheitsgruppe zuzulassen.

Erstellen eines HSM

Nachdem Sie einen Cluster erstellt haben, können Sie einen HSM erstellen. Ein HSM kann nur dann im Cluster erstellt werden, wenn sich der Cluster im nicht initialisierten Status befindet. Um den Status des Clusters zu ermitteln, rufen Sie die Cluster-Seite in der AWS CloudHSM Konsole auf, verwenden Sie die CLI, um den describe-clusters Befehl auszuführen, oder senden Sie eine DescribeClustersAnfrage in der AWS CloudHSM API. Sie können ein HSM über die AWS CloudHSM Konsole, die CLI oder die AWS CloudHSM API erstellen.

Ein HSM erstellen (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Aktivieren Sie das Kontrollkästchen neben dem Cluster, für den Sie ein HSM erstellen möchten.
3. Wählen Sie Aktionen aus. Wählen Sie aus dem Drop-down-Menü die Option Initialisieren.
4. Wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen.
5. Wählen Sie Erstellen aus.

So erstellen Sie ein HSM (CLI)

- Führen Sie über die Eingabeaufforderung den folgenden create-hsm-Befehl aus. Geben Sie die Cluster-ID des zuvor erstellten Clusters und eine Availability Zone für das HSM an. Geben Sie die Availability Zone in der Form `us-west-2a`, `us-west-2b`, usw. an.

```
$ aws cloudhsmv2 create-hsm --cluster-id <cluster ID> --availability-
zone <Availability Zone>

{
    "Hsm": {
        "HsmId": "hsm-ted36yp5b2x",
        "EniIp": "10.0.1.12",
        "AvailabilityZone": "us-west-2a",
        "ClusterId": "cluster-igklspoyj5v",
        "EniId": "eni-5d7ade72",
        "SubnetId": "subnet-fd54af9b",
        "State": "CREATE_IN_PROGRESS"
    }
}
```

Um ein HSM (AWS CloudHSM API) zu erstellen

- Senden Sie eine CreateHsm-Anforderung. Geben Sie die Cluster-ID des zuvor erstellten Clusters und eine Availability Zone für das HSM an.

Nachdem Sie einen Cluster und ein HSM erstellt haben, können Sie optional die Identität des HSM überprüfen oder direkt mit Initialisieren des Clusters fortfahren.

Überprüfen der Identität und Authentizität des HSM Ihres Clusters (optional)

Zum Initialisieren Ihres Clusters signieren Sie eine Zertifikatssignierungsanforderung (Certificate Signing Request, CSR), die vom ersten HSM des Clusters generiert wurde. Bevor Sie dies tun, sollten Sie die Identität und Authentizität des HSM überprüfen.

> **Note**
>
> Dieses Verfahren ist optional. Es funktioniert jedoch nur solange, bis ein Cluster initialisiert wird. Nachdem der Cluster initialisiert wurde, können Sie diesen Prozess nicht mehr zum Abrufen der Zertifikate oder zum Verifizieren der HSMs einsetzen.

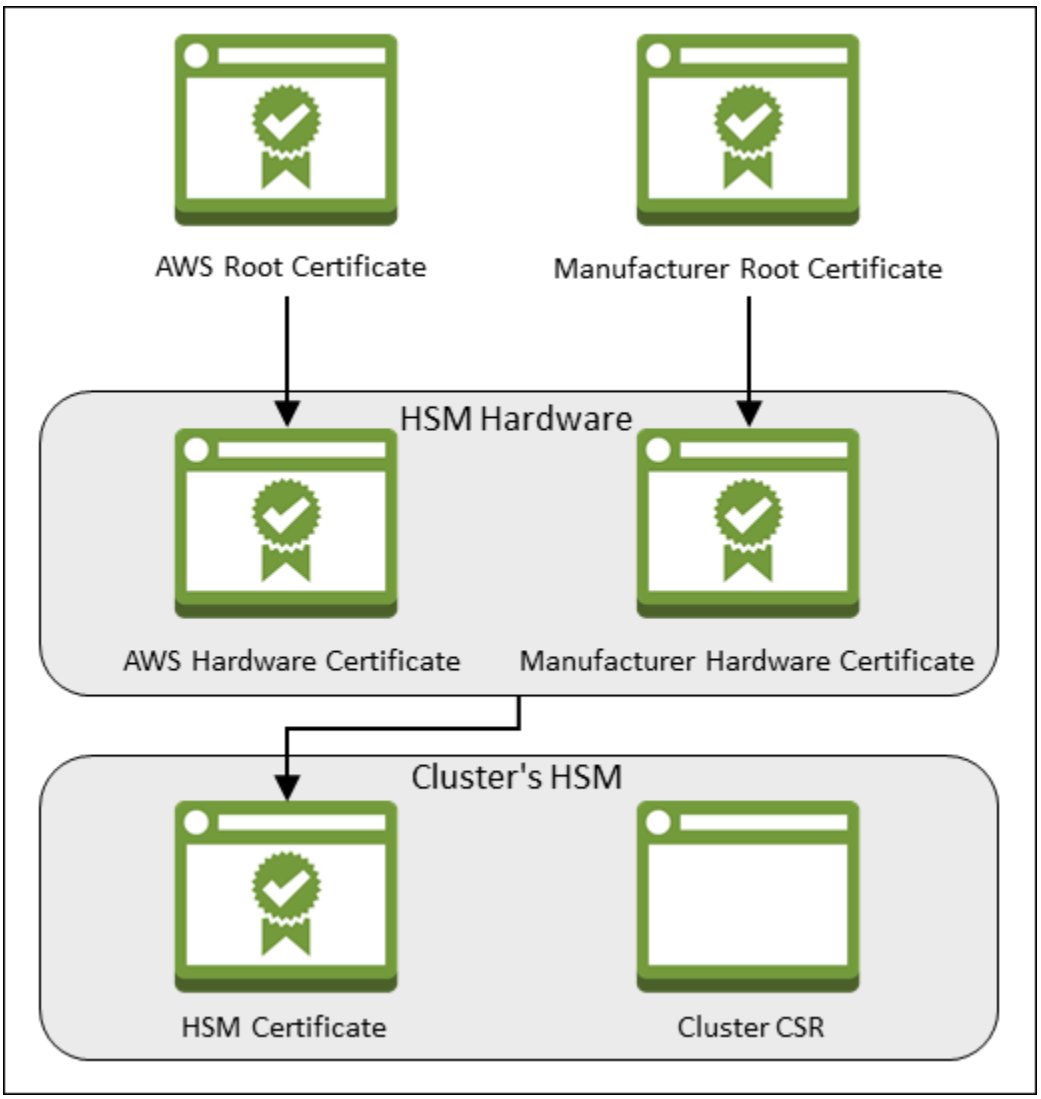
Themen

Übersicht

Um die Identität des ersten HSM Ihres Clusters zu überprüfen, führen Sie die folgenden Schritte aus:

1. Abrufen der Zertifikate und der CSR – In diesem Schritt laden Sie drei Zertifikate und eine CSR aus dem HSM. Sie erhalten außerdem zwei Stammzertifikate, eines vom AWS CloudHSM und eines vom HSM-Hardwarehersteller.
2. Überprüfen Sie die Zertifikatsketten — In diesem Schritt erstellen Sie zwei Zertifikatsketten, eine für das AWS CloudHSM Stammzertifikat und eine für das Stammzertifikat des Herstellers. Anschließend verifizieren Sie das HSM-Zertifikat anhand dieser Zertifikatsketten, um festzustellen, dass AWS CloudHSM sowohl das HSM-Zertifikat als auch der Hardwarehersteller die Identität und Echtheit des HSM bestätigen.
3. Vergleichen öffentlicher Schlüssel – In diesem Schritt extrahieren und vergleichen Sie den öffentlichen Schlüssel im HSM-Zertifikat und die Cluster-CSR, um sicherzustellen, dass sie identisch sind. Dadurch sollten Sie das Vertrauen erhalten, dass die CSR von einem authentischen, vertrauenswürdigen HSM generiert wurde.

Das folgende Diagramm zeigt die CSR, die Zertifikate und ihre Beziehung zueinander. In der folgenden Liste sind alle Zertifikate definiert.



AWS Stammzertifikat

Das ist AWS CloudHSM das Stammzertifikat.

Hersteller-Stammzertifikat

Dies ist das Stammzertifikat des Hardwareherstellers.

AWS Hardware-Zertifikat

AWS CloudHSM hat dieses Zertifikat erstellt, als die HSM-Hardware zur Flotte hinzugefügt wurde. Dieses Zertifikat bestätigt, dass der AWS CloudHSM Eigentümer der Hardware ist.

Hersteller-Hardwarezertifikat

Der HSM-Hardwarehersteller hat dieses Zertifikat erstellt, als er die HSM-Hardware hergestellt hat. Dieses Zertifikat versichert, dass die Hersteller die Hardware erstellt hat.

HSM-Zertifikat

Das HSM-Zertifikat wird mit der FIPS-validierten Hardware erstellt, wenn Sie das erste HSM im Cluster anlegen. Dieses Zertifikat versichert, dass die HSM-Hardware das HSM erstellt hat.

Cluster-CSR

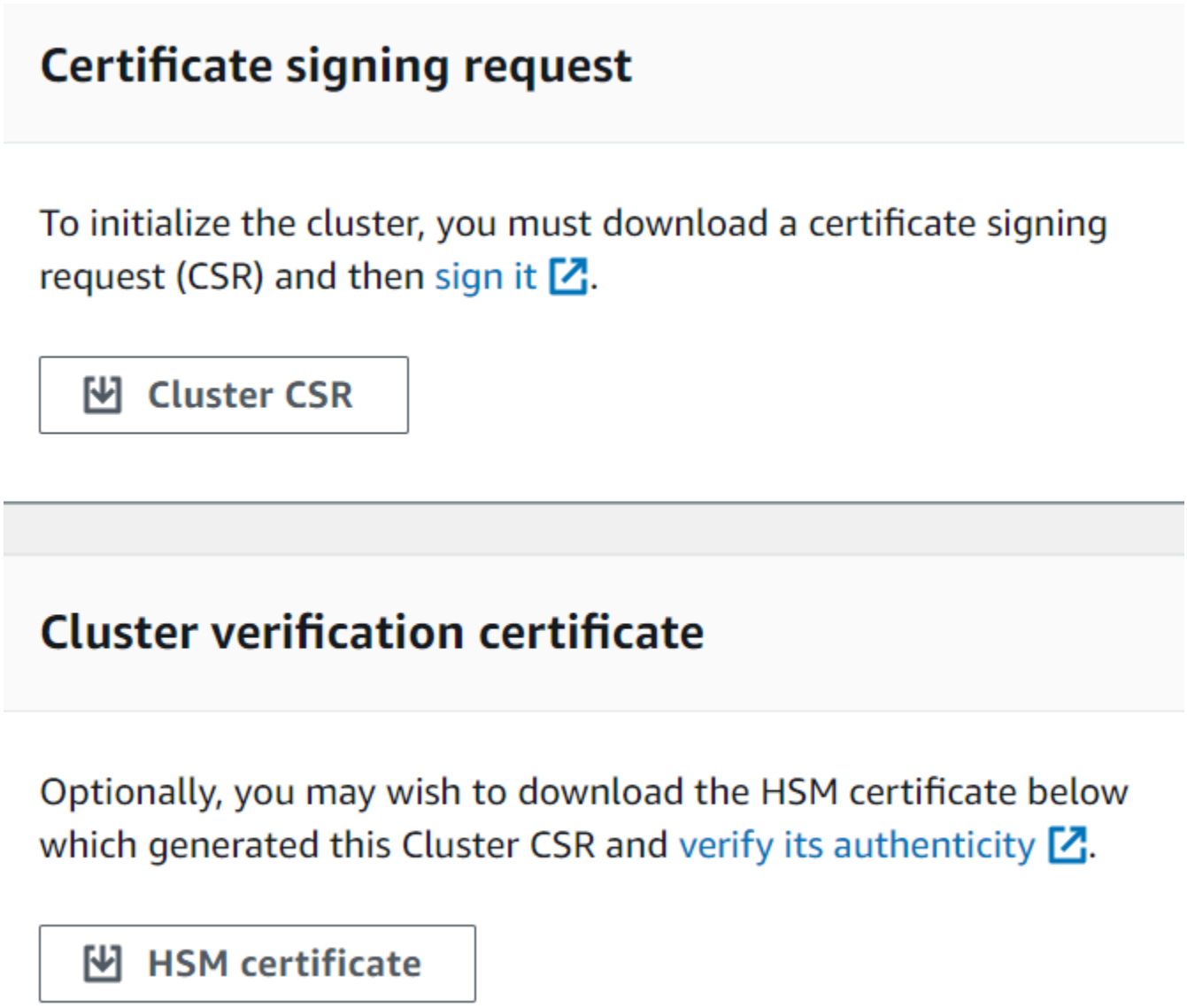
Das erste HSM erstellt die Cluster-CSR. Wenn Sie die Cluster-CSR signieren, beanspruchen Sie den Cluster für sich. Anschließend können Sie die signierte CSR nutzen, um den Cluster zu initialisieren.

Zertifikate aus dem HSM laden

Um die Identität und Authentizität Ihres HSM zu überprüfen, laden Sie zuerst eine CSR und fünf Zertifikate. Sie erhalten drei der Zertifikate vom HSM, was Sie mit der AWS CloudHSM Konsole, der AWS Command Line Interface (CLI) oder der AWS CloudHSM API tun können.

Laden der CSR und der Zertifikate (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie die Optionsschaltfläche neben der Cluster-ID mit dem HSM, das Sie überprüfen möchten.
3. Wählen Sie Aktionen aus. Wählen Sie aus dem Drop-down-Menü die Option Initialisieren.
4. Wenn Sie den vorherigen Schritt zur Erstellung eines HSM nicht abgeschlossen haben, wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen. Wählen Sie dann Erstellen aus.
5. Wenn die Zertifikate und die CSR bereit sind, finden Sie Links, um sie herunterzuladen.



6. Klicken Sie auf die einzelnen Links, um die CSR und Zertifikate herunterzuladen und zu speichern. Zur Vereinfachung der nachfolgenden Schritte speichern Sie alle Dateien in demselben Verzeichnis und nutzen die Standard-Dateinamen.

Um die CSR- und HSM-Zertifikate (CLI) zu erhalten

- Führen Sie über die Eingabeaufforderung den Befehl describe-clusters viermal aus, extrahieren Sie jedes Mal die CSR und die verschiedenen Zertifikate und speichern Sie sie in Dateien.

  a. Geben Sie zum Extrahieren der Cluster-CSR den folgenden Befehl aus. Ersetzen Sie *<cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query 'Clusters[].Certificates.ClusterCsr'
 \
                                   > <cluster ID>_ClusterCsr.csr
```

b. Geben Sie zum Extrahieren des HSM-Zertifikats den folgenden Befehl aus. Ersetzen Sie *<cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query
 'Clusters[].Certificates.HsmCertificate' \
                                   > <cluster ID>_HsmCertificate.crt
```

c. Geben Sie den folgenden Befehl ein, um das AWS Hardwarezertifikat zu extrahieren. Ersetzen Sie *<cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query
 'Clusters[].Certificates.AwsHardwareCertificate' \
                                   > <cluster ID>_AwsHardwareCertificate.crt
```

d. Geben Sie zum Extrahieren des Hardwarezertifikats des Herstellers den folgenden Befehl aus. Ersetzen Sie *<cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query
 'Clusters[].Certificates.ManufacturerHardwareCertificate' \
                                   > <cluster
 ID>_ManufacturerHardwareCertificate.crt
```

Um die CSR- und HSM-Zertifikate (AWS CloudHSM API) zu erhalten

- Senden Sie eine DescribeClusters-Anforderung und extrahieren und speichern dann die CSR und die Zertifikate aus der Antwort.

Die Stammzertifikate laden

Gehen Sie wie folgt vor, um die Stammzertifikate für AWS CloudHSM und den Hersteller zu erhalten. Speichern Sie die Stammzertifikatsdateien in das Verzeichnis, das die CSR und die HSM-Zertifikatsdateien enthält.

Um die Stammzertifikate AWS CloudHSM und die Stammzertifikate des Herstellers zu erhalten

1. Laden Sie das AWS CloudHSM Stammzertifikat herunter: AWS_CloudHSM_Root-G1.zip
2. Laden Sie das Stammzertifikat des Herstellers herunter: liquid_security_certificate.zip

   Um das Zertifikat von der Landingpage https://www.marvell.com/products/security-solutions/liquid-security-hsm-adapters -and-appliances/liquidsecurity-certificate.html herunterzuladen, wählen Sie dann Zertifikat herunterladen.

   Möglicherweise müssen Sie mit der rechten Maustaste auf den Link Download Certificate (Zertifikat herunterladen) klicken und dann Save Link As... (Link speichern unter) wählen, um die Zertifikatdatei zu speichern.
3. Nachdem Sie die Dateien heruntergeladen haben, extrahieren (entpacken) Sie den Inhalt.

Zertifikatketten prüfen

In diesem Schritt erstellen Sie zwei Zertifikatsketten, eine für das AWS CloudHSM Stammzertifikat und eine für das Stammzertifikat des Herstellers. Anschließend verwenden Sie OpenSSL, um das HSM-Zertifikat mit jeder Zertifikatkette zu verifizieren.

Öffnen Sie zum Erstellen der Zertifikatketten eine Linux-Shell. Sie benötigen OpenSSL (in den meisten Linux-Shells verfügbar) und Sie benötigen das Stammzertifikat und die HSM-Zertifikatsdateien, die Sie heruntergeladen haben. Für diesen Schritt benötigen Sie jedoch nicht die CLI, und die Shell muss nicht mit Ihrem AWS Konto verknüpft werden.

Um das HSM-Zertifikat mit dem AWS CloudHSM Stammzertifikat zu verifizieren

1. Navigieren Sie zu dem Verzeichnis, in dem das Stammzertifikat und die HSM-Zertifikatsdateien gespeichert sind, die Sie heruntergeladen haben. Die folgenden Befehle gehen davon aus, dass sich alle Zertifikate im aktuellen Verzeichnis befinden und die Standard-Dateinamen verwendet werden.

Verwenden Sie den folgenden Befehl, um eine Zertifikatskette zu erstellen, die das AWS Hardwarezertifikat und das AWS CloudHSM Stammzertifikat in dieser Reihenfolge umfasst. Ersetzen Sie *<cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ cat <cluster ID>_AwsHardwareCertificate.crt \
      AWS_CloudHSM_Root-G1.crt \
      > <cluster ID>_AWS_chain.crt
```

2. Verwenden Sie den folgenden OpenSSL-Befehl, um das HSM-Zertifikat anhand der AWS -Zertifikatkette zu überprüfen. Ersetzen Sie *<cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ openssl verify -CAfile <cluster ID>_AWS_chain.crt <cluster ID>_HsmCertificate.crt
<cluster ID>_HsmCertificate.crt: OK
```

Das HSM-Zertifikat anhand des Hersteller-Stammzertifikats überprüfen

1. Verwenden Sie den folgenden Befehl zum Erstellen einer Zertifikatkette, die das Hersteller-Hardware-Zertifikat und das Hersteller-Stammzertifikat (in dieser Reihenfolge) enthält. Ersetzen Sie *<cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ cat <cluster ID>_ManufacturerHardwareCertificate.crt \
      liquid_security_certificate.crt \
      > <cluster ID>_manufacturer_chain.crt
```

2. Verwenden Sie den folgenden Befehl, um das HSM-Zertifikat anhand der Hersteller-Zertifikatkette zu überprüfen. Ersetzen Sie *<cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ openssl verify -CAfile <cluster ID>_manufacturer_chain.crt <cluster
 ID>_HsmCertificate.crt
<cluster ID>_HsmCertificate.crt: OK
```

Extrahieren und Vergleichen der öffentlichen Schlüssel

Verwenden Sie OpenSSL, um die öffentlichen Schlüssel im HSM-Zertifikat und die Cluster-CSR zu extrahieren und zu vergleichen, um sicherzustellen, dass sie identisch sind.

Nutzen Sie zum Vergleichen der öffentlichen Schlüssel die Linux-Shell. Sie benötigen OpenSSL, das in den meisten Linux-Shells verfügbar ist, aber Sie benötigen die CLI für diesen Schritt nicht. Die Shell muss nicht mit Ihrem AWS Konto verknüpft sein.

Extrahieren und Vergleichen der öffentlichen Schlüssel

1. Um den öffentlichen Schlüssel aus dem HSM-Zertifikat zu extrahieren, führen Sie den folgenden Befehl aus.

   ```
   $ openssl x509 -in <cluster ID>_HsmCertificate.crt -pubkey -noout > <cluster
    ID>_HsmCertificate.pub
   ```

2. Um den öffentlichen Schlüssel aus der Cluster-CSR zu extrahieren, führen Sie den folgenden Befehl aus.

   ```
   $ openssl req -in <cluster ID>_ClusterCsr.csr -pubkey -noout > <cluster
    ID>_ClusterCsr.pub
   ```

3. Verwenden Sie den folgenden Befehl, um die öffentlichen Schlüssel zu vergleichen. Wenn die öffentlichen Schlüssel identisch sind, erzeugt der folgende Befehl keine Ausgabe.

   ```
   $ diff <cluster ID>_HsmCertificate.pub <cluster ID>_ClusterCsr.pub
   ```

Nach dem Überprüfen der Identität und Authentizität des HSM fahren Sie mit fort Initialisieren des Clusters.

Initialisieren des Clusters

Führen Sie die Schritte in den folgenden Themen aus, um Ihren AWS CloudHSM Cluster zu initialisieren.

> **Note**
>
> Bevor Sie den Cluster initialisieren, sehen Sie sich den Prozess an, mit dem Sie die Identität und Authentizität der HSMs verifizieren können. Dieser Prozess ist optional und funktioniert nur, bis ein Cluster initialisiert wird. Nachdem der Cluster initialisiert wurde, können Sie mit diesem Prozess keine Zertifikate mehr abrufen oder HSMs überprüfen.

Themen

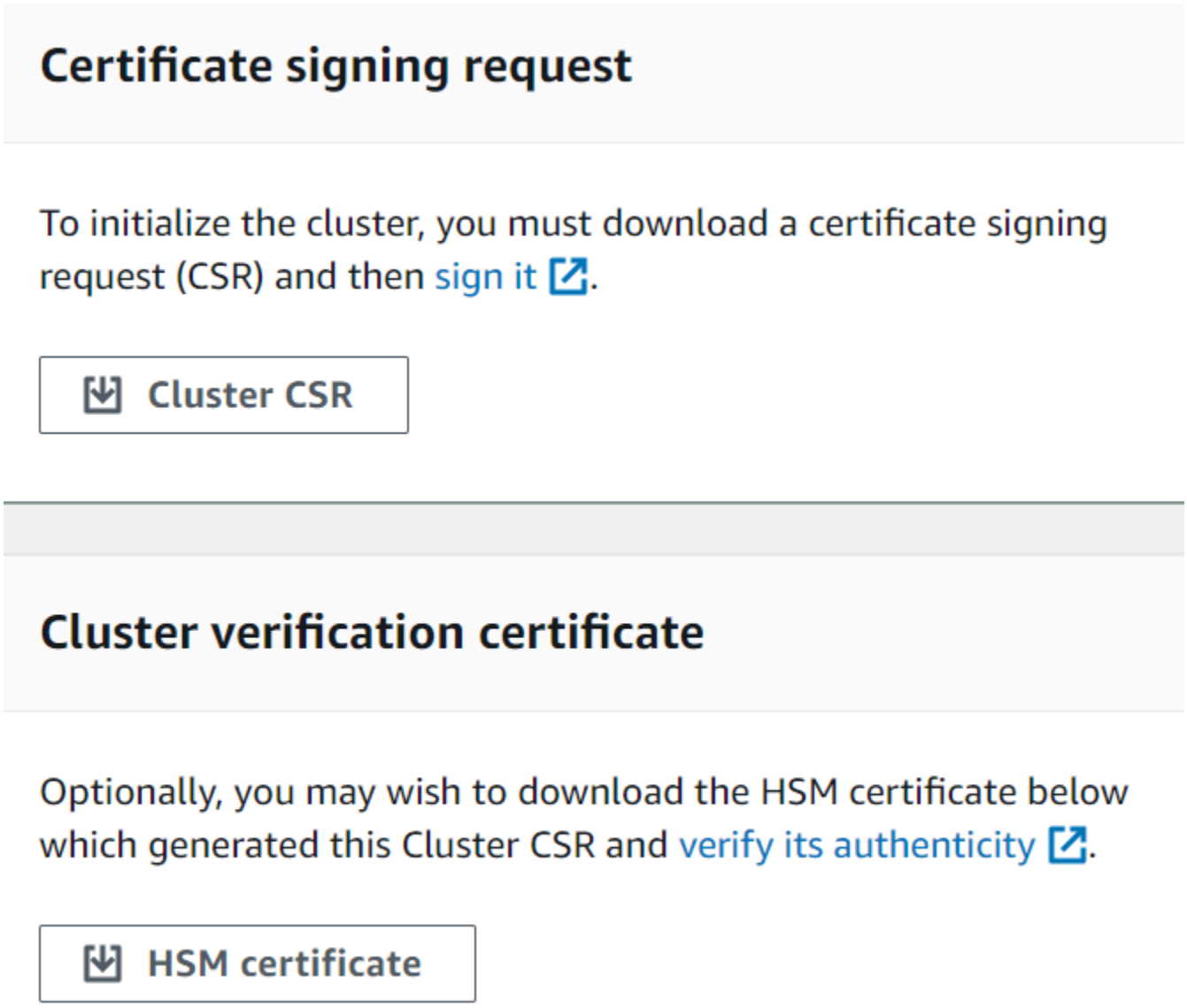
Anfordern der Cluster-CSR

Bevor Sie den Cluster initialisieren, müssen Sie eine Zertifikatsignieranforderung (CSR, Certificate Signing Request) herunterladen und signieren, die vom ersten HSM des Clusters erstellt wurde. Wenn Sie die Schritte ausgeführt haben, um die Identität des HSM Ihres Clusters zu überprüfen, befinden Sie sich bereits im Besitz der CSR und können sie signieren. Andernfalls rufen Sie die CSR jetzt mithilfe der AWS CloudHSM Konsole, der AWS Command Line Interface (CLI) oder der AWS CloudHSM API ab.

> **Important**
>
> Um Ihren Cluster zu initialisieren, muss Ihr Trust Anchor RFC 5280 entsprechen und die folgenden Anforderungen erfüllen:
>
> - Wenn Sie X509v3-Erweiterungen verwenden, muss die X509v3-Erweiterung Basic Constraints vorhanden sein.
> - Der Trust Anchor muss ein selbstsigniertes Zertifikat sein.
> - Erweiterungswerte dürfen nicht miteinander in Konflikt stehen.

Die CSR laden (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie die Optionsschaltfläche neben der Cluster-ID mit dem HSM, das Sie überprüfen möchten.
3. Wählen Sie Aktionen aus. Wählen Sie aus dem Drop-down-Menü die Option Initialisieren.
4. Wenn Sie den vorherigen Schritt zur Erstellung eines HSM nicht abgeschlossen haben, wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen. Wählen Sie dann Erstellen aus.
5. Wenn der CSR bereit ist, sehen Sie einen Link, über den Sie sie herunterladen können.



6. Wählen Sie Cluster-CSR, um die CSR herunterzuladen und zu speichern.

Um die CSR (CLI) zu erhalten

- Führen Sie an der Eingabeaufforderung den folgenden describe-clusters-Befehl aus, der die CSR extrahiert und in einer Datei speichert. Ersetzen Sie die *cluster ID>* durch die ID des Clusters, das Sie zuvor erstellt haben.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query 'Clusters[].Certificates.ClusterCsr' \
                                   > <cluster ID>_ClusterCsr.csr
```

Um die CSR (AWS CloudHSM API) zu erhalten

1. Senden Sie eine DescribeClusters-Anforderung.
2. Extrahieren und speichern Sie die CSR aus der Antwort.

Signieren der CSR

Derzeit müssen Sie ein selbstsigniertes Signaturzertifikat erstellen und die CSR für den Cluster damit signieren. Sie benötigen die CLI für diesen Schritt nicht, und die Shell muss nicht mit Ihrem AWS Konto verknüpft werden. Führen Sie zum Signieren der CSR die folgenden Schritte durch:

1. Füllen Sie den vorherigen Abschnitt aus (siehe Anfordern der Cluster-CSR).
2. Erstellen eines privaten Schlüssels
3. Erstellen eines Signaturzertifikats mit dem privaten Schlüssel
4. Signieren Sie Ihre CSR.

Erstellen eines privaten Schlüssels

> **Note**
>
> Für einen Produktionscluster sollte der Schlüssel, den Sie erstellen wollen, auf sichere Weise mit einer vertrauenswürdigen Zufallsquelle erstellt werden. Wir empfehlen, dass Sie ein sicheres standortexternes und Offline-HSM oder etwas Äquivalentes verwenden. Speichern Sie den Schlüssel sicher. Der Schlüssel legt die Identität des Clusters und Ihre alleinige Kontrolle über die darin enthaltenen HSMs fest.
> Für die Entwicklung und für Tests können Sie ein beliebiges Tool (z. B. OpenSSL) verwenden, um das Cluster-Zertifikat zu erstellen und zu signieren. Im folgenden Beispiel wird gezeigt, wie Sie einen Schlüssel erstellen. Nachdem Sie den Schlüssel verwendet haben, um ein selbstsigniertes Zertifikat zu erstellen (siehe unten), sollten Sie ihn sicher speichern. Um sich bei Ihrer AWS CloudHSM Instance anzumelden, muss das Zertifikat vorhanden sein, der private Schlüssel jedoch nicht.

Verwenden Sie den folgenden Befehl, um einen privaten Schlüssel zu erstellen. Bei der Initialisierung eines AWS CloudHSM Clusters müssen Sie das RSA 2048-Zertifikat oder das RSA 4096-Zertifikat verwenden.

```
$ openssl genrsa -aes256 -out customerCA.key 2048
Generating RSA private key, 2048 bit long modulus
........+++
............+++
e is 65537 (0x10001)
Enter pass phrase for customerCA.key:
Verifying - Enter pass phrase for customerCA.key:
```

Verwenden des privaten Schlüssels zum Erstellen eines selbstsignierten Zertifikats

Die vertrauenswürdige Hardware, die Sie verwenden, um den privaten Schlüssel für Ihre Produktions-Cluster zu erstellen, sollte auch ein Software-Tool bereitstellen, um unter Verwendung dieses Schlüssels ein selbstsigniertes Zertifikat zu generieren. Das folgende Beispiel verwenden OpenSSL und den privaten Schlüssel, den Sie im vorherigen Schritt erstellt haben, um ein Signaturzertifikat zu erstellen. Das Zertifikat gilt für 10 Jahre (3652 Tage). Lesen Sie die Anweisungen auf dem Bildschirm und befolgen Sie die Eingabeaufforderungen.

```
$ openssl req -new -x509 -days 3652 -key customerCA.key -out customerCA.crt
Enter pass phrase for customerCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

Mit diesem Befehl wird ein Zertifikat mit dem Namen `customerCA.crt` erstellt. Legen Sie dieses Zertifikat auf jedem Host ab, von dem aus Sie eine Verbindung zu Ihrem Cluster herstellen möchten. AWS CloudHSM Wenn Sie der Datei einen anderen Namen geben oder unter einem anderen Pfad als dem Root-Verzeichnis speichern, müssen Sie Ihre Client-Konfigurationsdatei entsprechend anpassen. Verwenden Sie das Zertifikat und den privaten Schlüssel, die Sie gerade erstellt haben,

um die Cluster-Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) im nächsten Schritt zu signieren.

Die Cluster-CSR signieren

Die vertrauenswürdige Hardware, die Sie verwenden, um Ihren privaten Schlüssel für Ihre Produktions-Cluster zu erstellen, sollte auch ein Tool bereitstellen, um die CSR mit diesem Schlüssel zu signieren. In dem folgenden Beispiel wird OpenSSL zum Signieren der Cluster-CSR verwendet. Das Beispiel verwendet Ihren privaten Schlüssel und das selbstsignierte Zertifikat, das Sie im vorherigen Schritt erstellt haben.

```
$ openssl x509 -req -days 3652 -in <cluster ID>_ClusterCsr.csr \
                               -CA customerCA.crt \
                               -CAkey customerCA.key \
                               -CAcreateserial \
                               -out <cluster ID>_CustomerHsmCertificate.crt
Signature ok
subject=/C=US/ST=CA/O=Cavium/OU=N3FIPS/L=SanJose/CN=HSM:<HSM
 identifer>:PARTN:<partition number>, for FIPS mode
Getting CA Private Key
Enter pass phrase for customerCA.key:
```

Mit diesem Befehl wird eine Datei mit dem Namen *<cluster ID>*`_CustomerHsmCertificate.crt` erstellt. Verwenden Sie diese Datei als signiertes Zertifikat, wenn Sie den Cluster initialisieren.

Initialisieren des Clusters

Verwenden Sie das signierte HSM-Zertifikat und Ihr Signaturzertifikat, um den Cluster zu initialisieren. Sie können die AWS CloudHSM Konsole, die CLI oder die AWS CloudHSM API verwenden.

Initialisieren eines Clusters (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie die Optionsschaltfläche neben der Cluster-ID mit dem HSM, das Sie überprüfen möchten.
3. Wählen Sie Aktionen aus. Wählen Sie aus dem Drop-down-Menü die Option Initialisieren.
4. Wenn Sie den vorherigen Schritt zur Erstellung eines HSM nicht abgeschlossen haben, wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen. Wählen Sie dann Erstellen aus.

5. Wählen Sie auf der Seite Herunterladen der Zertifikat-Signierungsanforderung Weiter aus. Wenn die Schaltfläche Weiter nicht verfügbar ist, wählen Sie zuerst einen der CSR- oder Zertifikat-Links. Wählen Sie anschließend Weiter.
6. Wählen Sie auf der Seite Zertifikatsignierungsanforderung (CSR) signieren Weiter.
7. Gehen Sie auf der Seite Die Zertifikate hochladen wie folgt vor:

   a. Wählen Sie neben Cluster-Zertifikat Datei hochladen. Suchen Sie anschließend das HSM-Zertifikat, das Sie zuvor signiert haben, und wählen Sie es aus. Wenn Sie die Schritte im vorigen Abschnitt ausgeführt haben, wählen Sie die Datei *<cluster ID>*_CustomerHsmCertificate.crt.

   b. Wählen Sie neben Zertifikat ausstellen Datei hochladen. Wählen Sie anschließend das Signaturzertifikat aus. Wenn Sie die Schritte im vorigen Abschnitt ausgeführt haben, wählen Sie die Datei customerCA.crt.

   c. Wählen Sie Hochladen und initialisieren.

So initialisieren Sie einen Cluster (CLI)

- Führen Sie über die Eingabeaufforderung den folgenden initialize-cluster-Befehl aus. Geben Sie Folgendes an:

  - Die ID des Clusters, den Sie zuvor erstellt haben.
  - Das HSM-Zertifikat, das Sie zuvor signiert haben. Wenn Sie die Schritte im vorigen Abschnitt ausgeführt haben, wurde es in der Datei *<cluster ID>*_CustomerHsmCertificate.crt gespeichert.
  - Ihr Signaturzertifikat. Wenn Sie die Schritte im vorigen Abschnitt ausgeführt haben, wird das Signaturzertifikat in der Datei customerCA.crt gespeichert.

```
$ aws cloudhsmv2 initialize-cluster --cluster-id <cluster ID> \
                                    --signed-cert file://<cluster
 ID>_CustomerHsmCertificate.crt \
                                    --trust-anchor file://customerCA.crt
{
    "State": "INITIALIZE_IN_PROGRESS",
    "StateMessage": "Cluster is initializing. State will change to INITIALIZED upon
 completion."
}
```

Um einen Cluster (AWS CloudHSM API) zu initialisieren

- Senden Sie eine InitializeCluster-Anforderung mit folgendem Inhalt:
  - Die ID des Clusters, den Sie zuvor erstellt haben.
  - Das HSM-Zertifikat, das Sie zuvor signiert haben.
  - Ihr Signaturzertifikat.

CloudHSM-CLI installieren und konfigurieren

Um mit dem HSM in Ihrem AWS CloudHSM Cluster zu interagieren, benötigen Sie die CloudHSM-CLI.

Aufgaben

- Installieren Sie die Befehlszeilentools AWS CloudHSM

Installieren Sie die Befehlszeilentools AWS CloudHSM

Connect zu Ihrer Client-Instance her und führen Sie die folgenden Befehle aus, um die AWS CloudHSM Befehlszeilentools herunterzuladen und zu installieren. Weitere Informationen finden Sie unter Amazon-EC2-Client-Instance starten.

Verwenden Sie die folgenden Befehle, um die CloudHSM-CLI herunterzuladen und zu installieren.

Amazon Linux 2

Amazon Linux 2 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

Amazon Linux 2 auf ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.aarch64.rpm
```

Amazon Linux 2023

Amazon Linux 2023 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

Amazon Linux 2023 auf ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

CentOS 7 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-
latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

RHEL 7 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-
latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

RHEL 8 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

RHEL 9 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.x86_64.rpm
```

RHEL 9 auf ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

Ubuntu 20.04 LTS auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-cli_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_amd64.deb
```

Ubuntu 22.04 LTS auf ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-
cli_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_arm64.deb
```

Windows Server 2016

Öffnen Sie Windows Server 2016 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

Öffnen Sie Windows Server 2019 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

Verwenden Sie die folgenden Befehle, um CloudHSM-CLI zu konfigurieren.

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IP addresses of the HSMs>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IP addresses
 of the HSMs>
```

Aktivieren des Clusters

Wenn Sie einen AWS CloudHSM-Cluster aktivieren, ändert sich der Status des Clusters von initialisiert in aktiv. Sie können dann die Benutzer des Hardware-Sicherheitsmoduls (HSM) verwalten und das HSM verwenden.

Important

Bevor Sie den Cluster aktivieren können, müssen Sie zunächst das ausstellende Zertifikat an den Standardspeicherort für die Plattform auf jeder EC2-Instance kopieren, die eine Verbindung zum Cluster herstellt (Sie erstellen das ausstellende Zertifikat, wenn Sie den Cluster initialisieren).

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

Nachdem Sie das ausstellende Zertifikat platziert haben, installieren Sie die CloudHSM-CLI und führen Sie den cluster activate-Befehl auf Ihrem ersten HSM aus. Sie werden feststellen, dass das Administratorkonto auf dem ersten HSM in Ihrem Cluster die Rolle unactivated-admin hat. Dies ist

eine temporäre Rolle, die nur vor der Clusteraktivierung existiert. Wenn Sie Ihren Cluster aktivieren, ändert sich die Rolle „unactivated-admin“ in „admin“.

Aktivieren eines Clusters

1. Stellen Sie eine Verbindung zu der Client-Instance her, die Sie zuvor gestartet haben. Weitere Informationen finden Sie unter Amazon-EC2-Client-Instance starten. Sie können eine Linux-Instance oder Windows Server starten.
2. Führen Sie die CloudHSM-CLI im interaktiven Modus aus.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

3. (Optional) Verwenden Sie den Befehl user list, um die vorhandenen Benutzer anzuzeigen.

   ```
   aws-cloudhsm >  user list
   {
     "error_code": 0,
     "data": {
       "users": [
         {
           "username": "admin",
           "role": "unactivated-admin",
           "locked": "false",
           "mfa": [],
           "cluster-coverage": "full"
         },
         {
           "username": "app_user",
           "role": "internal(APPLIANCE_USER)",
           "locked": "false",
           "mfa": [],
           "cluster-coverage": "full"
         }
       ]
     }
   }
   ```

4. Verwenden Sie den cluster activate-Befehl, um das anfängliche Admin-Passwort festzulegen.

```
aws-cloudhsm >  cluster activate

 Enter password:<NewPassword>
 Confirm password:<NewPassword>

{
  "error_code": 0,
  "data": "Cluster activation successful"
}
```

Wir empfehlen, das neue Passwort auf einem Passwort-Arbeitsblatt zu notieren. Verlieren Sie das Arbeitsblatt nicht. Wir empfehlen, eine Kopie des Passwort-Arbeitsblatts auszudrucken, Ihre wichtigen HSM-Passwörter darauf zu notieren und die Kopie anschließend an einem sicheren Ort abzulegen. Wir empfehlen, mindestens eine Kopie dieses Arbeitsblatts an einem sicheren Ort außerhalb des Standorts aufzubewahren.

5. (Optional) Verwenden Sie den Befehl user list, um sicherzustellen, dass sich der Typ des Benutzers in Admin/CO geändert hat.

```
aws-cloudhsm >  user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
       {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
```

```
      }
    ]
  }
}
```

6. Verwenden Sie den quit-Befehl, um das CloudHSM-CLI-Tool zu stoppen.

```
aws-cloudhsm > quit
```

Weitere Informationen zur Arbeit mit der CloudHSM-CLI oder der CMU finden Sie unter Grundlegendes zu HSM-Benutzern und Grundlegendes zur HSM-Benutzerverwaltung mit CMU.

Umkonfigurieren von SSL mit einem neuen Zertifikat und privaten Schlüssel (optional)

AWS CloudHSM verwendet ein SSL-Zertifikat zum Herstellen einer Verbindung mit einem HSM. Ein Standardschlüssel und ein SSL-Zertifikat sind enthalten, wenn Sie den Client installieren. Sie können aber eigene Schlüssel und Zertifikate erstellen und verwenden. Beachten Sie, dass Sie das selbstsignierte Zertifikat (*customerCA.crt*), benötigen, das Sie beim Initialisieren des Clusters erstellt haben.

Allgemein gesehen ist dies ein zweistufiger Prozess:

1. Zunächst erstellen Sie einen privaten Schlüssel und verwenden dann diesen Schlüssel, um eine Zertifikatssignierungsanforderung (CSR) zu erstellen. Verwenden Sie das ausstellende Zertifikat, das Zertifikat, das Sie bei der Initialisierung des Clusters erstellt haben, um die CSR zu signieren.
2. Als Nächstes verwenden Sie das Configure-Tool, um den Schlüssel und das Zertifikat in die entsprechenden Verzeichnisse zu kopieren.

Erstellen Sie einen Schlüssel, eine CSR, und signieren Sie dann die CSR

Die Schritte sind für Client-SDK 3 oder Client-SDK 5 identisch.

So konfigurieren Sie SSL mit einem neuen Zertifikat und privaten Schlüssel um

1. Verwenden Sie den folgenden OpenSSL-Befehl, um einen privaten Schlüssel zu erstellen:

```
openssl genrsa -out ssl-client.key 2048
Generating RSA private key, 2048 bit long modulus
........+++
............+++
e is 65537 (0x10001)
```

2. Verwenden Sie den folgenden OpenSSL-Befehl zum Erstellen einer Zertifikatssignierungsanforderung (Certificate Signing Request, CSR). Sie werden gebeten, eine Reihe von Fragen zum Zertifikat zu beantworten.

```
openssl req -new -sha256 -key ssl-client.key -out ssl-client.csr
Enter pass phrase for ssl-client.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Signieren Sie die CSR mit dem Zertifikat *`customerCA.crt`*, das Sie beim Initialisieren des Clusters erstellt haben.

```
openssl x509 -req -days 3652 -in ssl-client.csr \
        -CA customerCA.crt \
        -CAkey customerCA.key \
        -CAcreateserial \
        -out ssl-client.crt
```

```
Signature ok
subject=/C=US/ST=WA/L=Seattle/O=Example Company/OU=sales
Getting CA Private Key
```

Aktivieren Sie benutzerdefiniertes SSL für AWS CloudHSM

Die Schritte sind für Client-SDK 3 oder Client-SDK 5 unterschiedlich. Weitere Informationen über die Arbeit mit dem Befehlszeilentool Configure finden Sie unter ???.

Themen

- Benutzerdefiniertes SSL für Client-SDK 3
- Benutzerdefiniertes SSL für Client-SDK 5

Benutzerdefiniertes SSL für Client-SDK 3

Verwenden Sie das Configure-Tool für Client-SDK 3, um benutzerdefiniertes SSL zu aktivieren. Weitere Informationen zum Configure-Tool für das Client-SDK 3 finden Sie unter ???.

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 3 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
sudo /opt/cloudhsm/bin/configure --ssl \
 --pkey /opt/cloudhsm/etc/ssl-client.key \
 --cert /opt/cloudhsm/etc/ssl-client.crt
```

3. Fügen Sie das Zertifikat `customerCA.crt` dem Vertrauensspeicher hinzu. Erstellen Sie einen Hash des Zertifikat-Themennamens. Dadurch wird ein Index erstellt, in dem das Zertifikat über den Namen gesucht werden kann.

```
openssl x509 -in /opt/cloudhsm/etc/customerCA.crt -hash | head -n 1
```

```
1234abcd
```

Erstellen Sie ein Verzeichnis.

```
mkdir /opt/cloudhsm/etc/certs
```

Erstellen Sie eine Datei, die das Zertifikat mit dem Hashnamen enthält.

```
sudo cp /opt/cloudhsm/etc/customerCA.crt /opt/cloudhsm/etc/certs/1234abcd.0
```

Benutzerdefiniertes SSL für Client-SDK 5

Verwenden Sie eines der Configure-Tools des Client-SDK 5, um benutzerdefiniertes SSL zu aktivieren. Weitere Informationen zum Configure-Tool für das Client-SDK 5 finden Sie unter ???.

PKCS #11 library

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
```

```
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. Verwenden Sie mit einem PowerShell-Interpreter das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

OpenSSL Dynamic Engine

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

JCE provider

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. Verwenden Sie mit einem PowerShell-Interpreter das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

CloudHSM CLI

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
```

```
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. Verwenden Sie mit einem PowerShell-Interpreter das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

Erstellen einer Anwendung

Erstellen Sie Anwendungen und arbeiten Sie mit Schlüsseln mithilfe von AWS CloudHSM.

Um mit der Erstellung und Verwendung von Schlüsseln in Ihrem neuen Cluster zu beginnen, müssen Sie zunächst einen HSM-Benutzer (Hardware Security Module) mit CloudHSM Management Utility (CMU) erstellen. Weitere Informationen finden Sie unter Grundlegendes zu HSM-Benutzerverwaltungsaufgaben, Erste Schritte mit der AWS CloudHSM-Befehlszeilenschnittstelle (CLI) und So verwalten Sie HSM-Benutzer.

> **Note**
>
> Wenn Sie Client-SDK 3 verwenden, verwenden Sie CloudHSM Management Utility (CMU) anstelle von CloudHSM-CLI.

Wenn HSM-Benutzer eingerichtet sind, können Sie sich beim HSM anmelden und Schlüssel mit einer der folgenden Optionen erstellen und verwenden:

- Verwenden Sie das Schlüsselverwaltungsprogramm, ein Befehlszeilentool
- Erstellen Sie eine C-Anwendung mit der PKCS #11-Bibliothek
- Erstellen Sie eine Java-Anwendung mit dem JCE-Anbieter
- Verwenden Sie die OpenSSL Dynamic Engine direkt von der Befehlszeile aus
- Verwenden Sie die OpenSSL Dynamic Engine für TLS-Offload mit NGINX- und Apache-Webservern
- Verwenden Sie die CNG- und KSP-Anbieter für die Verwendung von AWS CloudHSM mit der Microsoft Windows Server Certificate Authority (CA)
- Verwenden Sie die CNG- und KSP-Anbieter für die Verwendung von AWS CloudHSM mit dem Microsoft Sign Tool
- Verwenden Sie die CNG- und KSP-Anbieter für das TLS-Offload mit dem Internet Information Server (IIS)-Webserver

Bewährte Methoden für AWS CloudHSM

Führen Sie die bewährten Methoden in diesem Thema durch, um sie effektiv zu nutzen AWS CloudHSM.

Inhalt

Clusterverwaltung

Folgen Sie den bewährten Methoden in diesem Abschnitt, wenn Sie Ihren AWS CloudHSM Cluster erstellen, darauf zugreifen und ihn verwalten.

Skalieren Sie Ihren Cluster, um den Spitzenverkehr zu bewältigen

Verschiedene Faktoren können den maximalen Durchsatz beeinflussen, den Ihr Cluster verarbeiten kann, darunter die Größe der Client-Instanz, die Clustergröße, die Netzwerktopographie und die kryptografischen Operationen, die Sie für Ihren Anwendungsfall benötigen.

Als Ausgangspunkt finden Sie im Thema AWS CloudHSM Leistung Leistungsschätzungen zu gängigen Clustergrößen und -konfigurationen. Wir empfehlen Ihnen, Ihren Cluster mit der zu erwartenden Spitzenlast zu testen, um festzustellen, ob Ihre aktuelle Architektur robust ist und die richtige Skalierung aufweist.

Richten Sie Ihren Cluster auf Hochverfügbarkeit aus

Fügen Sie Redundanz hinzu, um die Wartung zu berücksichtigen: AWS kann Ihr HSM bei geplanten Wartungsarbeiten oder wenn ein Problem erkannt wird, ersetzen. In der Regel sollte Ihre Clustergröße mindestens +1 Redundanz aufweisen. Wenn Sie beispielsweise zwei HSMs benötigen, damit Ihr Service in Spitzenzeiten betrieben werden kann, ist Ihre ideale Clustergröße dann drei. Wenn Sie die bewährten Methoden in Bezug auf die Verfügbarkeit befolgen, sollten sich diese HSM-

Ersetzungen nicht auf Ihren Service auswirken. Laufende Operationen am ausgetauschten HSM können jedoch fehlschlagen und müssen erneut versucht werden.

Verteilen Sie Ihre HSMs auf viele Availability Zones: Überlegen Sie, wie Ihr Service während eines Ausfalls in der Availability Zone funktionieren kann. AWS empfiehlt, dass Sie Ihre HSMs auf so viele Availability Zones wie möglich verteilen. Bei einem Cluster mit drei HSMs sollten Sie HSMs auf drei Availability Zones verteilen. Abhängig von Ihrem System benötigen Sie möglicherweise zusätzliche Redundanz.

Verwenden Sie mindestens drei HSMs, um die Haltbarkeit neu generierter Schlüssel zu gewährleisten

Für Anwendungen, die die Haltbarkeit neu generierter Schlüssel erfordern, empfehlen wir, mindestens drei HSMs auf verschiedene Availability Zones in einer Region zu verteilen.

Sicherer Zugriff auf Ihren Cluster

Verwenden Sie private Subnetze, um den Zugriff auf Ihre Instance zu beschränken: Starten Sie Ihre HSMs und Client-Instances in den privaten Subnetzen Ihrer VPC. Dadurch wird der Zugriff auf Ihre HSMs von außen eingeschränkt.

Verwenden Sie VPC-Endpunkte für den Zugriff auf APIs: Die AWS CloudHSM Datenebene wurde so konzipiert, dass sie ohne Zugriff auf das Internet oder AWS-APIs funktioniert. Wenn Ihre Client-Instance Zugriff auf die AWS CloudHSM API benötigt, können Sie VPC-Endpunkte verwenden, um auf die API zuzugreifen, ohne dass Sie einen Internetzugang auf Ihrer Client-Instance benötigen. Weitere Informationen finden Sie unter AWS CloudHSM und VPC-Endpunkte.

Konfigurieren Sie SSL neu, um die Client-Server-Kommunikation zu sichern: AWS CloudHSM Verwendet TLS, um eine Verbindung zu Ihrem HSM herzustellen. Nachdem Sie Ihren Cluster initialisiert haben, können Sie das Standard-TLS-Zertifikat und den Schlüssel, mit denen die äußere TLS-Verbindung hergestellt wurde, ersetzen. Weitere Informationen finden Sie unter Verbessern Sie die Sicherheit Ihres Webservers mit SSL/TLS-Offload in AWS CloudHSM.

Senken Sie die Kosten, indem Sie Ihren Bedürfnissen entsprechend skalieren

Es fallen keine Vorabkosten für die Nutzung AWS CloudHSM an. Sie zahlen für jedes HSM, das Sie starten, eine Stundengebühr, bis Sie das HSM kündigen. Wenn Ihr Service keine kontinuierliche

Nutzung von erfordert, können Sie die Kosten senken AWS CloudHSM, indem Sie Ihre HSMs auf Null herunterskalieren (löschen), wenn sie nicht benötigt werden. Wenn HSMs erneut benötigt werden, können Sie Ihre HSMs aus einem Backup wiederherstellen. Wenn Sie beispielsweise einen Workload haben, bei dem Sie Code einmal im Monat signieren müssen, und zwar am letzten Tag des Monats, können Sie Ihren Cluster vorher hochskalieren, ihn herunterskalieren, indem Sie Ihre HSMs nach Abschluss der Arbeit löschen, und dann Ihren Cluster wiederherstellen, um am Ende des nächsten Monats erneut Signierungsvorgänge durchzuführen.

AWS CloudHSM erstellt automatisch regelmäßige Backups der HSMs im Cluster. Wenn Sie zu einem späteren Zeitpunkt ein neues HSM hinzufügen, AWS CloudHSM wird das letzte Backup auf dem neuen HSM wiederhergestellt, sodass Sie es an derselben Stelle wieder verwenden können, an der Sie es verlassen haben. Informationen zur Berechnung Ihrer AWS CloudHSM Architekturkosten finden Sie unter AWS CloudHSM Preise.

Verwandte Ressourcen:

- Allgemeiner Überblick über Backups
- Richtlinie zur Aufbewahrung von Backup
- Kopieren von Backups zwischen AWS Regionen

HSM-Benutzerverwaltung

Folgen Sie den bewährten Methoden in diesem Abschnitt, um Benutzer in Ihrem AWS CloudHSM Cluster effektiv zu verwalten. HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer und Entitäten, die über eine identitätsbasierte Richtlinie mit den entsprechenden Berechtigungen verfügen, können HSMs erstellen, indem sie über die AWS-API mit Ressourcen interagieren. Nachdem das HSM erstellt wurde, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren. Eine ausführliche Anleitung für HSM-Benutzer finden Sie unter. Verwalten von HSM-Benutzern in AWS CloudHSM

Schützen Sie die Anmeldeinformationen Ihrer HSM-Benutzer

Es ist unerlässlich, die Anmeldeinformationen Ihrer HSM-Benutzer sicher zu schützen, da HSM-Benutzer die Entitäten sind, die auf Ihr HSM zugreifen und kryptografische und Verwaltungsvorgänge auf Ihrem HSM ausführen können. AWS CloudHSM hat keinen Zugriff auf Ihre HSM-Benutzeranmeldedaten und kann Ihnen nicht weiterhelfen, wenn Sie den Zugriff darauf verlieren.

Haben Sie mindestens zwei Administratoren, um eine Aussperrung zu verhindern

Um zu verhindern, dass Sie aus Ihrem Cluster ausgesperrt werden, empfehlen wir, dass Sie mindestens zwei Administratoren haben, falls ein Administratorkennwort verloren geht. In diesem Fall können Sie den anderen Administrator verwenden, um das Passwort zurückzusetzen.

> **Note**
>
> Admins im Client SDK 5 sind ein Synonym für Crypto Officers (COs) im Client SDK 3.

Aktivieren Sie das Quorum für alle Benutzerverwaltungsvorgänge

Mit Quorum können Sie eine Mindestanzahl von Administratoren festlegen, die einen Benutzerverwaltungsvorgang genehmigen müssen, bevor dieser Vorgang durchgeführt werden kann. Aufgrund der Rechte, die Administratoren haben, empfehlen wir, Quorum für alle Benutzerverwaltungsvorgänge zu aktivieren. Dies kann die potenziellen Auswirkungen einschränken, wenn eines Ihrer Administratorkennwörter kompromittiert wird. Weitere Informationen finden Sie unter Quorum verwalten.

Erstellen Sie mehrere Krypto-Benutzer, von denen jeder über eingeschränkte Berechtigungen verfügt

Durch die Trennung der Verantwortlichkeiten der Krypto-Benutzer hat kein Benutzer die vollständige Kontrolle über das gesamte System. Aus diesem Grund empfehlen wir Ihnen, mehrere Krypto-Benutzer zu erstellen und deren Berechtigungen einzuschränken. In der Regel erfolgt dies, indem verschiedenen Krypto-Benutzern deutlich unterschiedliche Verantwortlichkeiten und Aktionen zugewiesen werden (z. B. ein Krypto-Benutzer, der für die Generierung und gemeinsame Nutzung von Schlüsseln mit anderen Krypto-Benutzern verantwortlich ist, die diese dann in Ihrer Anwendung verwenden).

Verwandte Ressourcen:

- key share
- key unshare

HSM-Schlüsselverwaltung

Befolgen Sie die bewährten Methoden in diesem Abschnitt bei der Verwaltung von Schlüsseln in AWS CloudHSM.

Wählen Sie den richtigen Schlüsseltyp

Wenn Sie einen Sitzungsschlüssel verwenden, sind Ihre Transaktionen pro Sekunde (TPS) auf ein HSM beschränkt, für das der Schlüssel vorhanden ist. Zusätzliche HSMs in Ihrem Cluster erhöhen den Durchsatz der Anfragen für diesen Schlüssel nicht. Wenn Sie einen Token-Schlüssel für dieselbe Anwendung verwenden, erfolgt ein Lastenausgleich für Ihre Anfragen auf alle verfügbaren HSMs in Ihrem Cluster. Weitere Informationen finden Sie unter Schlüsselsynchronisations- und Haltbarkeitseinstellungen in AWS CloudHSM.

Verwalten Sie wichtige Speicherlimits

Bei HSMs ist die maximale Anzahl von Token- und Sitzungsschlüsseln, die gleichzeitig auf einem HSM gespeichert werden können, begrenzt. Informationen zu Speicherbeschränkungen für Schlüssel finden Sie unter. AWS CloudHSM-Kontingente Wenn Ihre Anwendung mehr als das Limit benötigt, können Sie eine oder mehrere der folgenden Strategien anwenden, um Schlüssel effektiv zu verwalten:

Verwenden Sie Trusted Wrapping, um Ihre Schlüssel in einem externen Datenspeicher zu speichern: Mit Trusted Key Wrapping können Sie das Speicherlimit für Schlüssel umgehen, indem Sie alle Ihre Schlüssel in einem externen Datenspeicher speichern. Wenn Sie diesen Schlüssel verwenden müssen, können Sie ihn als Sitzungsschlüssel in das HSM entpacken, den Schlüssel für den gewünschten Vorgang verwenden und dann den Sitzungsschlüssel verwerfen. Die ursprünglichen Schlüsseldaten bleiben sicher in Ihrem Datenspeicher gespeichert, sodass Sie sie jederzeit verwenden können. Wenn Sie zu diesem Zweck vertrauenswürdige Schlüssel verwenden, wird Ihr Schutz maximiert.

Schlüssel auf Cluster verteilen: Eine weitere Strategie zur Überwindung des Speicherlimits für Schlüssel besteht darin, Ihre Schlüssel in mehreren Clustern zu speichern. Bei diesem Ansatz behalten Sie eine Zuordnung der Schlüssel bei, die in jedem Cluster gespeichert sind. Verwenden Sie diese Zuordnung, um Ihre Client-Anfragen mit dem erforderlichen Schlüssel an den Cluster weiterzuleiten. Informationen dazu, wie Sie von derselben Client-Anwendung aus eine Verbindung zu mehreren Clustern herstellen können, finden Sie in den folgenden Themen:

- Verbindung zu mehreren Clustern mit dem JCE-Anbieter herstellen

- Verbindung zu mehreren Steckplätzen mit PKCS #11

Verwaltung und Sicherung der Schlüsselverpackung

Schlüssel können über das Attribut entweder als extrahierbar oder nicht extrahierbar gekennzeichnet werden. `EXTRACTABLE` Standardmäßig sind HSM-Schlüssel als extrahierbar gekennzeichnet.

Extrahierbare Schlüssel sind Schlüssel, die durch Key-Wrapping aus dem HSM exportiert werden dürfen. Umhüllte Schlüssel sind verschlüsselt und müssen mit demselben Wrapping-Schlüssel entpackt werden, bevor sie verwendet werden können. Nicht extrahierbare Schlüssel dürfen unter keinen Umständen aus dem HSM exportiert werden. Es gibt keine Möglichkeit, einen nicht extrahierbaren Schlüssel extrahierbar zu machen. Aus diesem Grund ist es wichtig zu überlegen, ob Ihre Schlüssel extrahierbar sein müssen oder nicht, und das entsprechende identifizierende Attribut entsprechend festzulegen.

Wenn Sie in Ihrer Anwendung das Umschließen von Schlüsseln erfordern, sollten Sie Trusted Key Wrapping verwenden, um die Fähigkeit Ihrer HSM-Benutzer zu beschränken, nur Schlüssel zu umschließen/zu entpacken, die von einem Administrator ausdrücklich als vertrauenswürdig markiert wurden. Weitere Informationen finden Sie in den Themen zum Einschließen vertrauenswürdiger Schlüssel. Verwalten von Schlüsseln in AWS CloudHSM

Zugehörige Ressourcen

- Funktionen „Wrap“ und „Unwrap“
- Verschlüsselungsfunktionen für JCE
- Unterstützte Java-Schlüsselattribute
- Schlüsselattribute für CloudHSM-CLI

Anwendungsintegration

Folgen Sie den bewährten Methoden in diesem Abschnitt, um die Integration Ihrer Anwendung in Ihren Cluster zu optimieren. AWS CloudHSM

Bootstrap für Ihr Client-SDK

Bevor Ihr Client-SDK eine Verbindung zu Ihrem Cluster herstellen kann, muss es gebootet werden. Beim Bootstrapping von IP-Adressen zu Ihrem Cluster empfehlen wir, den `--cluster-id`

Parameter nach Möglichkeit zu verwenden. Diese Methode füllt Ihre Konfiguration mit allen HSM-IP-Adressen in Ihrem Cluster, ohne dass Sie jede einzelne Adresse verfolgen müssen. Auf diese Weise wird Ihre Anwendungsinitialisierung für den Fall, dass ein HSM gewartet wird oder während eines Ausfalls der Availability Zone, zusätzliche Stabilität verleiht. Weitere Details finden Sie unter Bootstrap für das Client-SDK.

Authentifizieren Sie sich, um Operationen auszuführen

In AWS CloudHSM müssen Sie sich bei Ihrem Cluster authentifizieren, bevor Sie die meisten Operationen, wie z. B. kryptografische Operationen, ausführen können.

Authentifizieren Sie sich mit der CloudHSM-CLI: Sie können sich mit der CloudHSM-CLI entweder im Einzelbefehlsmodus oder im interaktiven Modus authentifizieren. Verwenden Sie den login Befehl, um sich im interaktiven Modus zu authentifizieren. Um sich im Einzelbefehlsmodus zu authentifizieren, müssen Sie die Umgebungsvariablen `CLOUDHSM_ROLE` und festlegen. `CLOUDHSM_PIN` Einzelheiten dazu finden Sie unter. Einzelbefehlsmodus AWS CloudHSM empfiehlt, Ihre HSM-Anmeldeinformationen sicher zu speichern, wenn sie nicht von Ihrer Anwendung verwendet werden.

Authentifizieren Sie sich mit PKCS #11: In PKCS #11 melden Sie sich mit der C_Login-API an, nachdem Sie eine Sitzung mit C_ geöffnet haben. OpenSession Sie müssen nur einen C_Login pro Slot (Cluster) durchführen. Nachdem Sie sich erfolgreich angemeldet haben, können Sie weitere Sitzungen mit C_ öffnen, OpenSession ohne zusätzliche Anmeldevorgänge durchführen zu müssen. Beispiele für die Authentifizierung bei PKCS #11 finden Sie unter. Codebeispiele für die PKCS #11-Bibliothek

Mit JCE authentifizieren: Der AWS CloudHSM JCE Provider unterstützt sowohl implizite als auch explizite Anmeldung. Welche Methode für Sie funktioniert, hängt von Ihrem Anwendungsfall ab. Wenn möglich, empfehlen wir die Verwendung von Implicit Login, da das SDK die Authentifizierung automatisch übernimmt, wenn Ihre Anwendung vom Cluster getrennt wird und erneut authentifiziert werden muss. Wenn Sie die implizite Anmeldung verwenden, können Sie auch Anmeldeinformationen für Ihre Anwendung angeben, wenn Sie eine Integration verwenden, bei der Sie keine Kontrolle über Ihren Anwendungscode haben. Weitere Informationen zu Anmeldemethoden finden Sie unterGeben Sie Anmeldeinformationen für den JCE-Anbieter ein.

Authentifizieren Sie sich mit OpenSSL: Mit der OpenSSL Dynamic Engine geben Sie Anmeldeinformationen über Umgebungsvariablen an. AWS CloudHSM empfiehlt, Ihre HSM-Anmeldeinformationen sicher zu speichern, wenn sie nicht von Ihrer Anwendung verwendet werden. Wenn möglich, sollten Sie Ihre Umgebung so konfigurieren, dass diese Umgebungsvariablen

systematisch und ohne manuelle Eingabe abgerufen und festgelegt werden. Einzelheiten zur Authentifizierung mit OpenSSL finden Sie unter. Installieren von OpenSSL Dynamic Engine

Verwalten Sie Schlüssel in Ihrer Anwendung effektiv

Verwenden Sie Schlüsselattribute, um zu steuern, was Schlüssel bewirken können: Verwenden Sie beim Generieren eines Schlüssels Schlüsselattribute, um eine Reihe von Berechtigungen zu definieren, die bestimmte Arten von Vorgängen für diesen Schlüssel zulassen oder verweigern. Wir empfehlen, Schlüssel mit der geringsten Anzahl an Attributen zu generieren, die zur Ausführung ihrer Aufgabe erforderlich sind. Beispielsweise sollte ein AES-Schlüssel, der für die Verschlüsselung verwendet wird, nicht auch Schlüssel aus dem HSM herausnehmen dürfen. Weitere Informationen finden Sie auf unseren Attributseiten für die folgenden Client-SDKs:

- PKCS #11-Schlüsselattribute
- JCE-Schlüsselattribute

Wenn möglich, sollten Sie wichtige Objekte zwischenspeichern, um die Latenz zu minimieren: Bei Schlüsselsuchoperationen wird jedes HSM in Ihrem Cluster abgefragt. Dieser Vorgang ist teuer und lässt sich nicht mit der HSM-Anzahl in Ihrem Cluster skalieren.

- Mit PKCS #11 finden Sie Schlüssel mithilfe der `C_FindObjects` API.
- Mit JCE finden Sie Schlüssel mithilfe der. KeyStore

Um eine optimale Leistung zu erzielen, AWS empfiehlt es sich, Befehle zur Schlüsselsuche (wie findKey undkey list) nur einmal beim Start der Anwendung zu verwenden und das zurückgegebene Schlüsselobjekt im Anwendungsspeicher zwischenzuspeichern. Wenn Sie dieses Schlüsselobjekt zu einem späteren Zeitpunkt benötigen, sollten Sie das Objekt aus Ihrem Cache abrufen, anstatt es bei jedem Vorgang abzufragen, was zu einem erheblichen Leistungsaufwand führt.

Verwenden Sie Multithreading

AWS CloudHSM unterstützt Multithread-Anwendungen, aber es gibt einige Dinge, die Sie bei Multithread-Anwendungen beachten sollten.

Mit PKCS #11 sollten Sie die PKCS #11 -Bibliothek (Aufruf) nur einmal initialisieren. `C_Initialize` Jedem Thread sollte eine eigene Sitzung () zugewiesen werden. `C_OpenSession` Es wird nicht empfohlen, dieselbe Sitzung in mehreren Threads zu verwenden.

Bei JCE sollte der AWS CloudHSM Anbieter nur einmal initialisiert werden. Teilen Sie Instanzen von SPI-Objekten nicht über mehrere Threads hinweg. Beispielsweise sollten Cipher, Signature, Digest, Mac KeyFactory oder KeyGenerator Objekte nur im Kontext ihres eigenen Threads verwendet werden.

Gehen Sie mit Drosselungsfehlern um

Unter den folgenden Umständen können HSM-Drosselungsfehler auftreten:

- Ihr Cluster ist nicht richtig skaliert, um Spitzenverkehr zu bewältigen.
- Ihr Cluster verfügt nicht über eine +1-Redundanz bei Wartungsereignissen.
- Ausfälle in der Availability Zone führen zu einer geringeren Anzahl verfügbarer HSMs in Ihrem Cluster.

Informationen HSM-Drosselung zur optimalen Handhabung dieses Szenarios finden Sie unter.

Um sicherzustellen, dass Ihr Cluster ausreichend dimensioniert ist und nicht gedrosselt wird, AWS empfiehlt es sich, einen Lasttest in Ihrer Umgebung mit dem zu erwartenden Spitzenverkehr durchzuführen.

Integrieren Sie Wiederholungsversuche bei Clustervorgängen

AWS kann Ihr HSM aus Betriebs- oder Wartungsgründen ersetzen. Um Ihre Anwendung gegen solche Situationen widerstandsfähig zu machen, AWS empfiehlt es sich, für alle Operationen, die an Ihren Cluster weitergeleitet werden, eine clientseitige Wiederholungslogik zu implementieren. Nachfolgende Wiederholungen bei fehlgeschlagenen Vorgängen aufgrund von Ersatzvorgängen werden voraussichtlich erfolgreich sein.

Implementieren Sie Strategien für die Notfallwiederherstellung

Als Reaktion auf ein Ereignis kann es erforderlich sein, Ihren Datenverkehr von einem ganzen Cluster oder einer Region weg zu verlagern. In den folgenden Abschnitten werden mehrere Strategien beschrieben, um dies zu erreichen.

Verwenden Sie VPC-Peering, um von einem anderen Konto oder einer anderen Region aus auf Ihren Cluster zuzugreifen: Sie können VPC-Peering verwenden, um von einem anderen Konto oder einer anderen Region aus auf Ihren AWS CloudHSM Cluster zuzugreifen. Informationen zur

Einrichtung finden Sie unter Was ist VPC-Peering? im VPC Peering Guide. Sobald Sie Ihre Peering-Verbindungen eingerichtet und Ihre Sicherheitsgruppen entsprechend konfiguriert haben, können Sie wie gewohnt mit HSM-IP-Adressen kommunizieren.

Connect zu mehreren Clustern aus derselben Anwendung her: Der JCE-Anbieter, die PKCS #11 -Bibliothek und die CLI im Client SDK 5 unterstützen die Verbindung zu mehreren Clustern aus derselben Anwendung. Sie können beispielsweise über zwei aktive Cluster verfügen, die sich jeweils in unterschiedlichen Regionen befinden, und Ihre Anwendung kann eine Verbindung zu beiden gleichzeitig herstellen und im Rahmen des normalen Betriebs einen Lastenausgleich zwischen den beiden durchführen. Wenn Ihre Anwendung nicht das Client SDK 5 (das neueste SDK) verwendet, können Sie von derselben Anwendung aus keine Verbindung zu mehreren Clustern herstellen. Alternativ können Sie einen anderen Cluster am Laufen halten und, falls es zu einem regionalen Ausfall kommt, Ihren Datenverkehr auf den anderen Cluster verlagern, um Ausfallzeiten zu minimieren. Einzelheiten finden Sie auf den jeweiligen Seiten:

- Verbindung zu mehreren Steckplätzen mit PKCS #11
- Verbindung zu mehreren Clustern mit dem JCE-Anbieter herstellen
- Mit CLI eine Verbindung zu mehreren Clustern herstellen

Einen Cluster aus einem Backup wiederherstellen: Sie können einen neuen Cluster aus einer Sicherung eines vorhandenen Clusters erstellen. Weitere Informationen finden Sie unter AWS CloudHSM Backups verwalten.

Überwachen

In diesem Abschnitt werden mehrere Mechanismen beschrieben, mit denen Sie Ihren Cluster und Ihre Anwendung überwachen können. Weitere Informationen zur Überwachung finden Sie unterÜberwachung von AWS CloudHSM.

Überwachen Sie die Client-Protokolle

Jedes Client-SDK schreibt Protokolle, die Sie überwachen können. Informationen zur Client-Protokollierung finden Sie unterArbeiten mit Client-SDK-Protokollen.

Auf Plattformen, die so konzipiert sind, dass sie kurzlebig sind, wie Amazon ECS und AWS Lambda, kann das Sammeln von Client-Protokollen aus einer Datei schwierig sein. In diesen Situationen empfiehlt es sich, die Client-SDK-Protokollierung so zu konfigurieren, dass Protokolle in die Konsole

geschrieben werden. Die meisten Dienste sammeln diese Ausgabe automatisch und veröffentlichen sie in CloudWatch Amazon-Protokollen, damit Sie sie behalten und einsehen können.

Wenn Sie zusätzlich zum AWS CloudHSM Client-SDK eine Drittanbieter-Integration verwenden, sollten Sie sicherstellen, dass Sie das Softwarepaket so konfigurieren, dass auch die Ausgabe in der Konsole protokolliert wird. Die Ausgabe des AWS CloudHSM Client-SDK kann von diesem Paket erfasst und andernfalls in eine eigene Protokolldatei geschrieben werden.

Informationen Configure-Tool für das Client-SDK 5 zur Konfiguration der Protokollierungsoptionen in Ihrer Anwendung finden Sie im.

Überwachen Sie die Überwachungsprotokolle

AWS CloudHSM veröffentlicht Auditprotokolle auf Ihrem CloudWatch Amazon-Konto. Die Auditprotokolle stammen aus dem HSM und verfolgen bestimmte Vorgänge zu Prüfungszwecken.

Sie können Auditprotokolle verwenden, um alle Verwaltungsbefehle zu verfolgen, die auf Ihrem HSM aufgerufen werden. Sie können beispielsweise einen Alarm auslösen, wenn Sie feststellen, dass ein unerwarteter Verwaltungsvorgang ausgeführt wird.

Weitere Details finden Sie unter So funktioniert die HSM-Auditprotokollierung.

Überwachen AWS CloudTrail

AWS CloudHSM ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Dienst in ausgeführt wurden AWS CloudHSM. AWS CloudTrail erfasst alle API-Aufrufe AWS CloudHSM als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der AWS CloudHSM Konsole und Codeaufrufen für die AWS CloudHSM API-Operationen.

Sie können AWS CloudTrail damit jeden API-Aufruf überprüfen, der an die AWS CloudHSM Steuerungsebene gesendet wird, um sicherzustellen, dass in Ihrem Konto keine unerwünschten Aktivitäten stattfinden.

Details dazu finden Sie unter Arbeiten mit AWS CloudTrail und AWS CloudHSM.

Überwachen Sie CloudWatch Amazon-Metriken

Sie können CloudWatch Amazon-Metriken verwenden, um Ihren AWS CloudHSM Cluster in Echtzeit zu überwachen. Die Metriken können nach Region, Cluster-ID oder HSM-ID und Cluster-ID gruppiert werden.

Mithilfe von CloudWatch Amazon-Metriken können Sie CloudWatch Amazon-Alarme so konfigurieren, dass Sie vor potenziellen Problemen gewarnt werden, die sich auf Ihren Service auswirken könnten. Wir empfehlen, Alarme zu konfigurieren, um Folgendes zu überwachen:

- Wir nähern uns Ihrem Tasten-Limit auf einem HSM
- Wir nähern uns dem Limit für die Anzahl der HSM-Sitzungen auf einem HSM
- Annäherung an das Limit für die HSM-Benutzeranzahl auf einem HSM
- Unterschiede in der HSM-Benutzer- oder Schlüsselanzahl zur Identifizierung von Synchronisationsproblemen
- Fehlerhafte HSMs, sodass Sie Ihren Cluster hochskalieren müssen, bis das Problem behoben AWS CloudHSM werden kann

Weitere Details finden Sie unter Arbeiten mit Amazon CloudWatch Logs und AWS CloudHSM Audit Logs.

Verwaltung von AWS CloudHSM Clustern

Sie können Ihre AWS CloudHSM Cluster über die AWS CloudHSM Konsole oder eines der AWS SDKs oder Befehlszeilentools verwalten. Weitere Informationen finden Sie unter den folgenden Themen.

Informationen zum Erstellen eines Clusters finden Sie unter Erste Schritte.

Clusterarchitektur

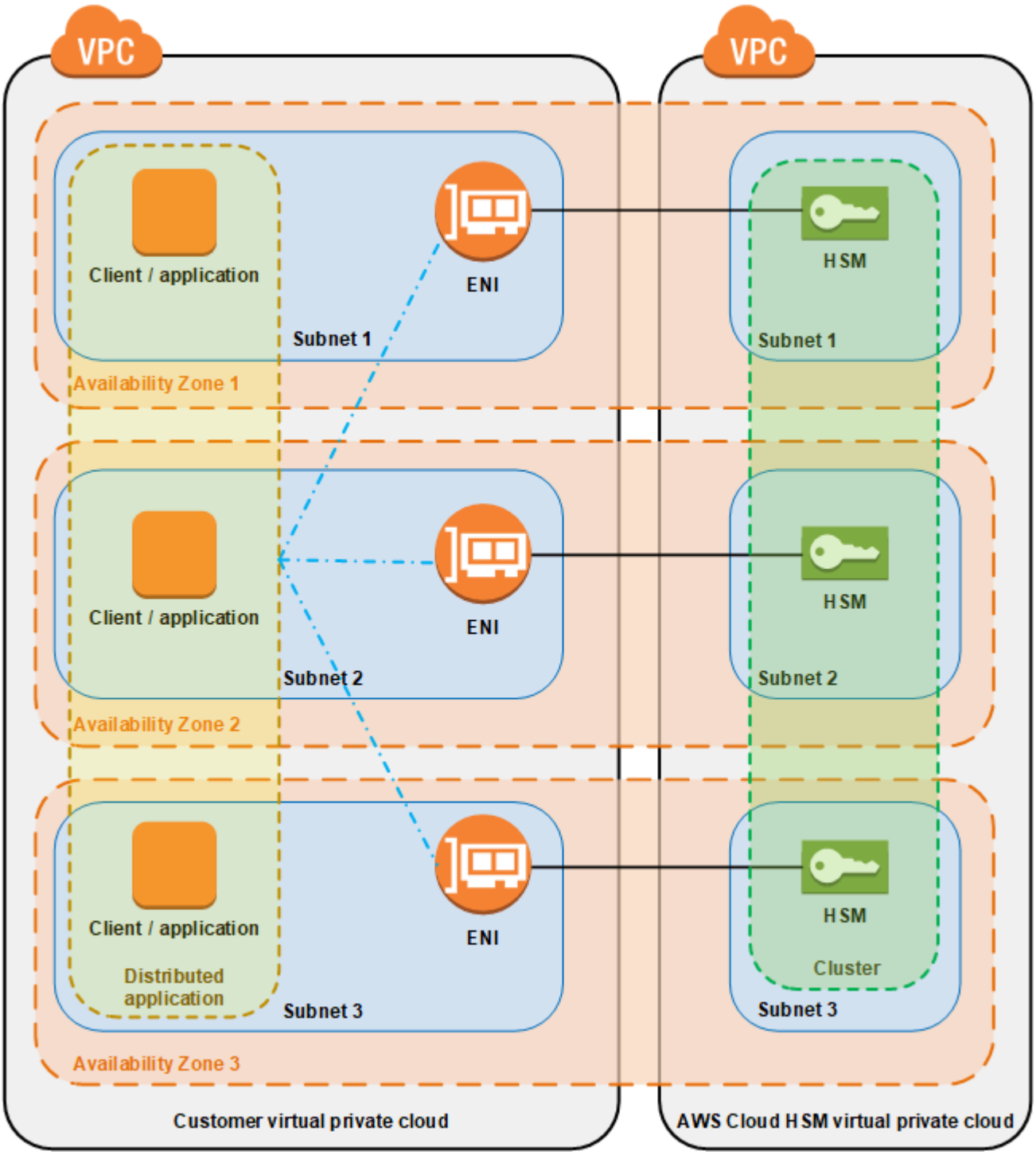
Wenn Sie einen Cluster erstellen, geben Sie eine Amazon Virtual Private Cloud (VPC) in Ihrem AWS Konto und ein oder mehrere Subnetze in dieser VPC an. Wir empfehlen, dass Sie in jeder Availability Zone (AZ) in der von Ihnen ausgewählten Region ein Subnetz erstellen. AWS Sie können private Subnetze erstellen, wenn Sie eine VPC erstellen. Weitere Informationen hierzu finden Sie unter Erstellen einer Virtual Private Cloud (VPC).

Jedes Mal, wenn Sie ein HSM-Cluster erstellen, geben Sie den Cluster und die Availability Zone für das HSM. Wenn Sie die HSMs in verschiedenen Availability Zones unterbringen, erhalten Sie Redundanz und hohe Verfügbarkeit, falls eine Availability Zone nicht verfügbar ist.

Wenn Sie ein HSM erstellen, AWS CloudHSM fügt es ein elastic network interface (ENI) in das angegebene Subnetz in Ihrem AWS Konto ein. Die Elastic Network-Schnittstelle ist die Schnittstelle für die Interaktion mit dem HSM. Das HSM befindet sich in einer separaten VPC in einem AWS Konto, das Eigentum von ist. AWS CloudHSM Das HSM und die entsprechende Netzwerkschnittstelle befinden sich in derselben Availability Zone.

Um mit den HSMs in einem Cluster zu interagieren, benötigen Sie die Client-Software. AWS CloudHSM Normalerweise installieren Sie den Client auf Amazon EC2-Instances, den so genannten Client-Instances, die sich in derselben VPC befinden wie die HSM ENIs, wie in der folgenden Abbildung gezeigt. Das ist technisch nicht erforderlich, Sie können den Client auf jedem kompatiblen Computer installieren, solange er eine Verbindung mit dem HSM ENIs einrichten kann. Der Client kommuniziert mit den einzelnen HSMs in Ihrem Cluster über ihre ENIs.

Die folgende Abbildung stellt einen AWS CloudHSM Cluster mit drei HSMs dar, die sich jeweils in einer anderen Availability Zone in der VPC befinden.



Synchronisierung von Clustern

AWS CloudHSM Hält in einem AWS CloudHSM Cluster die Schlüssel auf den einzelnen HSMs synchron. Sie brauchen die Schlüssel auf Ihren HSMs nicht manuell zu synchronisieren. Um die Benutzer und Richtlinien auf jedem HSM synchron zu halten, aktualisieren Sie die AWS CloudHSM

Client-Konfigurationsdatei, bevor Sie HSM-Benutzer verwalten. Weitere Informationen finden Sie unter Aufrechthalten der Synchronität von HSM-Benutzern.

Wenn Sie einem Cluster ein neues HSM hinzufügen, AWS CloudHSM erstellt eine Sicherungskopie aller Schlüssel, Benutzer und Richtlinien auf einem vorhandenen HSM. Anschließend stellt es diese Sicherung auf dem neuen HSM wieder her. Auf diese Weise wird sichergestellt, dass die beiden HSMs synchronisiert bleiben.

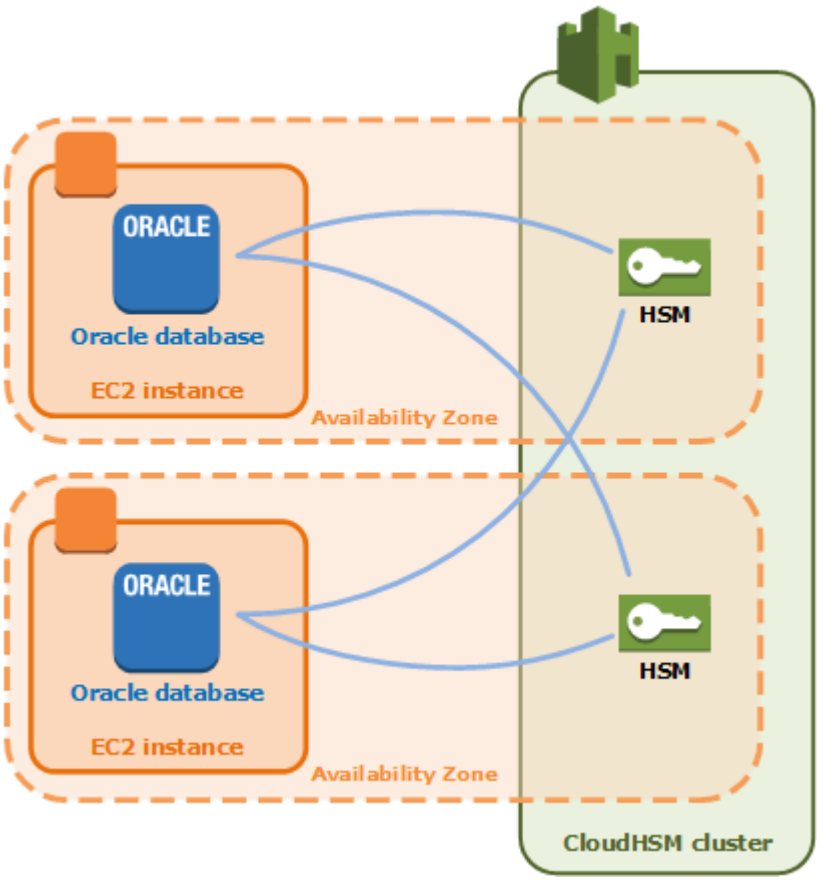
Wenn die HSMs in einem Cluster nicht mehr synchronisiert AWS CloudHSM werden, werden sie automatisch neu synchronisiert. Um dies zu aktivieren, werden die Anmeldeinformationen des Appliance-Benutzers AWS CloudHSM verwendet. Dieser Benutzer ist auf allen HSMs vorhanden, die von bereitgestellt werden, AWS CloudHSM und verfügt über eingeschränkte Berechtigungen. Er kann einen Hash von Objekten auf dem HSM erhalten und maskierte (verschlüsselte) Objekte extrahieren und einfügen. AWS kann Ihre Benutzer oder Schlüssel weder anzeigen oder ändern, noch mit diesen Schlüsseln kryptografischen Operationen durchführen.

Hohe Verfügbarkeit und Load Balancing von Clustern

Wenn Sie einen AWS CloudHSM Cluster mit mehr als einem HSM erstellen, erhalten Sie automatisch einen Lastenausgleich. Load Balancing bedeutet, dass der AWS CloudHSM -Client kryptografische Operationen über alle HSMs in dem Cluster verteilt, basierend auf der Kapazität jedes HSM für zusätzliche Verarbeitungsleistungen.

Wenn Sie die HSMs in verschiedenen AWS Availability Zones erstellen, erhalten Sie automatisch Hochverfügbarkeit. Hohe Verfügbarkeit bedeutet, dass Sie die Zuverlässigkeit verbessern, da keines der HSMs für sich einen Single Point of Failure darstellt. Wir empfehlen, dass Sie mindestens zwei HSMs in jedem Cluster haben, wobei sich jedes HSM in verschiedenen Availability Zones innerhalb einer Region befindet. AWS

Die folgende Abbildung zeigt z. B. eine Oracle-Datenbankanwendung, die auf zwei verschiedene Availability Zones verteilt ist. Die Datenbank-Instances speichern ihre Masterschlüssel in einem Cluster, der in jeder Availability Zone ein HSM enthält. AWS CloudHSM synchronisiert die Schlüssel automatisch mit beiden HSMs, sodass sie sofort zugänglich und redundant sind.



Connect das Client-SDK mit dem AWS CloudHSM Cluster

Um mit Client-SDK 5 oder Client-SDK 3 eine Verbindung zum Cluster herzustellen, müssen Sie zunächst zwei Dinge tun:

- Verfügen Sie über ein ausstellendes Zertifikat auf der EC2-Instance
- Das Client-SDK per Bootstrap auf den Cluster übertragen

Platzieren Sie das ausstellende Zertifikat auf jeder EC2-Instance

Sie erstellen das ausstellende Zertifikat, wenn Sie den Cluster initialisieren. Kopieren Sie das ausstellende Zertifikat an den Standardspeicherort für die Plattform auf jeder EC2-Instance, die eine Verbindung zum Cluster herstellt.

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

Geben Sie den Speicherort des ausstellenden Zertifikats an

Beim Client-SDK 5 verwenden Sie das Configure-Tool, um den Speicherort des ausstellenden Zertifikats anzugeben.

PKCS #11 library

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --hsm-ca-cert <customerCA certificate
 file>
```

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --hsm-ca-
cert <customerCA certificate file>
```

OpenSSL Dynamic Engine

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --hsm-ca-cert <customerCA certificate
 file>
```

JCE provider

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce --hsm-ca-cert <customerCA certificate
 file>
```

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --hsm-ca-cert <customerCA
 certificate file>
```

CloudHSM CLI

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli --hsm-ca-cert <customerCA certificate
 file>
```

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --hsm-ca-cert <customerCA
 certificate file>
```

Weitere Informationen finden Sie unter Configure Tool.

Weitere Informationen zur Initialisierung des Clusters oder zum Erstellen und Signieren des Zertifikats finden Sie unter Cluster initialisieren.

Bootstrap für das Client-SDK

Der Bootstrap-Vorgang ist je nach der Version des Client-SDK, die Sie verwenden, unterschiedlich. Sie benötigen jedoch die IP-Adresse eines der Hardware-Sicherheitsmodule (HSM) im Cluster. Sie können die IP-Adresse eines beliebigen HSM verwenden, das an Ihren Cluster angeschlossen ist. Nachdem das Client-SDK eine Verbindung hergestellt hat, ruft es die IP-Adressen aller zusätzlichen HSMs ab und führt Lastenausgleich und clientseitige Schlüsselsynchronisierung durch.

Um eine IP-Adresse für den Cluster zu erhalten

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.
4. Um die IP-Adresse abzurufen, wählen Sie auf der Registerkarte HSMs eine der IP-Adressen aus, die unter ENI-IP-Adresse aufgeführt sind.

Um eine IP-Adresse für ein HSM (CLI) zu erhalten

- Rufen Sie die IP-Adresse eines HSM mit dem describe-clusters Befehl von der CLI ab. In der Ausgabe des Befehls sind die IP-Adressen der HSMs die Werte von `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

Weitere Informationen zum Bootstrapping finden Sie unter Configure Tool.

So bootstrappen Sie das Client-SDK 5

PKCS #11 library

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 -a <HSM IP addresses>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" -a <HSM IP
 addresses>
```

OpenSSL Dynamic Engine

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn -a <HSM IP addresses>
```

JCE provider

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce -a <HSM IP addresses>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" -a <HSM IP addresses>
```

CloudHSM CLI

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IP addresses of the HSMs>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IP
 addresses of the HSMs>
```

> **Note**
>
> Sie können den `--cluster-id`-Parameter anstelle von `-a <HSM_IP_ADDRESSES>` verwenden. Informationen zu den Anforderungen für die Verwendung von `--cluster-id` finden Sie unter Configure-Tool für das Client-SDK 5.

So bootstrappen Sie das Client-SDK 3

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 3

- Verwenden Sie configure diese Option, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
sudo /opt/cloudhsm/bin/configure -a <IP address>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 3

- Wird verwendetconfigure, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe -a <HSM IP address>
```

Weitere Informationen zur -Konfiguration finden Sie unter ???.

Hinzufügen oder Entfernen von HSMs in einem Cluster AWS CloudHSM

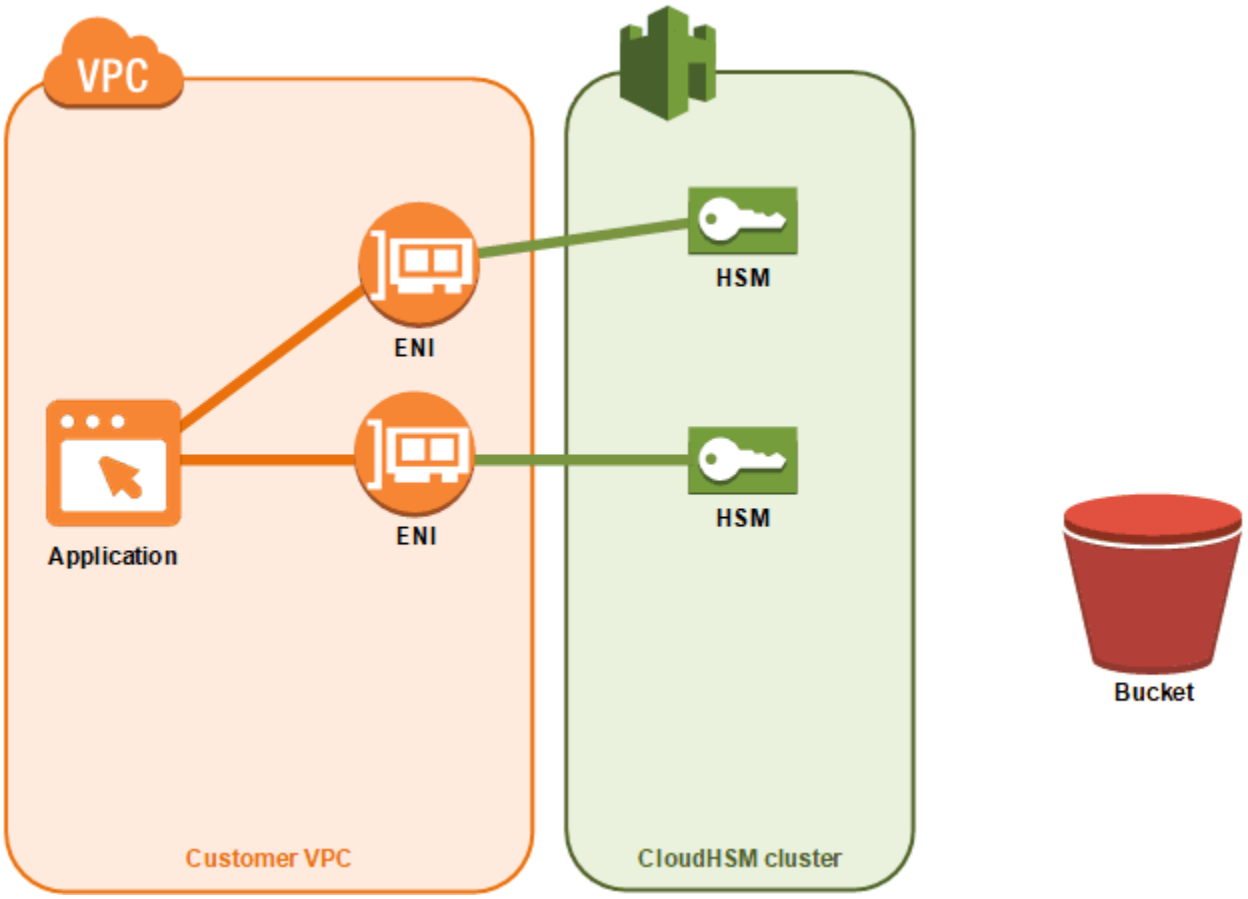
Um Ihren AWS CloudHSM Cluster nach oben oder unten zu skalieren, fügen Sie HSMs mithilfe der AWS CloudHSM Konsole oder eines der AWS SDKs oder Befehlszeilentools hinzu oder entfernen Sie sie. Wir empfehlen, Ihren Cluster einem Belastungstest zu unterziehen, um die zu erwartende Spitzenlast zu ermitteln, und dann ein weiteres HSM hinzuzufügen, um eine hohe Verfügbarkeit zu gewährleisten.

Themen

- Hinzufügen eines HSM
- Entfernen eines HSM

Hinzufügen eines HSM

Die folgende Abbildung zeigt die Ereignisse, die beim Hinzufügen eines HSM in einem Cluster auftreten.



1. Sie fügen einem Cluster ein neues HSM hinzu. In den folgenden Verfahren wird erklärt, wie Sie dies von der AWS CloudHSM Konsole, der AWS Command Line Interface (CLI) und der AWS CloudHSM API aus tun.

   Dies ist die einzige Aktion, die Sie ausführen. Die restlichen Ereignisse werden automatisch durchgeführt.
2. AWS CloudHSM erstellt eine Sicherungskopie eines vorhandenen HSM im Cluster. Weitere Informationen finden Sie unter Sicherungen.
3. AWS CloudHSM stellt das Backup auf dem neuen HSM wieder her. Auf diese Weise wird sichergestellt, dass das HSM mit den anderen im Cluster synchron ist.
4. Die vorhandenen HSMs im Cluster benachrichtigen den AWS CloudHSM Client, dass es ein neues HSM im Cluster gibt.
5. Der Client stellt eine Verbindung mit dem neuen HSM her.

So fügen Sie ein HSM hinzu (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie einen Cluster für das HSM aus, das Sie hinzufügen möchten.
3. Wählen Sie auf der Registerkarte HSMs die Option HSM erstellen aus.
4. Wählen Sie eine Availability Zone (AZ) für das HSM, das Sie erstellen. Wählen Sie die Option Erstellen aus.

Um ein HSM (CLI) hinzuzufügen

- Geben Sie an der Eingabeaufforderung den Befehl create-hsm samt einer Cluster-ID sowie einer Availability Zone für das HSM ein, das Sie erstellen. Wenn Sie die Cluster-ID des bevorzugten Clusters nicht kennen, verwenden Sie den describe-clusters-Befehl. Geben Sie die Availability Zone in der Form `us-east-2a`, `us-east-2b`, usw. an.

```
$ aws cloudhsmv2 create-hsm --cluster-id <cluster ID> --availability-
zone <Availability Zone>
{
    "Hsm": {
        "State": "CREATE_IN_PROGRESS",
        "ClusterId": "cluster-5a73d5qzrdh",
        "HsmId": "hsm-lgavqitns2a",
        "SubnetId": "subnet-0e358c43",
        "AvailabilityZone": "us-east-2c",
        "EniId": "eni-bab18892",
        "EniIp": "10.0.3.10"
    }
}
```

Um ein HSM (AWS CloudHSM API) hinzuzufügen

- Senden Sie eine CreateHsm-Anforderung und geben Sie die Cluster-ID und eine Availability Zone für das HSM an, das Sie erstellen.

Entfernen eines HSM

Sie können ein HSM mithilfe der AWS CloudHSM Konsole, der CLI oder der AWS CloudHSM API entfernen.

So entfernen Sie ein HSM (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie den Cluster mit dem HSM aus, das Sie entfernen möchten.
3. Klicken Sie auf der Registerkarte HSMs auf das HSM, das Sie entfernen möchten. Wählen Sie dann HSM löschen aus.
4. Bestätigen Sie, dass Sie das HSM löschen möchten. Wählen Sie dann Löschen.

Um ein HSM (CLI) zu entfernen

- Geben Sie in der Eingabeaufforderung den Befehl delete-hsm ein. Geben Sie die ID des Clusters mit dem HSM, das Sie löschen möchten, und eine der folgenden HSM-IDs ein:
  - Die HSM-ID (`--hsm-id`)
  - Die HSM-IP-Adresse (`--eni-ip`)
  - Die Elastic-Network-Schnittstellen-ID des HSM (`--eni-id`)

  Wenn Sie die Werte für diese IDs nicht kennen, geben Sie den Befehl describe-clusters ein.

```
$ aws cloudhsmv2 delete-hsm --cluster-id <cluster ID> --eni-ip <HSM IP address>
{
    "HsmId": "hsm-lgavqitns2a"
}
```

Um ein HSM (AWS CloudHSM API) zu entfernen

- Senden Sie eine DeleteHsm-Anforderung und geben Sie die Cluster-ID und eine ID Zone für das HSM an, das Sie löschen möchten.

Löschen eines AWS CloudHSM Clusters

Bevor Sie einen Cluster löschen können, müssen Sie alle HSMs aus dem Cluster entfernen. Weitere Informationen finden Sie unter Entfernen eines HSM.

Nachdem Sie alle HSMs entfernt haben, können Sie einen Cluster mithilfe der AWS CloudHSM Konsole, der AWS Command Line Interface (CLI) oder der AWS CloudHSM API löschen.

So löschen Sie einen Cluster (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie den Cluster aus, den Sie löschen möchten. Wählen Sie dann Cluster löschen aus.
3. Bestätigen Sie, dass Sie den Cluster löschen möchten, und wählen Sie dann Löschen aus.

So löschen Sie einen Cluster (CLI)

- Geben Sie an der Eingabeaufforderung den Befehl delete-cluster ein und übergeben Sie die ID des Clusters, das Sie löschen möchten. Wenn Sie die Cluster-ID nicht kennen, geben Sie den Befehl describe-clusters ein.

```
$ aws cloudhsmv2 delete-cluster --cluster-id <cluster ID>
{
    "Cluster": {
        "Certificates": {
            "ClusterCertificate": "<certificate string>"
        },
        "SourceBackupId": "backup-rtq2dwi2gq6",
        "SecurityGroup": "sg-40399d28",
        "CreateTimestamp": 1504903546.035,
        "SubnetMapping": {
            "us-east-2a": "subnet-f1d6e798",
            "us-east-2c": "subnet-0e358c43",
            "us-east-2b": "subnet-40ed9d3b"
        },
        "ClusterId": "cluster-kdmrayrc7gi",
        "VpcId": "vpc-641d3c0d",
        "State": "DELETE_IN_PROGRESS",
        "HsmType": "hsm1.medium",
        "StateMessage": "The cluster is being deleted.",
        "Hsms": [],
        "BackupPolicy": "DEFAULT"
    }
}
```

So löschen Sie einen AWS CloudHSM -Cluster (API)

- Senden Sie eine DeleteCluster-Anforderung, wobei Sie die ID des Clusters angeben, den Sie löschen möchten.

AWS CloudHSM Cluster aus Backups erstellen

Gehen Sie wie in diesem Thema beschrieben vor, um einen AWS CloudHSM Cluster aus einer Sicherung wiederherzustellen. Ihr Cluster enthält die gleichen Benutzer, das gleiche Schlüsselmaterial, die gleichen Zertifikate, die gleiche Konfiguration und die gleichen Richtlinien wie bei der Sicherung. Weitere Informationen zur Verwaltung von Sicherungen finden Sie unter Sicherungen verwalten.

Cluster aus Sicherungen erstellen (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie Cluster erstellen.
3. Führen Sie im Abschnitt Cluster-Konfiguration Folgendes aus:
   a. Wählen Sie für die VPC eine VPC für den Cluster aus, den Sie erstellen möchten.
   b. Wählen Sie für die AZ(s) ein privates Subnetz für jede Availability Zone aus, die Sie dem Cluster hinzufügen möchten.
4. Gehen Sie im Abschnitt Cluster source (Cluster-Quelle) wie folgt vor:
   a. Wählen Sie Cluster von einer früheren Sicherung wiederherstellen aus.
   b. Wählen Sie die Sicherung aus, die Sie wiederherstellen möchten.
5. Wählen Sie Weiter: Prüfen aus.
6. Überprüfen Sie Ihre Cluster-Konfiguration, dann wählen Sie Cluster erstellen aus.
7. Geben Sie an, wie lange der Dienst Sicherungen aufbewahren soll.

   Akzeptieren Sie den standardmäßigen Aufbewahrungszeitraum von 90 Tagen oder geben Sie einen neuen Wert zwischen 7 und 379 Tagen ein. Der Dienst löscht automatisch Sicherungen in diesem Cluster, die älter sind als der hier angegebene Wert. Sie können dies später ändern. Weitere Informationen finden Sie unter Konfiguration der Aufbewahrung für Sicherungen.
8. Wählen Sie Weiter.
9. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Tag hinzufügen, wenn Sie mehr als ein Tag zum Cluster hinzufügen möchten.
10. Wählen Sie Überprüfen.
11. Überprüfen Sie Ihre Cluster-Konfiguration und wählen Sie dann Cluster erstellen aus.

> **Tip**
>
> Um in diesem Cluster ein HSM zu erstellen, das dieselben Benutzer, dasselbe Schlüsselmaterial, dieselben Zertifikate, dieselbe Konfiguration und dieselben Richtlinien enthält wie das Backup, das Sie wiederhergestellt haben, fügen Sie dem Cluster ein HSM hinzu.

Cluster aus Backups erstellen (CLI)

Geben Sie den describe-backups-Befehl ein, um die Sicherungs-ID zu ermitteln.

- Geben Sie in der Eingabeaufforderung den Befehl create-cluster ein. Geben Sie den HSM-Instance-Typ, die Subnetz-IDs der Subnetze, auf denen Sie die HSMs erstellen möchten, und die Sicherungs-ID der Sicherung ein, die Sie wiederherstellen möchten.

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm1.medium \
                                --subnet-ids <subnet ID 1> <subnet ID 2> <subnet ID
 N> \
                                --source-backup-id <backup ID>
{
    "Cluster": {
        "HsmType": "hsm1.medium",
        "VpcId": "vpc-641d3c0d",
        "Hsms": [],
        "State": "CREATE_IN_PROGRESS",
        "SourceBackupId": "backup-rtq2dwi2gq6",
        "BackupPolicy": "DEFAULT",
        "BackupRetentionPolicy": {
            "Type": "DAYS",
            "Value": 90
         },
        "SecurityGroup": "sg-640fab0c",
        "CreateTimestamp": 1504907311.112,
        "SubnetMapping": {
            "us-east-2c": "subnet-0e358c43",
            "us-east-2a": "subnet-f1d6e798",
            "us-east-2b": "subnet-40ed9d3b"
        },
        "Certificates": {
            "ClusterCertificate": "<certificate string>"
```

```
        },
        "ClusterId": "cluster-jxhlf7644ne"
    }
}
```

Cluster aus Backups erstellen (AWS CloudHSM API)

Im folgenden Thema erfahren Sie, wie Sie mithilfe der API Cluster aus Sicherungen erstellen.

- CreateCluster

AWS CloudHSM Backups verwalten

AWS CloudHSM erstellt regelmäßig Backups Ihres Clusters, mindestens einmal alle 24 Stunden. Jede Sicherung enthält verschlüsselte Kopien der folgenden Daten:

- Benutzer (COs, CUs und AUs)
- Schlüsselmaterial und Zertifikate
- Hardware Security Module (HSM)-Konfiguration und -Richtlinien

Sie können den Dienst nicht anweisen, Sicherungen zu erstellen, aber Sie können bestimmte Aktionen ergreifen, die den Dienst dazu zwingen, eine Sicherung zu erstellen. Der Dienst erstellt eine Sicherung, wenn Sie eine der folgenden Aktionen ausführen:

- Aktivieren eines Clusters
- Fügen Sie einem aktiven Cluster ein HSM hinzu
- Entfernen Sie ein HSM aus einem aktiven Cluster

AWS CloudHSM löscht Backups auf der Grundlage der Backup-Aufbewahrungsrichtlinie, die Sie bei der Erstellung von Clustern festgelegt haben. Informationen zur Verwaltung der Aufbewahrungsrichtlinie für Sicherungen finden Sie unter Konfiguration der Aufbewahrung für Sicherungen.

Themen

Arbeiten mit Backups

Wenn Sie ein HSM zu einem Cluster hinzufügen, der zuvor ein oder mehrere aktive HSMs enthielt, stellt der Dienst die letzte Sicherung auf dem neuen HSM wieder her. Verwenden Sie Sicherungen, um HSMs zu verwalten, die Sie selten verwenden. Wenn Sie das HSM nicht mehr benötigen, löschen Sie es, um eine Sicherung zu erstellen. Wenn Sie das HSM später benötigen, erstellen Sie ein

neues im selben Cluster. Mit dieser Aktion wird das Backup wiederhergestellt, das Sie zuvor mit dem Vorgang „HSM löschen“ erstellt haben.

Abgelaufene Schlüssel oder inaktive Benutzer werden entfernt

Möglicherweise möchten Sie unerwünschtes kryptografisches Material wie abgelaufene Schlüssel oder inaktive Benutzer aus Ihrer Umgebung entfernen. Dies ist ein zweistufiger Prozess. Löschen Sie zunächst diese Materialien aus Ihrem HSM. Löschen Sie anschließend alle vorhandenen Sicherungen. Wenn Sie diesen Vorgang befolgen, stellen Sie sicher, dass Sie gelöschte Informationen nicht wiederherstellen, wenn Sie einen neuen Cluster aus einem Backup initialisieren. Weitere Informationen finden Sie unter the section called “Löschen und Wiederherstellen von Sicherungen”.

Notfallwiederherstellung erwägen

Sie können einen Cluster aus einer Sicherung erstellen. Möglicherweise möchten Sie dies tun, um einen Erholungspunkt für Ihren Cluster festzulegen. Nominieren Sie eine Sicherung, die alle Benutzer, das Schlüsselmaterial und die Zertifikate enthält, die Sie für Ihren Recovery Point benötigen, und verwenden Sie dann diese Sicherung, um einen neuen Cluster zu erstellen. Weitere Informationen zur Erstellung eines Clusters aus einer Sicherung finden Sie unter Cluster aus Sicherungen erstellen.

Sie können auch eine Sicherung eines Clusters in eine andere Region kopieren, in der Sie einen neuen Cluster als Klon des Originals erstellen können. Diese Lösung bietet sich für eine Reihe von Gründen an, einschließlich der Vereinfachung des Notfallwiederherstellungsprozesses. Weitere Informationen zum Kopieren von Sicherungen in Regionen finden Sie unter Regionsübergreifendes Kopieren von Sicherungen.

Löschen und Wiederherstellen von Sicherungen

Nachdem Sie eine Sicherung gelöscht haben, speichert der Dienst die Sicherung sieben Tage lang. Während dieser Zeit können Sie die Sicherung wiederherstellen. Nach Ablauf der sieben Tage können Sie das Backup nicht mehr wiederherstellen. Weitere Informationen zur Verwaltung von Sicherungen finden Sie unter Sicherungen verwalten.

Sicherungen löschen und wiederherstellen (Konsole)

So löschen Sie eine Sicherung (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Sicherungen aus.
4. Wählen Sie eine Sicherung zum Löschen aus.
5. Um die ausgewählte Sicherung zu löschen, wählen Sie Aktionen, Löschen.

   Das Dialogfeld „Sicherungen löschen“ wird angezeigt.
6. Wählen Sie Löschen aus.

   Der Status der Sicherung ändert sich zu `PENDING_DELETE`. Sie können eine Sicherung, dessen Löschung noch aussteht, bis zu 7 Tage lang wiederherstellen, nachdem Sie den Löschvorgang beantragt haben.

So stellen Sie eine Sicherung wieder her (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Sicherungen aus.
4. Wählen Sie eine Sicherung in dem `PENDING_DELETE`-Zustand aus, den Sie wiederherstellen möchten.
5. Um das ausgewählte Backup wiederherzustellen, wählen Sie Aktionen, Wiederherstellen.

Backups löschen und wiederherstellen (CLI)

Überprüfen Sie den Status eines Backups oder ermitteln Sie dessen ID, indem Sie den describe-backups Befehl aus der CLI verwenden.

Um ein Backup zu löschen (CLI)

- Führen Sie an der Eingabeaufforderung den Befehl delete-backup aus und übergeben Sie dabei die ID der Sicherung, die gelöscht werden soll.

```
$ aws cloudhsmv2 delete-backup --backup-id <backup ID>
{
    "Backup": {
        "CreateTimestamp": 1534461854.64,
        "ClusterId": "cluster-dygnwhmscg5",
        "BackupId": "backup-ro5c4er4aac",
        "BackupState": "PENDING_DELETION",
        "DeleteTimestamp": 1536339805.522
    }
}
```

So stellen Sie ein Backup wieder her (CLI)

- Um eine Sicherung wiederherzustellen, geben Sie den restore-backup Befehl aus und übergeben Sie die ID einer Sicherung mit dem Status `PENDING_DELETION`.

```
$ aws cloudhsmv2 restore-backup --backup-id <backup ID>
{
    "Backup": {
        "ClusterId": "cluster-dygnwhmscg5",
        "CreateTimestamp": 1534461854.64,
        "BackupState": "READY",
        "BackupId": "backup-ro5c4er4aac"
    }
}
```

Um Backups aufzulisten (CLI)

- Wenn Sie eine Liste aller Sicherungen mit dem Status `PENDING_DELETION` anzeigen wollen, führen Sie den Befehl describe-backups aus und schließen Sie als Filter `states=PENDING_DELETION` ein.

```
$ aws cloudhsmv2 describe-backups --filters states=PENDING_DELETION
{
```

```
    "Backups": [
        {
            "BackupId": "backup-ro5c4er4aac",
            "BackupState": "PENDING_DELETION",
            "CreateTimestamp": 1534461854.64,
            "ClusterId": "cluster-dygnwhmscg5",
            "DeleteTimestap": 1536339805.522,
        }
}
```

Backups löschen und wiederherstellen (AWS CloudHSM API)

In den folgenden Themen erfahren Sie, wie Sie Sicherungen mit Hilfe der API löschen und wiederherstellen können.

- DeleteBackup
- RestoreBackup

Konfiguration der Aufbewahrungsrichtlinie für AWS CloudHSM Backups

Mit Ausnahme von Clustern, die vor dem 18. November 2020 erstellt wurden, beträgt die Standardrichtlinie für die Aufbewahrung von Sicherungen für Cluster 90 Tage. Sie können für diesen Zeitraum eine beliebige Zahl zwischen 7 und 379 Tagen festlegen. AWS CloudHSM löscht das letzte Backup eines Clusters nicht. Weitere Informationen zur Verwaltung von Sicherungen finden Sie unter Sicherungen verwalten.

Grundlegendes zur Richtlinie zur Aufbewahrung von Sicherungen

AWS CloudHSM löscht Backups auf der Grundlage der Backup-Aufbewahrungsrichtlinie, die Sie bei der Erstellung eines Clusters festgelegt haben. Die Aufbewahrungsrichtlinie für Sicherungen gilt für Cluster. Wenn Sie eine Sicherung in eine andere Region verschieben, ist diese Sicherung nicht mehr mit einem Cluster verknüpft und es gibt keine Aufbewahrungsrichtlinie für Sicherungen. Sie müssen alle Backups, die nicht mit einem Cluster verknüpft sind, manuell löschen. AWS CloudHSM löscht das letzte Backup eines Clusters nicht.

AWS CloudTrail meldet Sicherungen, die zum Löschen markiert sind. Sie können Sicherungen, die der Service löscht, genauso wiederherstellen, wie Sie manuell gelöschte Sicherungen

wiederherstellen würden. Um zu verhindern, dass es zu einem Race Condition kommt, sollten Sie die Aufbewahrungsrichtlinie für Sicherungen für den Cluster ändern, bevor Sie eine vom Service gelöschte Sicherung wiederherstellen. Wenn Sie die Aufbewahrungsrichtlinie beibehalten und ausgewählte Sicherungen beibehalten möchten, können Sie angeben, dass der Service Sicherungen von der Aufbewahrungsrichtlinie für Cluster-Sicherungen ausschließt.

Ausnahme für existierende Cluster

AWS CloudHSM hat am 18. November 2020 die verwaltete Aufbewahrung von Backups eingeführt. Für Cluster, die vor dem 18. November 2020 erstellt wurden, gilt eine Richtlinie zur Aufbewahrung von Sicherungen von 90 Tagen zuzüglich des Alters des Clusters. Wenn Sie beispielsweise am 18. November 2019 einen Cluster erstellt haben, würde der Service Ihrem Cluster eine Backup-Aufbewahrungsrichtlinie von einem Jahr plus 90 Tagen (455 Tagen) zuweisen.

> **Note**
>
> Sie können die verwaltete Aufbewahrung von Sicherungen ganz deaktivieren, indem Sie sich an den Support wenden (https://aws.amazon.com/support).

Konfigurieren Sie die Aufbewahrung von Sicherungen (Konsole)

So konfigurieren Sie die Aufbewahrungsrichtlinie für Sicherungen (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Klicken Sie auf die Cluster-ID eines Clusters im Status Aktiv, um die Sicherungsaufbewahrungsrichtlinie für diesen Cluster zu verwalten.
4. Um die Aufbewahrungsrichtlinie für Sicherungen zu ändern, wählen Sie Aktionen, Aufbewahrungszeitraum für Sicherungen ändern.

   Daraufhin wird das Dialogfeld Sicherungsdauer ändern angezeigt.
5. Geben Sie im Feld Aufbewahrungszeitraum für Sicherungen (in Tagen) einen Wert zwischen 7 und 379 Tagen ein.
6. Wählen Sie Aufbewahrungszeitraum für Sicherungen ändern aus.

Um eine Sicherung von der Aufbewahrungsrichtlinie für Sicherungen auszuschließen oder aufzunehmen (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um Ihre Sicherungen anzusehen, wählen Sie im Navigationsbereich Sicherungen.
3. Klicken Sie auf die Sicherungs-ID eines Sicherungen im Status Bereit, um es auszuschließen oder einzubeziehen.
4. Führen Sie auf der Seite Sicherungsdetail eine der folgenden Aktionen aus.
   - Um eine Sicherung mit einem Datum als Ablaufzeit auszuschließen, wählen Sie Aktionen, Ablauf deaktivieren.
   - Um eine Sicherung einzubeziehen, das nicht abläuft, wählen Sie Aktionen, Cluster-Aufbewahrungsrichtlinie verwenden aus.

Backup-Aufbewahrung (CLI) konfigurieren

Überprüfen Sie den Status eines Backups oder ermitteln Sie dessen ID, indem Sie den describe-backups Befehl aus der CLI verwenden.

So konfigurieren Sie die Backup-Aufbewahrungsrichtlinie (CLI)

- Geben Sie in der Eingabeaufforderung den Befehl modify-cluster ein. Geben Sie die Cluster-ID und die Aufbewahrungsrichtlinie für Sicherungen an.

```
$ aws cloudhsmv2 modify-cluster --cluster-id <cluster ID> \
                                --backup-retention-policy  Type=DAYS,Value=<number
 of days to retain backups>
{
   "Cluster": {
      "BackupPolicy": "DEFAULT",
      "BackupRetentionPolicy": {
         "Type": "DAYS",
         "Value": 90
      },
      "Certificates": {},
      "ClusterId": "cluster-kdmrayrc7gi",
      "CreateTimestamp": 1504903546.035,
      "Hsms": [],
      "HsmType": "hsm1.medium",
      "SecurityGroup": "sg-40399d28",
```

```
      "State": "ACTIVE",
      "SubnetMapping": {
         "us-east-2a": "subnet-f1d6e798",
         "us-east-2c": "subnet-0e358c43",
         "us-east-2b": "subnet-40ed9d3b"
      },
      "TagList": [
         {
            "Key": "Cost Center",
            "Value": "12345"
         }
      ],
      "VpcId": "vpc-641d3c0d"
   }
}
```

So schließen Sie ein Backup von der Backup-Aufbewahrungsrichtlinie (CLI) aus

- Geben Sie in der Eingabeaufforderung den Befehl modify-backup-attributes ein. Geben Sie die Sicherungs-ID an und setzen Sie das Never-Ablauf-Flag, um die Sicherung beizubehalten.

```
$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
                                          --never-expires
{
   "Backup": {
      "BackupId": "backup-ro5c4er4aac",
      "BackupState": "READY",
      "ClusterId": "cluster-dygnwhmscg5",
      "NeverExpires": true
   }
}
```

Um ein Backup in die Backup-Aufbewahrungsrichtlinie (CLI) aufzunehmen

- Geben Sie in der Eingabeaufforderung den Befehl modify-backup-attributes ein. Geben Sie die Backup-ID an und setzen Sie das no-never-expires Kennzeichen so, dass das Backup in die Backup-Aufbewahrungsrichtlinie aufgenommen wird, was bedeutet, dass der Service das Backup irgendwann löscht.

```
$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
```

```
                                          --no-never-expires
{
   "Backup": {
      "BackupId": "backup-ro5c4er4aac",
      "BackupState": "READY",
      "ClusterId": "cluster-dygnwhmscg5",
      "NeverExpires": false
   }
}
```

Konfigurieren Sie die Aufbewahrung von Backups (AWS CloudHSM API)

In den folgenden Themen erfahren Sie, wie Sie die Aufbewahrung von Sicherungen mithilfe der API verwalten.

- ModifyCluster
- ModifyBackupAttributes

Kopieren von Backups zwischen AWS Regionen

Sie können Sicherungen aus vielen Gründen regionsübergreifend kopieren, z. B. für regionsübergreifende Ausfallsicherheit, globale Workloads und Notfallwiederherstellung. Nachdem Sie Sicherungen kopiert haben, werden sie in der Zielregion mit einem `CREATE_IN_PROGRESS`-Status angezeigt. Nach erfolgreichem Abschluss des Kopiervorgangs ändert sich der Status der Sicherung in `READY`. Wenn die Kopie fehlschlägt, ändert sich der Status der Sicherung auf `DELETED`. Überprüfen Sie Ihre Eingabeparameter auf Fehler und stellen Sie sicher, dass sich die angegebene Quellsicherung nicht in einem `DELETED`-Status befindet, bevor Sie den Vorgang erneut ausführen. Informationen über Sicherungen oder wie Sie einen Cluster aus einer Sicherung erstellen, finden Sie unter Sicherungen verwalten oder Cluster aus Sicherungen erstellen.

Beachten Sie Folgendes:

- Um eine Cluster-Sicherung in eine Zielregion zu kopieren, muss Ihr Konto über die entsprechenden IAM-Richtlinienberechtigungen verfügen. Damit Sie die Sicherung in eine andere Region kopieren können, muss Ihre IAM-Richtlinie den Zugriff auf die Quellregion erlauben, in der sich die Sicherung befindet. Sobald regionsübergreifend kopiert wurde, muss Ihre IAM-Richtlinie den Zugriff auf die Zielregion erlauben, um mit der kopierten Sicherung zu interagieren, welches die

Verwendung der Operation CreateCluster einschließt. Weitere Informationen finden Sie unter Erstellen von IAM-Administratoren.

- Der ursprüngliche Cluster und der Cluster, der möglicherweise aus einer Sicherung in der Zielregion erstellt wird, werden nicht verknüpft. Sie müssen jeden dieser Cluster unabhängig voneinander verwalten. Weitere Informationen finden Sie unter Verwalten von -Clustern.
- Backups können nicht zwischen AWS eingeschränkten Regionen und Standardregionen kopiert werden. Backups können zwischen den Regionen AWS GovCloud (US-Ost) und AWS GovCloud (US-West) kopiert werden.

Sicherungen in verschiedene Regionen kopieren (Konsole)

Um Sicherungen in verschiedene Regionen zu kopieren (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Wählen Sie im Navigationsbereich Sicherungen aus.
4. Wählen Sie eine Sicherung zum Kopieren in eine andere Region.
5. Um die ausgewählte Sicherung zu kopieren, wählen Sie Aktionen, Sicherung in eine andere Region kopieren.

   Das Dialogfenster „Sicherung in eine andere Region kopieren“ wird angezeigt.
6. Wählen Sie im Feld Zielregion eine Region von Eine Region auswählen aus.
7. (Optional) Geben Sie einen Tag-Schlüssel und einen optionalen Tag-Wert ein. Wählen Sie Tag hinzufügen, wenn Sie mehr als ein Tag zum Cluster hinzufügen möchten.
8. Klicken Sie auf Copy backup (Backup kopieren).

Backups in verschiedene Regionen kopieren (CLI)

Um die Sicherungs-ID zu ermitteln, führen Sie den describe-backups-Befehl aus.

Um Backups in verschiedene Regionen zu kopieren (CLI)

- Führen Sie über die Eingabeaufforderung den folgenden copy-backup-to-region-Befehl aus. Geben Sie die Zielregion und die Sicherungs-ID der Sicherung an, von der die Sicherung stammt. Wenn Sie eine Sicherungs-ID angeben, wird die zugehörige Sicherung kopiert.

```
$ aws cloudhsmv2 copy-backup-to-region --destination-region <destination region> \
                                       --backup-id <backup ID>
```

Backups in verschiedene Regionen kopieren (AWS CloudHSM API)

Im folgenden Thema erfahren Sie, wie Sie Sicherungen mithilfe der API in verschiedene Regionen kopieren.

- CopyBackupToRegion

Ressourcen taggen AWS CloudHSM

Ein Tag ist eine Bezeichnung, die Sie einer AWS Ressource zuweisen. Sie können Ihren AWS CloudHSM -Clustern Tags zuweisen. Jedes Tag besteht aus einem Tag-Schlüssel und einem Tag-Wert, die Sie beide selbst definieren können. Der Tag-Schlüssel kann beispielsweise Kostenstelle und der Tag-Wert 12345 sein. Tag-Schlüssel müssen für jeden Cluster eindeutig sein.

Sie können Tags für eine Vielzahl von Zwecken verwenden. Sie können damit Ihre AWS -Kosten kategorisieren und verfolgen. Sie können Tags anwenden, die geschäftliche Kategorien (wie Kostenstellen, Anwendungsnamen oder Eigentümer) darstellen, um die Kosten für mehrere Services zu organisieren. Wenn Sie Ihren AWS Ressourcen Tags hinzufügen, AWS wird ein Kostenzuordnungsbericht generiert, in dem die Nutzung und die Kosten nach Tags zusammengefasst sind. Sie können diesen Bericht verwenden, um Ihre AWS CloudHSM Kosten in Form von Projekten oder Anwendungen anzuzeigen, anstatt alle AWS CloudHSM Kosten als einzelne Position anzuzeigen.

Weitere Informationen zur Verwendung von Tags für die Kostenzuordnung finden Sie unter Verwenden von Kostenzuordnungs-Tags im AWS Billing -Benutzerhandbuch.

Sie können die AWS CloudHSM -Konsole, eines der AWS SDKs oder die Befehlszeilen-Tools verwenden, um Tags hinzuzufügen, zu aktualisieren, aufzulisten und zu entfernen.

Themen

Hinzufügen oder Aktualisieren von Tags

Sie können Tags über die AWS CloudHSM Konsole, die AWS Command Line Interface (CLI) oder die AWS CloudHSM API hinzufügen oder aktualisieren.

So fügen Sie Tags hinzu oder aktualisieren sie (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie den Cluster aus, den Sie mit Tags versehen möchten.

3. Wählen Sie Tags aus.
4. Führen Sie die folgenden Schritte aus, um ein Tag hinzuzufügen:
   a. Wählen Sie Tag bearbeiten und dann Tag hinzufügenaus.
   b. Geben Sie unter Key (Schlüssel) einen Schlüssel für das Tag ein.
   c. (Optional) Geben Sie unter Value (Wert) einen Wert für das Tag ein.
   d. Wählen Sie Speichern.
5. Führen Sie die folgenden Schritte aus, um ein Tag zu aktualisieren:
   a. Wählen Sie Tag bearbeiten.

      > **Note**
      >
      > Wenn Sie den Tag-Schlüssel für ein vorhandenes Tag aktualisieren, löscht die Konsole das vorhandene Tag und erstellt ein neues.

   b. Geben Sie den neuen Tag-Wert ein.
   c. Wählen Sie Speichern.

Um Tags hinzuzufügen oder zu aktualisieren (CLI)

1. Geben Sie an der Eingabeaufforderung den Befehl tag-resource unter Angabe der Tags und der ID des Clusters ein, den Sie mit Tags versehen möchten. Wenn Sie die Cluster-ID nicht kennen, geben Sie den Befehl describe-clusters ein.

```
$ aws cloudhsmv2 tag-resource --resource-id <cluster ID> \
                              --tag-list Key="<tag key>",Value="<tag value>"
```

2. Verwenden Sie zum Aktualisieren von Tags den gleichen Befehl, geben Sie jedoch einen vorhandenen Tag-Schlüssel an. Wenn Sie einen neuen Tag-Wert für ein vorhandenes Tag angeben, wird das Tag mit dem neuen Wert überschrieben.

Um Tags hinzuzufügen oder zu aktualisieren (AWS CloudHSM API)

- Senden Sie eine TagResource-Anforderung. Geben Sie die Tags und die ID des Clusters an, den Sie mit Tags versehen möchten.

Auflisten von Tags

Sie können Tags für einen Cluster über die AWS CloudHSM Konsole, die CLI oder die AWS CloudHSM API auflisten.

So listen Sie Tags auf (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie den Cluster, dessen Tags Sie auflisten möchten.
3. Wählen Sie Tags aus.

Um Tags aufzulisten (CLI)

- Geben Sie an der Eingabeaufforderung den Befehl list-tags unter Angabe der ID des Clusters ein, dessen Tags Sie auflisten. Wenn Sie die Cluster-ID nicht kennen, geben Sie den Befehl describe-clusters ein.

```
$ aws cloudhsmv2 list-tags --resource-id <cluster ID>
{
    "TagList": [
        {
            "Key": "Cost Center",
            "Value": "12345"
        }
    ]
}
```

Um Tags aufzulisten (AWS CloudHSM API)

- Senden Sie eine ListTags-Anforderung, wobei Sie die ID des Clusters angeben, dessen Tags Sie auflisten.

Entfernen von Tags

Sie können Tags mithilfe der AWS CloudHSM Konsole, der CLI oder der AWS CloudHSM API aus einem Cluster entfernen.

So entfernen Sie Tags (Konsole)

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Wählen Sie den Cluster aus, dessen Tags Sie entfernen.
3. Wählen Sie Tags aus.
4. Wählen Sie für das Tag, das Sie entfernen möchten, Tag bearbeiten und dann Tag entfernen aus.
5. Wählen Sie Speichern.

Um Tags zu entfernen (CLI)

- Geben Sie in der Eingabeaufforderung den Befehl untag-resource unter Angabe der Tag-Schlüssel der zu entfernenden Tags und der ID des Clusters ein, dessen Tags Sie entfernen möchten. Wenn Sie die CLI verwenden, um Tags zu entfernen, geben Sie nur die Tag-Schlüssel an, nicht die Tag-Werte.

```
$ aws cloudhsmv2 untag-resource --resource-id <cluster ID> \
                                --tag-key-list "<tag key>"
```

Um Tags zu entfernen (AWS CloudHSM API)

- Senden Sie eine UntagResourceAnfrage in der AWS CloudHSM API und geben Sie die ID des Clusters und die Tags an, die Sie entfernen möchten.

Verwalten von HSM-Benutzern und -Schlüsseln in AWS CloudHSM

Bevor Sie Ihren AWS CloudHSM-Cluster für die Kryptoverarbeitung verwenden können, müssen Sie Benutzer und Schlüssel in den HSMs Ihres Clusters erstellen. Weitere Informationen zur Verwaltung von HSM-Benutzern und -Schlüsseln finden Sie in den folgenden Themen in AWS CloudHSM. Außerdem erfahren Sie, wie Sie die Quorum-Authentifizierung verwenden.

Themen

Verwalten von HSM-Benutzern in AWS CloudHSM

In AWS CloudHSM müssen Sie die Befehlszeilentools CloudHSM-CLI oder CloudHSM Management Utility (CMU) verwenden, um die Benutzer auf Ihrem HSM zu erstellen und zu verwalten. CloudHSM-CLI ist für die Verwendung mit der neuesten SDK-Versionsserie konzipiert, während die CMU für die Verwendung mit der vorherigen SDK-Versionsserie konzipiert ist.

Themen

HSM-Benutzer mit CloudHSM CLI verwalten

Verwenden Sie die Befehlszeilentools von CloudHSM CLI, um die Benutzer auf Ihrem HSM mit dem neuesten SDK zu erstellen und zu verwalten.

Themen

- Verwendung von CloudHSM CLI zur Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle)

Grundlegendes zu HSM-Benutzern

Für die meisten Operationen, die Sie auf dem HSM ausführen, sind die Anmeldeinformationen eines HSM-Benutzers erforderlich. Das HSM authentifiziert jeden HSM-Benutzer, und jeder HSM-Benutzer hat einen Typ, der bestimmt, welche Operationen Sie als dieser Benutzer auf dem HSM ausführen können.

> **Note**
>
> HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer mit den richtigen Anmeldeinformationen können HSMs erstellen, indem sie über die AWS-API mit Ressourcen interagieren. Nachdem das HSM erstellt wurde, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren.

Benutzertypen

- Nicht aktivierter Admin
- Admin.
- Crypto-Benutzer (Crypto User, CU)
- Appliance-Benutzer (Appliance User, AU)

Nicht aktivierter Admin

In der CloudHSM CLI ist der nicht aktivierte Administrator ein temporärer Benutzer, der nur auf dem ersten HSM in einem AWS CloudHSM -Cluster existiert, der noch nie aktiviert wurde. Um einen Cluster zu aktivieren, führen Sie den cluster activate-Befehl in der CloudHSM CLI aus. Nach der Ausführung dieses Befehls werden nicht aktivierte Administratoren aufgefordert, das Passwort zu ändern. Nach dem Ändern des Passworts wird der inaktivierte Administrator zum Administrator.

Admin.

In der CloudHSM CLI kann der Administrator Benutzerverwaltungsvorgänge durchführen. Beispielsweise können Sie Benutzer erstellen und löschen und Benutzerpasswörter ändern. Weitere Informationen über Admins finden Sie in der Tabelle der HSM-Benutzerberechtigungen.

Crypto-Benutzer (Crypto User, CU)

Ein Crypto-Benutzer (CU) kann die folgenden Schlüsselverwaltungs- und kryptografischen Vorgänge ausführen.

- Schlüsselverwaltung – Erstellen, Löschen, Freigeben, Importieren und Exportieren von kryptographischen Schlüsseln.
- Kryptografische Operationen – Verwenden Sie kryptographische Schlüssel für die Verschlüsselung, Entschlüsselung, Signatur, Verifizierung und mehr.

Weitere Informationen hierzu finden Sie unter Tabelle der HSM-Benutzerberechtigungen.

Appliance-Benutzer (Appliance User, AU)

Der Appliance-Benutzer (AU) kann Kloning- und Synchronisierungsvorgänge an den HSMs Ihres Clusters durchführen. AWS CloudHSM verwendet die AU, um die HSMs in einem Cluster zu synchronisieren. AWS CloudHSM Die AU ist auf allen HSMs vorhanden, die von bereitgestellt werden AWS CloudHSM, und verfügt über eingeschränkte Berechtigungen. Weitere Informationen hierzu finden Sie unter Tabelle der HSM-Benutzerberechtigungen.

AWS kann keine Operationen an Ihren HSMs ausführen. AWS kann Ihre Benutzer oder Schlüssel nicht anzeigen oder ändern und mit diesen Schlüsseln keine kryptografischen Operationen ausführen.

Tabelle der HSM-Benutzerberechtigungen

In der folgenden Tabelle sind HSM-Operationen aufgeführt, sortiert nach dem Typ des HSM-Benutzers oder der HSM-Sitzung, die den Vorgang ausführen kann.

	Admin.	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentifizierte Sitzungen
Grundlegende Cluster-Informationen erhalten[1]	 Ja	 Ja	 Ja	 Ja

	Admin.	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentifizierte Sitzungen
Das eigene Passwort ändern	 Ja	 Ja	 Ja	Nicht zutreffend
Das Passwort eines Benutzers ändern	 Ja	 Nein	 Nein	 Nein
Benutzer hinzufügen oder entfernen	 Ja	 Nein	 Nein	 Nein
Den Synchronisierungsstatus erhalten[2]	 Ja	 Ja	 Ja	 Nein
Maskierte Objekte extrahieren und einfügen[3]	 Ja	 Ja	 Ja	 Nein
Funktionen zur Schlüsselverwaltung[4]	 Nein	 Ja	 Nein	 Nein

	Admin.	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentifizierte Sitzungen
Verschlüsseln/ Entschlüsseln	 Nein	 Ja	 Nein	 Nein
Signieren/ Überprüfen	 Nein	 Ja	 Nein	 Nein
Generieren von Digests und HMACs	 Nein	 Ja	 Nein	 Nein

- [1] Zu den grundlegenden Informationen gehören u. a die Anzahl der HSMs im Cluster sowie IP-Adresse, Modell, Seriennummer, Geräte-ID, Firmware-ID usw. der einzelnen HSMs.
- [2] Der Benutzer kann verschiedene Digests (Hashes) erhalten, die den Schlüsseln auf dem HSM entsprechen. Eine Anwendung kann diese Digest-Gruppen vergleichen, um den Synchronisierungsstatus von HSMs in einem Cluster zu erfahren.
- [3] Maskierte Objekte sind Schlüssel, die verschlüsselt werden, bevor sie das HSM verlassen. Sie können außerhalb des HSM nicht entschlüsselt werden. Sie werden erst entschlüsselt, nachdem sie in ein HSM eingefügt wurden, das sich in demselben Cluster befindet wie das HSM, aus dem sie extrahiert wurden. Eine Anwendung kann maskierte Objekte extrahieren und einfügen, um die HSMs in einem Cluster zu synchronisieren.
- [4] Die Funktionen zur Schlüsselverwaltung umfassen unter anderem das Erstellen, Löschen, Verpacken, Entpacken und Ändern der Schlüsselattribute.

Verwenden von CloudHSM CLI zur Benutzerverwaltung

Dieses Thema enthält step-by-step Anweisungen zur Verwaltung von Benutzern des Hardware-Sicherheitsmoduls (HSM) mit der CloudHSM-CLI. Weitere Informationen zu CloudHSM-CLI- oder HSM-Benutzern finden Sie unter CloudHSM-CLI und Verwenden der CloudHSM CLI.

Sections

Grundlegendes zur HSM-Benutzerverwaltung mit CloudHSM CLI

Um HSM-Benutzer zu verwalten, müssen Sie sich mit dem Benutzernamen und dem Passwort eines Administrators beim HSM anmelden. Nur Administratoren können Benutzer verwalten. Das HSM enthält einen Standard-Admin namens admin. Sie haben das Passwort für admin festgelegt, wenn Sie das Cluster aktiviert haben.

Um die CloudHSM CLI zu verwenden, müssen Sie das Configure-Tool verwenden, um die lokale Konfiguration zu aktualisieren. Anweisungen zum Ausführen des Configure-Tools mit der CloudHSM CLI finden Sie unter Erste Schritte mit der CloudHSM-Befehlszeilenschnittstelle (CLI). Für den `-a`-Parameter müssen Sie die IP-Adresse eines HSM in Ihrem Cluster hinzufügen. Wenn Sie mehrere HSMs haben, können Sie eine beliebige IP-Adresse verwenden. Dadurch wird sichergestellt, dass CloudHSM CLI alle Änderungen, die Sie vornehmen, auf den gesamten Cluster übertragen kann. Denken Sie daran, dass die CloudHSM-CLI ihre lokale Datei verwendet, um Clusterinformationen zu verfolgen. Wenn sich der Cluster seit der letzten Verwendung von CloudHSM CLI von einem bestimmten Host aus geändert hat, müssen Sie diese Änderungen zur lokalen Konfigurationsdatei hinzufügen, die auf diesem Host gespeichert ist. Entfernen Sie niemals ein HSM, während Sie die CloudHSM-CLI verwenden.

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.

4. Um die IP-Adresse abzurufen, wählen Sie auf der Registerkarte HSMs eine der IP-Adressen aus, die unter ENI-IP-Adresse aufgeführt sind.

Um eine IP-Adresse für ein HSM (CLI) zu erhalten

- Rufen Sie die IP-Adresse eines HSM mit dem describe-clusters Befehl von der CLI ab. In der Ausgabe des Befehls sind die IP-Adressen der HSMs die Werte von `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

Laden Sie die CloudHSM CLI herunter

Die neueste Version von CloudHSM CLI ist für HSM-Benutzerverwaltungsaufgaben für Client-SDK 5 verfügbar. Um CloudHSM-CLI herunterzuladen und zu installieren, folgen Sie den Anweisungen unter CloudHSM CLI installieren und konfigurieren.

So verwalten Sie HSM-Benutzer mit CloudHSM-CLI

Dieser Abschnitt enthält grundlegende Befehle zur Verwaltung von HSM-Benutzern mit der CloudHSM CLI.

> **Note**
>
> Hinweis: CloudHSM CLI-Benutzerbefehle sind in der CloudHSM-CLI-Benutzerbefehlsreferenz aufgeführt

Themen

Erstellen eines Admins

Gehen Sie wie folgt vor, um einen Admin zu erstellen.

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Verwenden Sie den login-Befehl und melden Sie sich beim Cluster als Administrator an.

   ```
   aws-cloudhsm > login --username <USERNAME> --role admin
   ```

3. Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Geben Sie das Passwort ein und die Ausgabe zeigt, dass der Befehl erfolgreich war.

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

4. Geben Sie den folgenden Befehl ein, um einen Admin zu erstellen.

```
aws-cloudhsm > user create --username <USERNAME> --role admin
```

5. Geben Sie das Passwort für den neuen Benutzer ein.
6. Geben Sie das Passwort erneut ein, um zu bestätigen, dass das eingegebene Passwort korrekt ist.

: Erstellen eines Crypto-Benutzers

Führen Sie zur Benutzererstellung die folgenden Schritte aus.

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Verwenden Sie den login-Befehl und melden Sie sich beim Cluster als Administrator an.

   ```
   aws-cloudhsm > login --username <USERNAME> --role admin
   ```

3. Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Geben Sie das Passwort ein und die Ausgabe zeigt, dass der Befehl erfolgreich war.

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

4. Geben Sie den folgenden Befehl ein, um einen Crypto-Benutzer zu erstellen.

   ```
   aws-cloudhsm > user create --username <USERNAME> --role crypto-user
   ```

5. Geben Sie das Passwort für den neuen Crypto-Benutzer ein.
6. Geben Sie das Passwort erneut ein, um zu bestätigen, dass das eingegebene Passwort korrekt ist.

Um alle HSM-Benutzer im Cluster aufzulisten

Verwenden Sie den user list-Befehl, um alle Benutzer im Cluster aufzulisten. Sie müssen sich nicht anmelden, um user list auszuführen. Alle Benutzertypen können Benutzer auflisten.

Gehen Sie wie folgt vor, um alle Benutzer im Cluster aufzulisten

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Geben Sie den folgenden Befehl ein, um alle Benutzer im Cluster aufzulisten:

   ```
   aws-cloudhsm > user list
   ```

Für weitere Informationen über user list finden Sie in der Benutzerliste.

Um HSM-Benutzerpasswörter zu ändern

Verwenden Sie den user change-password-Befehl, um ein Passwort zu ändern.

Bei Benutzertypen und Passwörtern wird zwischen Groß- und Kleinschreibung unterschieden, bei Benutzernamen jedoch nicht.

Admin, Crypto-Benutzer (CU) und Appliance-Benutzer (AU) können ihr eigenes Passwort ändern. Um das Passwort eines anderen Benutzers zu ändern, müssen Sie sich als Administrator anmelden. Sie können das Passwort eines Benutzers, der gerade angemeldet ist, nicht ändern.

So ändern Sie Ihr eigenes Passwort

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Verwenden Sie den login-Befehl und melden Sie sich als Benutzer mit dem Passwort an, das Sie ändern möchten.

   ```
   aws-cloudhsm > login --username <USERNAME> --role <ROLE>
   ```

3. Geben Sie das Passwort des Benutzers ein.

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin1",
       "role": "admin"
     }
   }
   ```

4. Geben Sie den user change-password-Befehl ein.

   ```
   aws-cloudhsm > user change-password --username <USERNAME> --role <ROLE>
   ```

5. Geben Sie das neue Passwort ein.
6. Geben Sie das neue Passwort erneut ein.

So ändern Sie das Passwort eines anderen Benutzers

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. Verwenden Sie den login-Befehl und melden Sie sich als Administrator an.

```
aws-cloudhsm > login --username <USERNAME> --role admin
```

3. Geben Sie das Admin-Passwort ein.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

4. Geben Sie den user change-password-Befehl zusammen mit dem Benutzernamen des Benutzers ein, dessen Passwort Sie ändern möchten.

```
aws-cloudhsm > user change-password --username <USERNAME> --role <ROLE>
```

5. Geben Sie das neue Passwort ein.
6. Geben Sie das neue Passwort erneut ein.

Weitere Informationen über user change-password finden Sie unter user change-password.

So löschen Sie HSM-Benutzer

Verwenden Sie user delete, um einen Benutzer zu löschen. Sie müssen sich als Administrator anmelden, um einen anderen Benutzer zu löschen.

> **Tip**
>
> Sie können keine Crypto-Benutzer (CU) löschen, die Schlüssel besitzen.

Benutzer löschen

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Verwenden Sie den login-Befehl und melden Sie sich beim Cluster als Administrator an.

   ```
   aws-cloudhsm > login --username <USERNAME> --role admin
   ```

3. Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Geben Sie das Passwort ein und die Ausgabe zeigt, dass der Befehl erfolgreich war.

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

4. Verwenden Sie den Befehl user delete, um den Benutzer zu löschen.

   ```
   aws-cloudhsm > user delete --username <USERNAME> --role <ROLE>
   ```

Für weitere Informationen über user delete finden Sie unter deleteUser.

Verwenden von CloudHSM-CLI zur Verwaltung von MFA

Um die Sicherheit zu erhöhen, können Sie die Multi-Faktor-Authentifizierung (MFA) konfigurieren, um den Cluster zu schützen. Weitere Informationen finden Sie in den folgenden Themen.

Themen

MFA für HSM-Benutzer verstehen

Wenn Sie sich mit einem MFA-aktivierten HSM-Benutzerkonto bei einem Cluster anmelden, geben Sie der CloudHSM CLI Ihr Passwort – der erste Faktor, den Sie kennen – und die CloudHSM CLI stellt Ihnen ein Token zur Verfügung und fordert Sie auf, das Token zu signieren.

Um den zweiten Faktor bereitzustellen – was Sie haben – signieren Sie das Token mit einem privaten Schlüssel aus einem Schlüsselpaar, das Sie bereits erstellt und dem HSM-Benutzer zugeordnet haben. Um auf den Cluster zuzugreifen, stellen Sie der CloudHSM CLI das signierte Token zur Verfügung.

Weitere Informationen zur Einrichtung von MFA für einen Benutzer finden Sie unter MFA für CloudHSM-CLI einrichten.

Quorum-Authentifizierung und MFA

Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für MFA. Das bedeutet, dass ein Benutzer mit aktivierter MFA effektiv für die MofN- oder Quorum-Zugriffskontrolle registriert ist. Beachten Sie die folgenden Punkte, um MFA und die Quorum-Authentifizierung erfolgreich für denselben HSM-Benutzer zu verwenden:

- Wenn Sie heute die Quorum-Authentifizierung für einen Benutzer verwenden, sollten Sie dasselbe Schlüsselpaar verwenden, das Sie für den Quorumbenutzer erstellt haben, um MFA für den Benutzer zu aktivieren.
- Wenn Sie die MFA-Anforderung für einen Nicht-MFA-Benutzer hinzufügen, der kein Quorum-Authentifizierungsbenutzer ist, registrieren Sie diesen Benutzer als Quorum (MofN) registrierten Benutzer mit MFA-Authentifizierung.
- Wenn Sie die MFA-Anforderung aufheben oder das Passwort für einen MFA-Benutzer ändern, der auch ein registrierter Quorum-Authentifizierungsbenutzer ist, entfernen Sie auch die Registrierung des Benutzers als Quorum-Benutzer (MoFN).

- Wenn Sie die MFA-Anforderung aufheben oder das Passwort für einen MFA-Benutzer ändern, der auch ein Quorum-Authentifizierungsbenutzer ist, Sie aber trotzdem möchten, dass dieser Benutzer an der Quorum-Authentifizierung teilnimmt, müssen Sie diesen Benutzer erneut als Quorum-Benutzer (MofN) registrieren.

Weitere Informationen über die Quorum-Authentifizierung finden Sie unter Verwaltung des Quorums (M von N).

Arbeiten mit MFA für HSM-Benutzer

Dieses Thema enthält Informationen und Anweisungen zur Verwendung der CloudHSM-CLI zur Verwaltung der Multi-Faktor-Authentifizierung (MFA). Weitere Informationen über CloudHSM CLI finden Sie unter CloudHSM-Befehlszeilenschnittstelle (CLI).

Themen

- Anforderungen an das MFA-Schlüsselpaar
- MFA für CloudHSM-CLI einrichten
- Benutzer mit aktiviertem MFA erstellen
- Benutzer mit aktivierter MFA anmelden
- Schlüssel für Benutzer mit aktiviertem MFA rotieren
- Einen öffentlichen MFA-Schlüssel für Admin-Benutzer abmelden, wenn der öffentliche MFA-Schlüssel registriert ist
- Tokendatei-Referenz

Weitere Informationen zur Arbeit mit HSM-Benutzern finden Sie unter CloudHSM-Befehlszeilenschnittstelle (CLI).

Anforderungen an das MFA-Schlüsselpaar

Um MFA für einen HSM-Benutzer zu aktivieren, können Sie ein neues Schlüsselpaar erstellen oder einen vorhandenen Schlüssel verwenden, der die folgenden Anforderungen erfüllt:

- Schlüsseltyp: Asymmetrisch
- Schlüsselnutzung: Signieren und verifizieren
- Schlüsselspezifikation: RSA_2048
- Der Signierungsalgorithmus umfasst: SHA256 mit RSA-Verschlüsselung

> **Note**
>
> Wenn Sie die Quorumauthentifizierung verwenden oder planen, die Quorumauthentifizierung zu verwenden, finden Sie weitere Informationen unter Quorum-Authentifizierung und MFA.

Sie können die CloudHSM-CLI und das Schlüsselpaar verwenden, um einen neuen Admin-Benutzer mit aktiviertem MFA zu erstellen.

MFA für CloudHSM-CLI einrichten

Führen Sie diese Schritte aus, um MFA für CloudHSM CLI einzurichten.

1. Um MFA mithilfe der Token-Sign-Strategie einzurichten, müssen Sie zunächst einen privaten 2048-Bit-RSA-Schlüssel und den zugehörigen öffentlichen Schlüssel generieren.

```
$ openssl genrsa -out officer1.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................+++++
.....................................................................+++++
e is 65537 (0x010001)

$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
writing RSA key
```

2. Melden Sie sich mit der CloudHSM CLI bei Ihrem Benutzerkonto an.

```
$ cloudhsm-cli interactive
aws-cloudhsm > login --username admin --role admin --cluster-id <cluster ID>
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

3. Führen Sie als Nächstes den Befehl aus, um Ihre MFA-Strategie zu ändern. Sie müssen den Parameter `--token` angeben. Dieser Parameter gibt eine Datei an, in die unsignierte Token geschrieben werden.

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
username <USERNAME> --role crypto-user --change-quorum
Enter password:
Confirm password:
```

4. Sie haben jetzt eine Datei mit unsignierten Token, die signiert werden müssen: `unsigned-tokens.json`. Die Anzahl der Token in dieser Datei hängt von der Anzahl der HSMs in Ihrem Cluster ab. Jedes Token steht für ein HSM. Diese Datei ist im JSON-Format und enthält Token, die signiert werden müssen, um nachzuweisen, dass Sie über einen privaten Schlüssel verfügen.

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
{
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

5. Der nächste Schritt besteht darin, diese Token mit dem in Schritt 1 erstellten privaten Schlüssel zu signieren. Platzieren Sie die Signaturen wieder in der Datei. Zuerst müssen Sie die Base64-codierten Token extrahieren und dekodieren.

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

6. Jetzt haben Sie binäre Token, die Sie mit dem in Schritt 1 erstellten privaten RSA-Schlüssel signieren können.

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

7. Jetzt haben Sie binäre Signaturen der Token. Sie müssen sie mit Base64 codieren und sie wieder in Ihre Tokendatei einfügen.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

8. Schließlich können Sie die Base64-Werte kopieren und wieder in Ihre Token-Datei einfügen:

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
```

```
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

9. Jetzt, da Ihre Token-Datei alle erforderlichen Signaturen enthält, können Sie fortfahren. Geben Sie den Namen der Datei ein, die die signierten Token enthält, und drücken Sie die Eingabetaste. Geben Sie abschließend den Pfad Ihres öffentlichen Schlüssels ein.

```
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:officer1.pub
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "crypto-user"
  }
}
```

Jetzt haben Sie Ihren Benutzer mit MFA eingerichtet.

```
{
    "username": "<USERNAME>",
    "role": "crypto-user",
    "locked": "false",
```

```
    "mfa": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
},
```

Benutzer mit aktiviertem MFA erstellen

Gehen Sie wie folgt vor, um Benutzer mit aktiviertem MFA zu erstellen.

1. Verwenden Sie die CloudHSM CLI, um sich als Administrator beim HSM anzumelden.
2. Verwenden Sie den user create-Befehl, um einen Benutzer Ihrer Wahl zu erstellen. Folgen Sie dann den Schritten unter MFA für CloudHSM-CLI einrichten, um MFA für den Benutzer einzurichten.

Benutzer mit aktivierter MFA anmelden

Folgen Sie diesen Schritten, um Benutzer mit aktivierter MFA anzumelden.

1. Verwenden Sie den login mfa-token-sign-Befehl in der CloudHSM-CLI, um den Anmeldevorgang mit MFA für einen Benutzer zu starten, der MFA aktiviert hat.

```
aws-cloudhsm > login --username <USERNAME> --role <ROLE> mfa-token-sign --token
 unsigned-tokens.json
Enter password:
```

2. Geben Sie Ihr Passwort ein. Sie werden dann aufgefordert, den Pfad zur Tokendatei einzugeben, die unsignierte/signierte Tokenpaare enthält, wobei signierte Token diejenigen sind, die mithilfe Ihres privaten Schlüssels generiert wurden.

```
aws-cloudhsm > login --username <USERNAME> --role <ROLE> mfa-token-sign --token
 unsigned-tokens.json
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
```

3. Während Sie aufgefordert werden, den Pfad der signierten Token-Datei einzugeben, können Sie die Datei mit dem unsignierten Token in einem separaten Terminal überprüfen. Identifizieren

Sie die Datei mit unsignierten Token, die signiert werden müssen: `unsigned-tokens.json`. Die Anzahl der Token in dieser Datei hängt von der Anzahl der HSMs in Ihrem Cluster ab. Jedes Token steht für ein HSM. Diese Datei ist im JSON-Format und enthält Token, die signiert werden müssen, um nachzuweisen, dass Sie über einen privaten Schlüssel verfügen.

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. Signieren Sie die unsignierten Token mit dem in Schritt 2 erstellten privaten Schlüssel. Zuerst müssen Sie die Base64-codierten Token extrahieren und dekodieren.

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

5. Sie haben jetzt binäre Token. Signieren Sie sie mit dem privaten RSA-Schlüssel, den Sie zuvor in Schritt 1 der MFA-Einrichtung erstellt haben.

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
```

```
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

6. Sie haben jetzt binäre Signaturen der Token. Kodieren Sie sie mit Base64 und platzieren Sie sie wieder in Ihrer Tokendatei.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

7. Kopieren Sie schließlich die base64-Werte und fügen Sie sie wieder in Ihre Token-Datei ein:

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
```

```
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

8. Jetzt, da Ihre Token-Datei alle erforderlichen Signaturen enthält, können Sie fortfahren. Geben Sie den Namen der Datei ein, die die signierten Token enthält, und drücken Sie die Eingabetaste. Sie sollten sich jetzt erfolgreich anmelden.

```
aws-cloudhsm > login --username <USERNAME> --role <ROLE> mfa-token-sign --token
 unsigned-tokens.json
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "<ROLE>"
  }
}
```

Schlüssel für Benutzer mit aktiviertem MFA rotieren

Gehen Sie wie folgt vor, um Schlüssel für Benutzer mit aktiviertem MFA zu rotieren.

<result>

Sie haben die generierte Tokendatei im JSON-Format mit Ihrem privaten Schlüssel signiert und einen neuen öffentlichen MFA-Schlüssel registriert.
</result>

1. Verwenden Sie die CloudHSM-CLI, um sich als beliebiger Administrator oder als der spezifische Benutzer, für den MFA aktiviert ist, am HSM anzumelden (weitere Informationen finden Sie unter Benutzer mit aktiviertem MFA anmelden).

2. Führen Sie als Nächstes den Befehl aus, um Ihre MFA-Strategie zu ändern. Sie müssen den Parameter `--token` angeben. Dieser Parameter gibt eine Datei an, in die unsignierte Token geschrieben werden.

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
username <USERNAME> --role crypto-user --change-quorum
Enter password:
Confirm password:
```

3. Identifizieren Sie die Datei mit unsignierten Token, die signiert werden müssen: `unsigned-tokens.json`. Die Anzahl der Token in dieser Datei hängt von der Anzahl der HSMs in Ihrem Cluster ab. Jedes Token steht für ein HSM. Diese Datei ist im JSON-Format und enthält Token, die signiert werden müssen, um nachzuweisen, dass Sie über einen privaten Schlüssel verfügen. Dies ist der neue private Schlüssel aus dem neuen öffentlichen/privaten RSA-Schlüsselpaar, das Sie für die Rotation des aktuell registrierten öffentlichen Schlüssels verwenden möchten.

```
$cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. Signieren Sie diese Token mit dem privaten Schlüssel, den Sie zuvor bei der Einrichtung erstellt haben. Zuerst müssen wir die Base64-codierten Token extrahieren und dekodieren.

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
```

```
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

5. Sie haben jetzt binäre Token. Signieren Sie sie mit dem privaten RSA-Schlüssel, den Sie zuvor bei der Installation erstellt haben.

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

6. Sie haben jetzt binäre Signaturen der Token. Kodieren Sie sie mit Base64 und platzieren Sie sie wieder in Ihrer Tokendatei.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

7. Kopieren Sie schließlich die base64-Werte und fügen Sie sie wieder in Ihre Token-Datei ein:

```
{
  "version": "2.0",
  "tokens": [
    {
```

```
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

8. Jetzt, da Ihre Token-Datei alle erforderlichen Signaturen enthält, können Sie fortfahren. Geben Sie den Namen der Datei ein, die die signierten Token enthält, und drücken Sie die Eingabetaste. Geben Sie abschließend den Pfad Ihres neuen öffentlichen Schlüssels ein. Jetzt sehen Sie Folgendes als Teil der Ausgabe der Benutzerliste.

```
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:officer1.pub
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "crypto-user"
  }
}
```

Jetzt haben wir unseren Benutzer mit MFA eingerichtet.

```
{
    "username": "<USERNAME>",
    "role": "crypto-user",
    "locked": "false",
    "mfa": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
},
```

Einen öffentlichen MFA-Schlüssel für Admin-Benutzer abmelden, wenn der öffentliche MFA-Schlüssel registriert ist

Gehen Sie wie folgt vor, um einen öffentlichen MFA-Schlüssel für Admin-Benutzer abzumelden, wenn der öffentliche MFA-Schlüssel registriert ist.

1. Verwenden Sie die CloudHSM CLI, um sich als Administrator mit aktiviertem MFA beim HSM anzumelden.
2. Verwenden Sie den user change-mfa token-sign-Befehl, um MFA für einen Benutzer zu entfernen.

```
aws-cloudhsm >user change-mfa token-sign --username <USERNAME> --role admin --
deregister --change-quorum
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "admin"
  }
}
```

Tokendatei-Referenz

Die Tokendatei, die entweder bei der Registrierung eines öffentlichen MFA-Schlüssels oder beim Versuch, sich mit MFA anzumelden, generiert wird, besteht aus folgenden Elementen:

- Tokens: Ein Base64-kodiertes Array mit unsignierten/signierten Tokenpaaren in Form von JSON-Objektliteralen.
- Unsigniert: Ein Base64-kodiertes und ein SHA256-Hash-Token.
- Signiert: Ein Base64-codiertes signiertes Token (Signatur) des unsignierten Tokens unter Verwendung des privaten RSA-2048-Bit-Schlüssels.


```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/TK0PVaxLN42X
+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37+j/
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQdErI
    }
  ]
}
```

Verwendung von CloudHSM CLI zur Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle)

Die HSMs in Ihrem AWS CloudHSM-Cluster unterstützen die Quorum-Authentifizierung (auch M of N-Zugriffskontrolle genannt). Mit der Quorum-Authentifizierung kann kein einzelner Benutzer im HSM quorum-kontrollierte Operationen im HSM durchführen. Stattdessen muss eine Mindestanzahl von HSM-Benutzern (mindestens 2) zusammenarbeiten, um diese Operationen durchzuführen. Mit der Quorum-Authentifizierung können Sie eine zusätzliche Schutzebene hinzufügen, indem Sie Genehmigungen von mehr als einem HSM-Benutzer erfordern.

Die Quorum-Authentifizierung kann die folgenden Vorgänge steuern:

- HSM-Benutzerverwaltung durch den Administrator – Erstellen und Löschen von HSM-Benutzern und Ändern des Passworts eines anderen HSM-Benutzers. Weitere Informationen finden Sie unter Quorum-Authentifizierung für Administratoren verwenden.

Die folgenden Themen stellen weitere Informationen zur Quorum-Authentifizierung in AWS CloudHSM zur Verfügung.

Themen

Überblick über die Quorum-Authentifizierung mit Token-Sign-Strategie

Die folgenden Schritte fassen die Verfahren zur Quorum-Authentifizierung zusammen. Die exakten Schritte und Tools finden Sie unter Quorum-Authentifizierung für Administratoren verwenden.

1. Jeder HSM-Benutzer erstellt einen asymmetrischen Schlüssel zum Signieren. Benutzer tun dies außerhalb des HSMs und achtet darauf, den Schlüssel angemessen zu schützen.
2. Jeder HSM-Benutzer meldet sich beim HSM an und registriert den öffentlichen Teil seines Signierungsschlüssels (den öffentlichen Schlüssel) beim HSM.

3. Wenn ein HSM-Benutzer eine Quorum-kontrollierte Operation durchführen möchte, meldet sich derselbe Benutzer beim HSM an und erhält ein Quorum-Token.
4. Der HSM-Benutzer gibt das Quorum-Token an einen oder mehrere andere HSM-Benutzer weiter und bittet um deren Zustimmung.
5. Die anderen HSM-Benutzer genehmigen, indem sie mit ihren Schlüsseln das Quorum-Token kryptographisch signieren. Dies geschieht außerhalb des HSMs.
6. Wenn der HSM-Benutzer über die erforderliche Anzahl von Genehmigungen verfügt, meldet sich derselbe Benutzer beim HSM an und führt den quorumgesteuerten Vorgang mit dem --approval-Argument aus. Dabei wird die signierte Quorum-Tokendatei bereitgestellt, die alle erforderlichen Genehmigungen (Signaturen) enthält.
7. Das HSM verwendet die registrierten, öffentlichen Schlüssel jedes Unterzeichners, um die Signaturen zu verifizieren. Wenn die Signaturen gültig sind, genehmigt das HSM das Token und der quorumgesteuerte Vorgang wird ausgeführt.

Weitere Details zur Quorum-Authentifizierung

Beachten Sie die folgenden, zusätzlichen Informationen zur Verwendung der Quorum-Authentifizierung in AWS CloudHSM.

- Ein HSM-Benutzer kann sein eigenes Quorum-Token signieren, d. h. der anfordernde Benutzer kann eine der erforderlichen Genehmigungen für die Quorum-Authentifizierung bereitstellen.
- Sie wählen die Mindestzahl der Quorum-Genehmiger für die Quorum-kontrollierten Operationen aus. Die kleinste Zahl, die Sie wählen können, ist zwei (2), und die größte Zahl, die Sie wählen können, ist acht (8).
- Das HSM kann bis zu 1024 Quorum-Token speichern. Wenn das HSM beim Versuch, ein neues zu erstellen, bereits 1024 Token hat, löscht das HSM eines der abgelaufenen Token. Standardmäßig verfallen Token zehn Minuten nach ihrer Erstellung.
- Wenn MFA aktiviert ist, verwendet der Cluster denselben Schlüssel für die Quorum-Authentifizierung und für die Multi-Faktor-Authentifizierung (MFA). Weitere Informationen zur Verwendung von Quorum-Authentifizierung und 2FA finden Sie unter CloudHSM CLI zur Verwaltung von MFA verwenden.
- Jedes HSM kann jeweils nur ein Token pro Service enthalten.

Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Jeder Diensttyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Service type (Servicetyp)	Serviceoperationen
user	Admin.	• user create • user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Token-Signatur set-quorum-value

Quorum-Authentifizierung für Admins verwenden: erstmalige Einrichtung

In den folgenden Themen werden die Schritte beschrieben, die Sie durchführen müssen, um Ihr Hardware-Sicherheitsmodul (HSM) so zu konfigurieren, dass Administratoren die Quorum-Authentifizierung verwenden können. Sie müssen diese Schritte nur einmal ausführen, wenn Sie die Quorum-Authentifizierung für Admins zum ersten Mal konfigurieren. Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Quorum-Authentifizierung für Administratoren verwenden fort.

Themen

- Voraussetzungen
- Erstellen und Registrieren eines Schlüssels für das Signieren
- Setzten des Quorum-Mindestwerts für das HSM

Voraussetzungen

Um dieses Beispiel zu verstehen, sollten Sie mit der CloudHSM-CLI vertraut sein. In diesem Beispiel verfügt der AWS CloudHSM-Cluster über zwei HSMs, die jeweils die gleichen Admins aufweisen, wie

die folgende Ausgabe des Befehls user list zeigt. Weitere Informationen zum Erstellen von Benutzern finden Sie unter Verwenden der CloudHSM CLI.

```
aws-cloudhsm>user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin3",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin4",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
```

```
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

Erstellen und Registrieren eines Schlüssels für das Signieren

Um die Quorum-Authentifizierung zu verwenden, muss jeder Administrator alle der folgenden Schritte ausführen:

Themen

- Erstellen eines RSA-Schlüsselpaares
- Erstellen und signieren Sie ein Registrierungstoken
- Registrieren Sie den öffentlichen Schlüssel beim HSM

Erstellen eines RSA-Schlüsselpaares

Es gibt viele verschiedene Möglichkeiten, ein Schlüsselpaar zu erstellen und zu schützen. In den folgenden Beispielen wird gezeigt, wie dies mit OpenSSL durchgeführt wird.

Example – Erstellen eines privaten Schlüssels mit OpenSSL

Im folgenden Beispiel wird gezeigt, wie OpenSSL verwendet wird, um einen 2048-Bit-RSA-Schlüssel zu erstellen, der durch eine Pass-Phrase geschützt ist. Ersetzen Sie, um dieses Beispiel zu verwenden, *<admin.key>* durch den Namen der Datei, in der Sie den Schlüssel speichern möchten.

```
$ openssl genrsa -out <admin.key> -aes256 2048
Generating RSA private key, 2048 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
Enter pass phrase for admin.key:
Verifying - Enter pass phrase for admin.key:
```

Generieren eines öffentlichen Schlüssels mit dem privaten Schlüssel, den Sie gerade erstellt haben.

Example – Erstellen Sie einen öffentlichen Schlüssel mit OpenSSL

Das folgende Beispiel zeigt, wie Sie OpenSSL verwenden, um einen öffentlichen Schlüssel aus dem privaten Schlüssel zu erstellen, den Sie gerade erstellt haben.

```
$ openssl rsa -in admin.key -outform PEM -pubout -out admin1.pub
Enter pass phrase for admin.key:
writing RSA key
```

Erstellen und signieren Sie ein Registrierungstoken

Sie erstellen ein Token und signieren es mit dem privaten Schlüssel, den Sie gerade im vorherigen Schritt generiert haben.

Example – Erstellen Sie ein Registrierungstoken

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Erstellen Sie ein Registrierungstoken, indem Sie den Befehl quorum token-sign generate ausführen:

   ```
   aws-cloudhsm > quorum token-sign generate --service registration --token /path/
   tokenfile
   {
     "error_code": 0,
     "data": {
       "path": "/path/tokenfile"
     }
   }
   ```

3. Der Befehl quorum token-sign generate generiert ein Registrierungstoken im angegebenen Dateipfad. Untersuchen Sie die Tokendatei:

   ```
   $ cat /path/tokenfile{
   ```

```
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": ""
    }
  ]
}
```

Die Tokendatei besteht aus Folgendem:

- approval_data: Ein Base64-kodiertes zufälliges Datentoken, dessen Rohdaten das Maximum von 245 Byte nicht überschreiten.
- unsigniert: Ein base64-kodiertes und SHA256-gehashtes Token der approval_data.
- signiert: Ein base64-kodiertes signiertes Token (Signatur) des unsignierten Tokens, unter Verwendung des zuvor mit OpenSSL generierten privaten RSA-Schlüssels mit 2048 Bit.

Sie signieren das unsignierte Token mit dem privaten Schlüssel, um nachzuweisen, dass Sie Zugriff auf den privaten Schlüssel haben. Sie benötigen die vollständig mit einer Signatur und dem öffentlichen Schlüssel aufgefüllte Registrierungstokendatei, um den Administrator als Quorumbenutzer im AWS CloudHSM-Cluster zu registrieren.

Example – Signieren Sie das unsignierte Registrierungstoken

1. Dekodieren Sie das Base64-kodierte unsignierte Token und platzieren Sie es in einer Binärdatei:

```
$ echo -n '6BMUj6mUjjko6ZLCEdzGlWpR5sILhFJfqhW1ej3Oq1g=' | base64 -d > admin.bin
```

2. Verwenden Sie OpenSSL und den privaten Schlüssel, um das jetzt binäre unsignierte Registrierungstoken zu signieren und eine binäre Signaturdatei zu erstellen:

```
$ openssl pkeyutl -sign \
-inkey admin.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

3. Kodieren Sie die binäre Signatur in Base64:

```
$ base64 -w0 admin.sig.bin > admin.sig.b64
```

4. Kopieren Sie die Base64-kodierte Signatur und fügen Sie sie in die Token-Datei ein:

```
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": <signed token in base64 encoding>
    }
  ]
}
```

Registrieren Sie den öffentlichen Schlüssel beim HSM

Nach der Erstellung eines Schlüssels muss der Administrator den öffentlichen Schlüssel im AWS CloudHSM-Cluster registrieren.

So registrieren Sie einen öffentlichen Schlüssel bei dem HSM

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

```
aws-cloudhsm > login --username admin --role admin
Enter password:
{
  "error_code": 0,
  "data": {
```

```
    "username": "admin",
    "role": "admin"
  }
}
```

3. Verwenden Sie den Befehl user change-quorum token-sign register, um den öffentlichen Schlüssel zu registrieren. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl help user change-quorum token-sign register ausführen.

Example – Registrieren Sie einen öffentlichen Schlüssel beim AWS CloudHSM-Cluster

Das folgende Beispiel zeigt, wie Sie den user change-quorum token-sign register-Befehl in CloudHSM CLI verwenden, um einen öffentlichen Schlüssel von admin im HSM zu registrieren. Der Admin muss bei dem HSM angemeldet sein, um diesen Befehl zu verwenden. Ersetzen Sie diese Werte durch Ihre eigenen Werte:

```
aws-cloudhsm > user change-quorum token-sign register --public-key </path/admin.pub> --
signed-token </path/tokenfile>
     {
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

> **Note**
>
> /path/admin.pub: Der Dateipfad zur PEM-Datei mit dem öffentlichen Schlüssel
> Erforderlich: Ja
> /path/tokenfile: Der Dateipfad mit dem Token, das mit dem privaten Schlüssel des Benutzers signiert wurde
> Erforderlich: Ja

Nachdem alle Admins ihre öffentlichen Schlüssel registriert haben, zeigt die Ausgabe des user list-Befehls dies im Quorumfeld an und gibt an, welche aktivierte Quorum-Strategie verwendet wird, wie unten dargestellt:

```
aws-cloudhsm > user list
```

```
    {
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin3",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
```

```
        "username": "admin4",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

Setzten des Quorum-Mindestwerts für das HSM

Um die Quorum-Authentifizierung zu verwenden, muss sich ein Administrator beim HSM anmelden und dann den Mindestwert für das Quorum festlegen. Dies ist die Mindestanzahl von Admin-Genehmigungen, die erforderlich sind, um HSM-Benutzermanagement-Vorgänge auszuführen. Jeder Administrator auf dem HSM kann den Mindestwert für das Quorum festlegen, auch Administratoren, die keinen Schlüssel zum Signieren registriert haben. Sie können den Quorum-Mindestwert jederzeit ändern. Weitere Informationen finden Sie unter Ändern Sie den Mindestwert.

So legen Sie den Quorum-Mindestwert im HSM fest

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten:

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. Melden Sie sich mit der CloudHSM-CLI als Administrator an.

```
aws-cloudhsm > login --username admin --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

3. Verwenden Sie den Quorum-Token-Zeichen set-quorum-value-Befehl, um den Quorum-Mindestwert festzulegen. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl help quorum token-sign set-quorum-value ausführen.

Example – Setzen des Quorum-Mindestwerts für das HSM

In diesem Beispiel wird ein Quorum-Mindestwert von zwei (2) verwendet. Sie können einen Wert zwischen zwei (2) und acht (8) wählen, bis zur Gesamtzahl der Administratoren auf dem HSM. In diesem Beispiel hat das HSM vier (4) Administratoren, sodass der maximal mögliche Wert vier (4) ist.

Um den folgenden Beispielbefehl zu verwenden, ersetzen Sie die letzte Zahl (*<2>*) durch den bevorzugten Quorum-Mindestwert.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service user --value <2>
{
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

In diesem Beispiel identifiziert der Dienst den HSM-Dienst, dessen Quorum-Mindestwert Sie festlegen. Der Quorum-Token-Zeichen list-quorum-values-Befehl listet die HSM-Diensttypen, -namen und -beschreibungen auf, die im Dienst enthalten sind.

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Jeder Diensttyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Service type (Servicetyp)	Serviceoperationen
user	Admin.	• user create • user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Token-Signatur set-quorum-value

Um den Quorum-Mindestwert für einen Dienst abzurufen, verwenden Sie den quorum token-sign list-quorum-values-Befehl:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}
```

Die Ausgabe des vorangegangenen quorum token-sign list-quorum-values-Befehls zeigt, dass der Quorum-Mindestwert für den HSM-Benutzerdienst, der für die Benutzerverwaltungsvorgänge zuständig ist, jetzt zwei (2) beträgt. Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Quorum verwenden (M von N) fort.

Quorum-Authentifizierung für Administratoren verwenden

Ein Administrator des HSM kann die Quorum-Authentifizierung für die folgenden Vorgänge im AWS CloudHSM-Cluster konfigurieren:

- user create
- user delete
- user change-password
- user change-mfa

Nachdem das AWS CloudHSM-Cluster für die Quorum-Authentifizierung konfiguriert wurde, können Admins selbständig HSM-Benutzermanagement-Vorgänge durchführen. Das folgende Beispiel zeigt die Ausgabe, die angezeigt wird, wenn ein Admin versucht, einen neuen Benutzer auf dem HSM anzulegen. Der Befehl schlägt fehl und es wird ein Fehler angezeigt, der besagt, dass eine Quorumauthentifizierung erforderlich ist.

```
aws-cloudhsm > user create --username user1 --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

Zum Ausführen eines HSM-Benutzermanagement-Vorgangs muss ein Admin folgende Aufgaben erledigen:

Themen

- Abrufen eines Quorum-Tokens
- Unterschriften von genehmigenden Administratoren einholen
- Genehmigen Sie das Token auf dem AWS CloudHSM-Cluster und führen Sie einen Benutzerverwaltungsvorgang aus

Abrufen eines Quorum-Tokens

Zunächst muss der Administrator die CloudHSM-CLI verwenden, um ein Quorum-Token anzufordern.

So fordern Sie ein Quorum-Token an

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Verwenden Sie den login-Befehl und melden Sie sich beim Cluster als Administrator an.

   ```
   aws-cloudhsm>login --username admin --role admin
   ```

3. Verwenden Sie zum Generieren eines Quorum-Tokens den Befehl quorum token-sign generate. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl help quorum token-sign generate ausführen.

Example – Generieren Sie ein Quorum-Token

Dieses Beispiel ruft ein Quorum-Token für den Admin mit Benutzernamen `admin` ab und speichert das Token in einer Datei namens `admin.token`. Zum Verwenden des Beispielbefehls ersetzen Sie diese Werte durch Ihre eigenen:

- *`<admin>`* – Der Name des Administrators, der das Token erhält. Dies muss derselbe Admin sein, der am HSM angemeldet ist und diesen Befehl ausführt.
- *`admin.token`* – Der Name der Datei, in der das Quorum-Token gespeichert wird.

In dem folgenden Befehl gibt `user` den Namen des Dienstes an, für den Sie das Token, das Sie erzeugen, verwenden können. In diesem Fall ist das Token für die HSM-Benutzermanagement-Vorgänge (Service `user`).

```
aws-cloudhsm > login --username <ADMIN> --role <ADMIN> --password <PASSWORD>
{
  "error_code": 0,
  "data": {
    "username": "admin",
```

```
    "role": "admin"
  }
}

aws-cloudhsm > quorum token-sign generate --service user --token </path/admin.token>
{
  "error_code": 0,
  "data": {
    "path": "/home/tfile"
  }
}
```

Der quorum token-sign generate-Befehl generiert ein Quorum-Token für den Benutzerdienst im angegebenen Dateipfad. Die Tokendatei kann wie folgt überprüft werden:

```
$cat </path/admin.token>
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

Die Tokendatei besteht aus Folgendem:

- approval_data: Ein Base64-codiertes Rohdaten-Token, das vom HSM generiert wurde.
- token: Ein Base64-kodiertes und SHA-256-Hash-Token der approval_data
- signatures: Ein Array von Base64-codierten signierten Tokens (Signaturen) des unsignierten Tokens, wobei jede Signatur eines Genehmiger die Form eines JSON-Objektliterals hat:

  ```
  {
        "username": "<APPROVER_USERNAME>",
        "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
  }
  ```

  Jede Signatur wird aus dem Ergebnis erstellt, wenn ein Genehmiger seinen entsprechenden privaten RSA-2048-Bit-Schlüssel verwendet, dessen öffentlicher Schlüssel beim HSM registriert wurde.

Durch Ausführen des quorum token-sign list-Befehls kann bestätigt werden, dass das generierte Benutzerdienst-Quorum-Token auf dem CloudHSM-Cluster vorhanden ist:

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
      {
        "username": "admin",
        "service": "user",
        "approvals-required": {
          "value": 2
        },
        "number-of-approvals": {
          "value": 0
        },
        "token-timeout-seconds": {
          "value": 597
        },
        "cluster-coverage": "full"
      }
    ]
  }
}
```

Die `token-timeout-seconds`-Zeit gibt den Zeitraum in Sekunden an, in dem ein generiertes Token genehmigt werden muss, bevor es abläuft.

Unterschriften von genehmigenden Administratoren einholen

Ein Administrator, der ein Quorum-Token besitzt, muss das Token von anderen Admins genehmigen lassen. Für die Genehmigung nutzen die anderen Admins Ihren Signaturschlüssel zur kryptografischen Token-Signierung. Dies geschieht außerhalb des HSMs.

Es gibt viele verschiedene Möglichkeiten, ein Token zu signieren. Im folgenden Beispiel wird gezeigt, wie dies mit OpenSSL durchgeführt wird. Wenn Sie ein anderes Signatur-Tool verwenden, müssen Sie sicherstellen, dass dieses den privaten Schlüssel (Signaturschlüssel) des Admins zur Signierung des SHA-256-Digests des Tokens verwendet.

Example – Unterschriften von genehmigenden Administratoren einholen

In diesem Beispiel benötigt der Administrator, der das Token (`admin`) besitzt, mindestens zwei (2) Genehmigungen. Das folgende Beispiel zeigt, wie zwei (2) Admins OpenSSL zur kryptografischen Signierung des Tokens einsetzen können.

1. Dekodieren Sie das Base64-kodierte unsignierte Token und platzieren Sie es in einer Binärdatei:

```
$echo -n '0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=' | base64 -d > admin.bin
```

2. Verwenden Sie OpenSSL und den jeweiligen privaten Schlüssel des Genehmigers (`admin3`), um das jetzt binäre unsignierte Quorum-Token für den Benutzerservice zu signieren und eine binäre Signaturdatei zu erstellen:

```
$openssl pkeyutl -sign \
-inkey admin3.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

3. Kodieren Sie die binäre Signatur in Base64:

```
$base64 -w0 admin.sig.bin > admin.sig.b64
```

4. Kopieren Sie abschließend die Base64-kodierte Signatur und fügen Sie sie in die Tokendatei ein. Folgen Sie dabei dem zuvor für die Genehmigersignatur angegebenen JSON-Objektliteralformat:

```
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAABAAMAA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": [
    {
      "username": "admin2",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
```

```
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    },
    {
      "username": "admin3",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    }
  ]
}
```


Genehmigen Sie das Token auf dem AWS CloudHSM-Cluster und führen Sie einen Benutzerverwaltungsvorgang aus

Sobald ein Administrator über die erforderlichen Genehmigungen/Signaturen verfügt hat, wie im vorherigen Abschnitt beschrieben, kann er dieses Token zusammen mit einem der folgenden Benutzerverwaltungsvorgänge an den AWS CloudHSM-Cluster weitergeben:

- create
- delete
- change-password
- user change-mfa

Weitere Informationen zur Verwendung dieser Befehle finden Sie unter Verwenden der CloudHSM CLI.

Während der Transaktion wird das Token innerhalb des AWS CloudHSM-Clusters genehmigt und der angeforderte Benutzerverwaltungsvorgang ausgeführt. Der Erfolg des Benutzerverwaltungsvorgangs hängt sowohl von einem gültigen genehmigten Quorum-Token als auch von einem gültigen Benutzerverwaltungsvorgang ab.

Der Admin kann das Token nur für einen Vorgang nutzen. Wurde dieser erfolgreich ausgeführt, verliert das Token seine Gültigkeit. Um einen weiteren HSM-Benutzerverwaltungsvorgang durchzuführen, muss der Administrator den oben beschriebenen Vorgang wiederholen. Das heißt, der Administrator muss ein neues Quorum-Token generieren, neue Signaturen von den

Genehmigenden einholen und dann das neue Token auf dem HSM mit dem angeforderten Benutzerverwaltungsvorgang genehmigen und verbrauchen.

> **Note**
>
> Das Quorum-Token ist nur gültig, solange Ihre aktuelle Anmeldesitzung geöffnet ist. Wenn Sie sich von CloudHSM-CLI abmelden oder wenn die Netzwerkverbindung unterbrochen wird, ist das Token nicht mehr gültig. Ebenso kann ein autorisiertes Token nur innerhalb der CloudHSM-CLI verwendet werden. Es kann nicht zur Authentifizierung in einer anderen Anwendung verwendet werden.

Example Einen neuen Benutzer als Administrator erstellen

Im folgenden Beispiel erstellt ein eingeloggter Administrator einen neuen Benutzer auf dem HSM:

```
aws-cloudhsm > user create --username user1 --role crypto-user --approval /path/
admin.token
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "user1",
    "role": "crypto-user"
  }
}
```

Der Administrator gibt dann den user list-Befehl ein, um die Erstellung des neuen Benutzers zu bestätigen:

```
aws-cloudhsm > user list{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
```

```
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin3",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin4",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
```

```
        "cluster-coverage": "full"
      },
      {
        "username": "user1",
        "role": "crypto-user",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

Wenn der Administrator versucht, einen anderen HSM-Benutzerverwaltungsvorgang durchzuführen, schlägt dieser mit einem Quorum-Authentifizierungsfehler fehl:

```
aws-cloudhsm > user delete --username user1 --role crypto-user
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

Wie unten gezeigt, zeigt der quorum token-sign list-Befehl, dass der Administrator keine genehmigten Token hat. Um eine weitere HSM-Benutzerverwaltungsoperation durchzuführen, muss der Administrator ein neues Quorum-Token generieren, neue Signaturen von den Genehmigern einholen und die gewünschte Benutzerverwaltungsoperation mit dem Argument --approval ausführen, um das Quorum-Token bereitzustellen, das bei der Ausführung der Benutzerverwaltungsoperation genehmigt und verbraucht werden soll.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
```

```
    "tokens": []
  }
}
```

Den Quorum-Mindestwert für Admins ändern

Nachdem Sie den Quorum-Mindestwert festgelegt haben, sodass Admins die Quorum-Authentifizierung verwenden können, können Sie den Quorum-Mindestwert ändern. Das HSM lässt nur eine Änderung des Quorum-Mindestwerts zu, wenn die Anzahl der Genehmiger gleich oder höher als der aktuelle Quorum-Mindestwert ist. Beispiel: Wenn der Quorum-Mindestwert zwei (2) beträgt, genehmigen mindestens zwei (2) Admins die Änderung des Quorum-Mindestwerts.

> **Note**
>
> Der Quorumwert des Benutzerdienstes muss immer kleiner als der Quorumwert des Quorumdienstes sein. Informationen zu Dienstnamen wie Quorumdienst und Benutzerdienst finden Sie unter Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen.

Um die Zustimmung des Quorums zur Änderung des Quorum-Mindestwerts zu erhalten, benötigen Sie ein Quorum-Token für den quorum service, der den quorum token-sign set-quorum-value-Befehl verwendet. Um ein Quorum-Token für den quorum service zu generieren, der den quorum token-sign set-quorum-value-Befehl verwendet, muss der Quorumdienst höher als eins (1) sein. Das heißt, bevor Sie den Quorum-Mindestwert für den Benutzerdienst ändern können, müssen Sie möglicherweise den Quorum-Mindestwert für den Quorum-Dienst ändern.

Um den Quorum-Mindestwert für Admins ändern

1. Verwenden Sie den folgenden Befehl, um den interaktiven CloudHSM-CLI-Modus zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Verwenden Sie den login-Befehl und melden Sie sich beim Cluster als Administrator an.

```
aws-cloudhsm>login --username <admin> --role admin
```

3. Verwenden Sie den quorum token-sign list-quorum-values-Befehl, um die Quorum-Mindestwerte für alle Dienstnamen abzurufen. Weitere Informationen finden Sie im folgenden Beispiel.
4. Wenn der Mindestwert für den Quorum-Dienst niedriger ist als der Wert für den Benutzerdienst, verwenden Sie den quorum token-sign set-quorum-value-Befehl, um den Wert für den Quorum-Dienst zu ändern. Ändern Sie den Wert für den Quorum-Dienst auf einen Wert (1), der gleich oder höher ist als der Wert für den Benutzerdienst. Weitere Informationen finden Sie im folgenden Beispiel.
5. Erzeugen Sie ein Quorum-Token und achten Sie darauf, dass Sie den Quorum-Dienst als den Dienst angeben, für den Sie das Token verwenden können.
6. Erhalten von Genehmigungen (Signaturen) von anderen Admins.
7. Genehmigen Sie das Token auf dem AWS CloudHSM-Cluster und führen Sie einen Benutzerverwaltungsvorgang aus.
8. Verwenden Sie den quorum token-sign set-quorum-value-Befehl, um den Quorum-Mindestwert für den Benutzerdienst zu ändern.

Example – Quorum-Mindestwerte abrufen und den Wert für den Quorum-Dienst ändern

Der folgende Beispielbefehl zeigt, dass der Quorum-Mindestwert für den Benutzerdienst derzeit zwei (2) beträgt.

```
aws-cloudhsm > quorum token-sign list-quorum-values{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}
```

Um den Quorum-Mindestwert für den Quorum-Dienst zu ändern, verwenden Sie den quorum token-sign set-quorum-value-Befehl und stellen einen Wert ein, der gleich oder höher ist als der Wert für den Benutzerdienst. Im folgenden Beispiel wird der Quorum-Mindestwert für den Quorum-Dienst auf zwei (2) gesetzt, derselbe Wert, der auch für den Benutzerdienst festgelegt wurde.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2{
```

```
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

Der folgende Befehl zeigt, dass der Quorum-Mindestwert jetzt zwei (2) für den Benutzerdienst und den Quorumdienst beträgt.

```
aws-cloudhsm > quorum token-sign list-quorum-values{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 2
  }
}
```

Verwaltung von HSM-Benutzern mit CloudHSM Management Utility (CMU)

In AWS CloudHSM müssen Sie die Befehlszeilentools CloudHSM CLI oder CloudHSM Management Utility (CMU) verwenden, um die Benutzer auf Ihrem HSM zu erstellen und zu verwalten. CloudHSM CLI ist für die Verwendung mit dem neuesten SDK konzipiert, während die CMU für die Verwendung mit früheren SDKs konzipiert ist.

Themen

- Grundlegendes zu HSM-Benutzern
- Tabelle der HSM-Benutzerberechtigungen
- Verwenden von CloudHSM Management Utility (CMU) zur Verwaltung von Benutzern
- Verwendung von CloudHSM Management Utility (CMU) zur Verwaltung der Zwei-Faktor-Authentifizierung (2FA) für Crypto Officers
- Verwendung von CloudHSM Management Utility (CMU) zur Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle)

Grundlegendes zu HSM-Benutzern

Für die meisten Operationen, die Sie auf dem HSM ausführen, sind die Anmeldeinformationen eines HSM-Benutzers erforderlich. Das HSM authentifiziert jeden HSM-Benutzer, und jeder HSM-Benutzer hat einen Typ, der bestimmt, welche Operationen Sie als dieser Benutzer auf dem HSM ausführen können.

> **Note**
>
> HSM-Benutzer unterscheiden sich von IAM-Benutzern. IAM-Benutzer mit den richtigen Anmeldeinformationen können HSMs erstellen, indem sie über die AWS-API mit Ressourcen interagieren. Nachdem das HSM erstellt wurde, müssen Sie HSM-Benutzeranmeldedaten verwenden, um Vorgänge auf dem HSM zu authentifizieren.

Benutzertypen

- Precrypto Officer (PRECO)
- Crypto Officer (CO)
- Crypto-Benutzer (Crypto User, CU)
- Appliance-Benutzer (Appliance User, AU)

Precrypto Officer (PRECO)

Sowohl im Cloud Management Utility (CMU) als auch im Key Management Utility (KMU) ist PRECO ein temporärer Benutzer, der nur auf dem ersten HSM in einem AWS CloudHSM -Cluster existiert. Das erste HSM in einem neuen Cluster enthält einen PRECO-Benutzer, der angibt, dass dieser Cluster noch nie aktiviert wurde. Um einen Cluster zu aktivieren, führen Sie die cloudhsm-cli aus und führen den cluster activate-Befehl aus. Loggen Sie sich in das HSM ein und ändern Sie das PRECO-Passwort. Wenn Sie das Passwort ändern, wird dieser Benutzer zum Crypto Officer (CO).

Crypto Officer (CO)

Sowohl im Cloud Management Utility (CMU) als auch im Key Management Utility (KMU) kann ein Crypto Officer (CO) Benutzerverwaltungsvorgänge durchführen. Beispielsweise können Sie Benutzer erstellen und löschen und Benutzerpasswörter ändern. Weitere Informationen über CO-Benutzer finden Sie unter Tabelle der HSM-Benutzerberechtigungen. Wenn Sie einen neuen Cluster aktivieren, wird der Benutzer von einem Precrypto Officer (PRECO) zu einem Crypto Officer (CO).

Crypto-Benutzer (Crypto User, CU)

Ein Crypto-Benutzer (CU) kann die folgenden Schlüsselverwaltungs- und kryptografischen Vorgänge ausführen.

- Schlüsselverwaltung – Erstellen, Löschen, Freigeben, Importieren und Exportieren von kryptographischen Schlüsseln.

- Kryptografische Operationen – Verwenden Sie kryptographische Schlüssel für die Verschlüsselung, Entschlüsselung, Signatur, Verifizierung und mehr.

Weitere Informationen hierzu finden Sie unter Tabelle der HSM-Benutzerberechtigungen.

Appliance-Benutzer (Appliance User, AU)

Der Appliance-Benutzer (AU) kann Kloning- und Synchronisierungsvorgänge an den HSMs Ihres Clusters durchführen. AWS CloudHSM verwendet die AU, um die HSMs in einem Cluster zu synchronisieren. AWS CloudHSM Die AU ist auf allen HSMs vorhanden, die von bereitgestellt werden AWS CloudHSM, und verfügt über eingeschränkte Berechtigungen. Weitere Informationen hierzu finden Sie unter Tabelle der HSM-Benutzerberechtigungen.

AWS kann keine Operationen an Ihren HSMs ausführen. AWS kann Ihre Benutzer oder Schlüssel nicht anzeigen oder ändern und mit diesen Schlüsseln keine kryptografischen Operationen ausführen.

Tabelle der HSM-Benutzerberechtigungen

In der folgenden Tabelle sind HSM-Operationen aufgeführt, sortiert nach dem Typ des HSM-Benutzers oder der HSM-Sitzung, die den Vorgang ausführen kann.

	Crypto Officer (CO)	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentif izierte Sitzungen
Grundlege nde Cluster-I nformationen erhalten[1]	Ja	Ja	Ja	Ja
Das eigene Passwort ändern	Ja	Ja	Ja	Nicht zutreffend

	Crypto Officer (CO)	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentifizierte Sitzungen
Das Passwort eines Benutzers ändern	 Ja	 Nein	 Nein	 Nein
Benutzer hinzufügen oder entfernen	 Ja	 Nein	 Nein	 Nein
Den Synchronisierungsstatus erhalten[2]	 Ja	 Ja	 Ja	 Nein
Maskierte Objekte extrahieren und einfügen[3]	 Ja	 Ja	 Ja	 Nein
Funktionen zur Schlüsselverwaltung[4]	 Nein	 Ja	 Nein	 Nein
Verschlüsseln/ Entschlüsseln	 Nein	 Ja	 Nein	 Nein

	Crypto Officer (CO)	Crypto-Benutzer (Crypto User, CU)	Appliance -Benutzer (Appliance User, AU)	Nicht authentifizierte Sitzungen
Signieren/ Überprüfen	 Nein	 Ja	 Nein	 Nein
Generieren von Digests und HMACs	 Nein	 Ja	 Nein	 Nein

- [1] Zu den grundlegenden Informationen gehören u. a die Anzahl der HSMs im Cluster sowie IP-Adresse, Modell, Seriennummer, Geräte-ID, Firmware-ID usw. der einzelnen HSMs.
- [2] Der Benutzer kann verschiedene Digests (Hashes) erhalten, die den Schlüsseln auf dem HSM entsprechen. Eine Anwendung kann diese Digest-Gruppen vergleichen, um den Synchronisierungsstatus von HSMs in einem Cluster zu erfahren.
- [3] Maskierte Objekte sind Schlüssel, die verschlüsselt werden, bevor sie das HSM verlassen. Sie können außerhalb des HSM nicht entschlüsselt werden. Sie werden erst entschlüsselt, nachdem sie in ein HSM eingefügt wurden, das sich in demselben Cluster befindet wie das HSM, aus dem sie extrahiert wurden. Eine Anwendung kann maskierte Objekte extrahieren und einfügen, um die HSMs in einem Cluster zu synchronisieren.
- [4] Die Funktionen zur Schlüsselverwaltung umfassen unter anderem das Erstellen, Löschen, Verpacken, Entpacken und Ändern der Schlüsselattribute.

Verwenden von CloudHSM Management Utility (CMU) zur Verwaltung von Benutzern

Dieses Thema enthält step-by-step Anweisungen zur Verwaltung von HSM-Benutzern (Hardware Security Module) mit dem CloudHSM Management Utility (CMU), einem Befehlszeilentool, das im Client-SDK enthalten ist. Weitere Informationen über CMU- oder HSM-Benutzer finden Sie unter CloudHSM-Verwaltungsdienstprogramm und Grundlegendes zu HSM-Benutzern.

Sections

Grundlegendes zur HSM-Benutzerverwaltung mit CMU

Um HSM-Benutzer zu verwalten, müssen Sie sich am HSM mit dem Benutzernamen und dem Passwort eines Crypto Officer (CO) anmelden. Nur COs können Benutzer verwalten. Das HSM enthält einen Standard-CO mit dem Namen admin. Sie haben das Passwort für admin festgelegt, wenn Sie das Cluster aktiviert haben.

Um CMU verwenden zu können, müssen Sie das Configure-Tool verwenden, um die lokale Konfiguration zu aktualisieren. Die CMU stellt eine eigene Verbindung zum Cluster her, und diese Verbindung ist nicht clusterfähig. Um Clusterinformationen nachzuverfolgen, verwaltet die CMU eine lokale Konfigurationsdatei. Das bedeutet, dass Sie bei jeder Verwendung der CMU zunächst die Konfigurationsdatei aktualisieren sollten, indem Sie das Befehlszeilentool configure mit dem `--cmu`-Parameter ausführen. Wenn Sie das Client-SDK 3.2.1 oder früher verwenden, müssen Sie einen anderen Parameter als `--cmu` verwenden. Weitere Informationen finden Sie unter the section called “Verwenden von CMU mit Client-SDK 3.2.1 und früher”.

Für den `--cmu`-Parameter müssen Sie die IP-Adresse eines HSM in Ihrem Cluster hinzufügen. Wenn Sie mehrere HSMs haben, können Sie eine beliebige IP-Adresse verwenden. Dadurch wird sichergestellt, dass die CMU alle von Ihnen vorgenommenen Änderungen auf den gesamten Cluster übertragen kann. Denken Sie daran, dass die CMU ihre lokale Datei verwendet, um Clusterinformationen zu verfolgen. Wenn sich der Cluster seit der letzten Verwendung der CMU von einem bestimmten Host aus geändert hat, müssen Sie diese Änderungen der lokalen Konfigurationsdatei hinzufügen, die auf diesem Host gespeichert ist. Fügen Sie niemals ein HSM hinzu oder entfernen Sie es, während Sie CMU verwenden.

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.

4. Um die IP-Adresse abzurufen, wählen Sie auf der Registerkarte HSMs eine der IP-Adressen aus, die unter ENI-IP-Adresse aufgeführt sind.

Um eine IP-Adresse für ein HSM (CLI) zu erhalten

- Rufen Sie die IP-Adresse eines HSM mit dem describe-clusters Befehl von der CLI ab. In der Ausgabe des Befehls sind die IP-Adressen der HSMs die Werte von `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

Verwenden von CMU mit Client-SDK 3.2.1 und früher

Mit Client SDK 3.3.0 AWS CloudHSM wurde Unterstützung für den `--cmu` Parameter hinzugefügt, was die Aktualisierung der Konfigurationsdatei für CMU vereinfacht. Wenn Sie eine CMU-Version von Client-SDK 3.2.1 oder früher verwenden, müssen Sie weiterhin die Parameter `-a` und `-m` verwenden, um die Konfigurationsdatei zu aktualisieren. Weitere Informationen zu diesen Parametern finden Sie unter Configure-Tool.

Laden Sie das CloudHSM Management Utility herunter

Die neueste Version von CMU ist für HSM-Benutzerverwaltungsaufgaben verfügbar, unabhängig davon, ob Sie Client-SDK 5 und Client-SDK 3 verwenden.

So laden Sie CMU herunter und installieren es

- Laden Sie jq herunter und installieren Sie sie.

  Amazon Linux

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-mgmt-util-latest.el6.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el6.x86_64.rpm
  ```

  Amazon Linux 2

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  CentOS 7.8+

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  CentOS 8.3+

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-mgmt-util-latest.el8.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el8.x86_64.rpm
  ```

  RHEL 7 (7.8+)

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-mgmt-util_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-mgmt-util_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-mgmt-util_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-mgmt-util_latest_u18.04_amd64.deb
```

Windows Server 2012

1. Laden Sie das CloudHSM Management Utility herunter
2. Führen Sie das CMU-Installationsprogramm (AWSCloudHSMManagementUtil-latest.msi) mit Windows-Administratorrechten aus.

Windows Server 2012 R2

1. Laden Sie das CloudHSM Management Utility herunter
2. Führen Sie das CMU-Installationsprogramm (AWSCloudHSMManagementUtil-latest.msi) mit Windows-Administratorrechten aus.

Windows Server 2016

1. Laden Sie das CloudHSM Management Utility herunter
2. Führen Sie das CMU-Installationsprogramm (AWSCloudHSMManagementUtil-latest.msi) mit Windows-Administratorrechten aus.

Wie verwaltet man HSM-Benutzer mit CMU

Dieser Abschnitt enthält grundlegende Befehle zur Verwaltung von HSM-Benutzern mit CMU.

So erstellen Sie HSM-Benutzer

Verwenden Sie createUser, um neue Benutzer auf dem HSM zu erstellen. Sie müssen sich als CO anmelden, um einen Benutzer zu erstellen.

Um einen neuen CO-Benutzer zu erstellen

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Starten von CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. Melden Sie sich beim HSM als CO-Benutzer an.

```
aws-cloudhsm>loginHSM CO admin co12345
```

Stellen Sie sicher, dass die Anzahl der Verbindungen, die in den CMU-Listen aufgeführt sind, mit der Anzahl der HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Verwenden Sie createUser, um einen CO-Benutzer namens **example_officer** mit dem Passwort **password1** zu erstellen.

```
aws-cloudhsm>createUser CO example_officer password1
```

Die CMU fordert Sie auf, den Vorgang zum Erstellen eines Benutzers auszuführen.

```
************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?
```

5. Typ **y**.

Um einen neuen CU-Benutzer zu erstellen

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Starten von CMU.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. Melden Sie sich beim HSM als CO-Benutzer an.

```
aws-cloudhsm>loginHSM CO admin co12345
```

Stellen Sie sicher, dass die Anzahl der Verbindungen, die in den CMU-Listen aufgeführt sind, mit der Anzahl der HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Verwenden Sie createUser, um einen CU-Benutzer namens **example_user** mit dem Passwort **password1** zu erstellen.

```
aws-cloudhsm>createUser CU example_user password1
```

Die CMU fordert Sie auf, den Vorgang zum Erstellen eines Benutzers auszuführen.

```
**********************CAUTION*****************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**********************************************************

Do you want to continue(y/n)?
```

5. Typ **y**.

Für weitere Informationen über createUser finden Sie unter createUser.

Um alle HSM-Benutzer im Cluster aufzulisten

Verwenden Sie den listUsers-Befehl, um alle Benutzer im Cluster aufzulisten. Sie müssen sich nicht anmelden, um listUsers auszuführen, und alle Benutzertypen können Benutzer auflisten.

Um alle Benutzer im Cluster aufzulisten

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Starten von CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. Verwenden Sie listUsers, um alle Benutzer im Cluster aufzulisten.

   ```
   aws-cloudhsm>listUsers
   ```

   CMU listet alle Benutzer des Clusters auf.

   ```
   Users on server 0(10.0.2.9):
   Number of users found:4

       User Id             User Type       User Name
    MofnPubKey    LoginFailureCnt         2FA
   ```

```
        1               AU              app_user                        NO
              0               NO
        2               CO              example_officer                 NO
              0               NO
        3               CU              example_user                    NO
              0               NO
Users on server 1(10.0.3.11):
Number of users found:4

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
        1               AU              app_user                        NO
              0               NO
        2               CO              example_officer                 NO
              0               NO
        3               CU              example_user                    NO
              0               NO
Users on server 2(10.0.1.12):
Number of users found:4

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
        1               AU              app_user                        NO
              0               NO
        2               CO              example_officer                 NO
              0               NO
        3               CU              example_user                    NO
              0               NO
```

Für weitere Informationen über listUsers finden Sie in der listUsers.

Um HSM-Benutzerpasswörter zu ändern

Verwenden Sie changePswd, um ein Passwort zu ändern.

Bei Benutzertypen und Passwörtern wird zwischen Groß- und Kleinschreibung unterschieden, bei Benutzernamen jedoch nicht.

CO, Crypto-Benutzer (CU) und Appliance-Benutzer (AU) können ihr eigenes Passwort ändern. Um das Passwort eines anderen Benutzers zu ändern, müssen Sie sich als CO anmelden. Sie können das Passwort eines Benutzers, der gerade angemeldet ist, nicht ändern.

So ändern Sie Ihr eigenes Passwort

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Starten von CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. Anmelden beim HSM.

   ```
   aws-cloudhsm>loginHSM CO admin co12345
   ```

   Stellen Sie sicher, dass die Anzahl der Verbindungen, die in den CMU-Listen aufgeführt sind, mit der Anzahl der HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Mit changePswd können Sie Ihr eigenes Passwort ändern.

   ```
   aws-cloudhsm>changePswd CO example_officer <new password>
   ```

   Die CMU informiert Sie über den Vorgang zum Ändern des Passworts.

   ```
   ************************CAUTION*********************************
   This is a CRITICAL operation, should be done on all nodes in the
   ```

```
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*************************************************************

Do you want to continue(y/n)?
```

5. Typ **y**.

   Die CMU informiert Sie über den Vorgang zum Ändern des Passworts.

```
Changing password for example_officer(CO) on 3 nodes
```

So ändern Sie das Passwort eines anderen Benutzers

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

   Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

   Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
```

2. Starten von CMU.

   Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

   Windows

```
C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. Melden Sie sich beim HSM als CO-Benutzer an.

```
aws-cloudhsm>loginHSM CO admin co12345
```

Stellen Sie sicher, dass die Anzahl der Verbindungen, die in den CMU-Listen aufgeführt sind, mit der Anzahl der HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Verwenden Sie changePswd, um das Passwort eines anderen Benutzers zu ändern.

```
aws-cloudhsm>changePswd CU example_user <new password>
```

Die CMU informiert Sie über den Vorgang zum Ändern des Passworts.

```
**************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
******************************************************************

Do you want to continue(y/n)?
```

5. Typ **y**.

Die CMU informiert Sie über den Vorgang zum Ändern des Passworts.

```
Changing password for example_user(CU) on 3 nodes
```

Für weitere Informationen über changePswd finden Sie unter changePswd.

So löschen Sie HSM-Benutzer

Verwenden Sie deleteUser, um einen Benutzer zu löschen. Sie müssen sich als CO anmelden, um einen anderen Benutzer zu löschen.

> **Tip**
>
> Sie können keine Crypto-Benutzer (CU) löschen, die Schlüssel besitzen.

Benutzer löschen

1. Verwenden Sie das Configure-Tool, um die CMU-Konfiguration zu aktualisieren.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Starten von CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. Melden Sie sich beim HSM als CO-Benutzer an.

   ```
   aws-cloudhsm>loginHSM CO admin co12345
   ```

   Stellen Sie sicher, dass die Anzahl der Verbindungen, die in den CMU-Listen aufgeführt sind, mit der Anzahl der HSMs im Cluster übereinstimmt. Wenn nicht, melden Sie sich ab und beginnen Sie von vorne.

4. Verwenden Sie deleteUser, um einen Benutzer zu löschen.

   ```
   aws-cloudhsm>deleteUser CO example_officer
   ```

   CMU löscht den Benutzer.

   ```
   Deleting user example_officer(CO) on 3 nodes
   deleteUser success on server 0(10.0.2.9)
   ```

```
deleteUser success on server 1(10.0.3.11)
deleteUser success on server 2(10.0.1.12)
```

Für weitere Informationen über deleteUser finden Sie unter deleteUser.

Verwendung von CloudHSM Management Utility (CMU) zur Verwaltung der Zwei-Faktor-Authentifizierung (2FA) für Crypto Officers

Für mehr Sicherheit können Sie die Zwei-Faktor-Authentifizierung (2FA) konfigurieren, um den Cluster zu schützen. Sie können 2FA nur für Crypto Officers (CO) aktivieren.

> **Note**
>
> Sie können 2FA nicht für Crypto-Benutzer (CU) oder Anwendungen aktivieren. Die Zwei-Faktor-Authentifizierung (2FA) ist nur für CO-Benutzer verfügbar.

Themen

- 2FA für HSM-Benutzer verstehen
- Mit 2FA für HSM-Benutzer arbeiten

2FA für HSM-Benutzer verstehen

Wenn Sie sich mit einem 2FA-fähigen HSM-Konto (Hardware Service Module) bei einem Cluster anmelden, geben Sie cloudhsm_mgmt_util (CMU) Ihr Passwort – den ersten Faktor, was Sie wissen – und CMU stellt Ihnen ein Token zur Verfügung und fordert Sie auf, das Token signieren zu lassen. Um den zweiten Faktor bereitzustellen – was Sie haben – signieren Sie das Token mit einem privaten Schlüssel aus einem Schlüsselpaar, das Sie bereits erstellt und dem HSM-Benutzer zugeordnet haben. Um auf den Cluster zuzugreifen, stellen Sie der CMU das signierte Token zur Verfügung.

Quorum-Authentifizierung und 2FA

Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für 2FA. Das bedeutet, dass ein Benutzer mit aktivierter 2FA effektiv für M-von-N-Zugriffskontrolle (MoFN) registriert ist. Beachten Sie die folgenden Punkte, um 2FA und die Quorum-Authentifizierung erfolgreich für denselben HSM-Benutzer zu verwenden:

- Wenn Sie heute die Quorum-Authentifizierung für einen Benutzer verwenden, sollten Sie dasselbe Schlüsselpaar verwenden, das Sie für den Quorum-Benutzer erstellt haben, um 2FA für den Benutzer zu aktivieren.
- Wenn Sie die 2FA-Anforderung für einen Nicht-2FA-Benutzer hinzufügen, der kein Quorum-Authentifizierungsbenutzer ist, registrieren Sie diesen Benutzer als MoFN-Benutzer mit 2FA-Authentifizierung.
- Wenn Sie die 2FA-Anforderung entfernen oder das Passwort für einen 2FA-Benutzer ändern, der auch ein Quorum-Authentifizierungsbenutzer ist, entfernen Sie auch die Registrierung des Quorumbenutzers als MoFN-Benutzer.
- Wenn Sie die 2FA-Anforderung aufheben oder das Passwort für einen 2FA-Benutzer ändern, der auch ein Quorum-Authentifizierungsbenutzer ist, Sie aber trotzdem möchten, dass dieser Benutzer an der Quorumauthentifizierung teilnimmt, müssen Sie diesen Benutzer erneut als MoFN-Benutzer registrieren.

Weitere Informationen über die Quorum-Authentifizierung finden Sie unter Verwendung von CMU zur Verwaltung der Quorum-Authentifizierung.

Mit 2FA für HSM-Benutzer arbeiten

In diesem Abschnitt wird beschrieben, wie HSM-Benutzer mit 2FA arbeiten, einschließlich der Erstellung von 2FA-HSM-Benutzern, der Rotation von Schlüsseln und der Anmeldung am HSM als 2FA-fähige Benutzer. Weitere Informationen zur Arbeit mit HSM-Benutzern finden Sie unter ???, ???, ???, ??? und ???.

Erstellen von 2FA-Benutzern

Um 2FA für einen HSM-Benutzer zu aktivieren, verwenden Sie einen Schlüssel, der die folgenden Anforderungen erfüllt.

Anforderungen an das 2FA-Schlüsselpaar

Sie können ein neues Schlüsselpaar erstellen oder einen vorhandenen Schlüssel verwenden, der die folgenden Anforderungen erfüllt.

- Schlüsseltyp: Asymmetrisch
- Schlüsselnutzung: Signieren und Verifizieren
- Schlüsselspezifikation: RSA_2048

- Der Signierungsalgorithmus umfasst:
  - `sha256WithRSAEncryption`

> **Note**
>
> Wenn Sie die Quorumauthentifizierung verwenden oder planen, die Quorumauthentifizierung zu verwenden, finden Sie weitere Informationen unter the section called “Quorum-Authentifizierung und 2FA”.

Verwenden Sie CMU und das Schlüsselpaar, um einen neuen CO-Benutzer mit aktivierter 2FA zu erstellen.

Um CO-Benutzer mit aktivierter 2FA zu erstellen

1. Führen Sie an einem Terminal die folgenden Schritte aus:

   a. Greifen Sie auf Ihr HSM zu und melden Sie sich beim CloudHSM Management Utility an:

   ```
   /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   b. Melden Sie sich als CO an und erstellen Sie mit dem folgenden Befehl eine neue Benutzer-MFA mit 2FA:

   ```
   aws-cloudhsm>createUser CO MFA <CO USER NAME> -2fa /home/ec2-user/authdata
   ************************CAUTION*********************************This is a
    CRITICAL operation, should be done on all nodes in the
   cluster. AWS does NOT synchronize these changes automatically with the
   nodes on which this operation is not executed or failed, please
   ensure this operation is executed on all nodes in the cluster.
   ************************************************************

   Do you want to continue(y/n)?

   yCreating User exampleuser3(CO) on 1 nodesAuthentication data written to: "/
   home/ec2-user/authdata"Generate Base64-encoded signatures for SHA256 digests in
    the authentication datafile.
   To generate the signatures, use the RSA private key, which is the second factor
    ofauthentication for this user. Paste the signatures and the corresponding
    public keyinto the authentication data file and provide
   ```

```
the file path below.Leave this field blank to use the path initially
 provided.Enter filename:
```

c. Belassen Sie das obige Terminal in diesem Zustand. Drücken Sie nicht die Eingabetaste und geben Sie keinen Dateinamen ein.

2. Führen Sie in einem anderen Terminal die folgenden Schritte aus:

a. Greifen Sie auf Ihr HSM zu und melden Sie sich beim CloudHSM Management Utility an:

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

b. Generieren Sie mit den folgenden Befehlen ein öffentlich-privates Schlüsselpaar:

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt
 rsa_keygen_bits:2048
```

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

c. Führen Sie den folgenden Befehl aus, um eine JSON-Abfragefunktion zum Extrahieren des Digest aus der Authdata-Datei zu installieren:

```
sudo yum install jq
```

d. Um den Digest-Wert zu extrahieren, suchen Sie zunächst die folgenden Daten in der Authdata-Datei:

```
{
  "Version":"1.0",
  "PublicKey":"",
  "Data":[
    {
      "HsmId": <"HSM ID">,
      "Digest": <"DIGEST">,
      "Signature":""
    }
  ]
}
```

> **Note**
>
> Der erhaltene Digest ist Base64-kodiert. Um den Digest zu signieren, muss die Datei jedoch zuerst dekodiert und dann signiert werden. Der folgende Befehl dekodiert den Digest und speichert den dekodierten Inhalt in „digest1.bin“.
>
> ```
> cat authdata | jq '.Data[0].Digest' | cut -c2- | rev | cut -c2- | rev |
>  base64 -d > digest1.bin
> ```

e. Konvertiert den Inhalt des öffentlichen Schlüssels, fügt „\n“ hinzu und entfernt Leerzeichen, wie hier gezeigt:

```
-----BEGIN PUBLIC KEY-----\n<PUBLIC KEY>\n-----END PUBLIC KEY-----
```

> **Important**
>
> Der obige Befehl zeigt, wie „\n“ unmittelbar nach BEGIN PUBLIC KEY----- hinzugefügt wird, Leerzeichen zwischen „\n“ und dem ersten Zeichen des öffentlichen Schlüssels entfernt werden, „\n“ vor -----END PUBLIC KEY hinzugefügt wird und Leerzeichen zwischen „\n“ und dem Ende des öffentlichen Schlüssels entfernt werden.

Dies ist das PEM-Format für den öffentlichen Schlüssel, das in der Authdata-Datei akzeptiert wird.

f. Fügen Sie den Inhalt des öffentlichen Schlüssels im PEM-Format in den Abschnitt mit dem öffentlichen Schlüssel der Authdata-Datei ein.

```
vi authdata
```

```
{
  "Version":"1.0",
  "PublicKey":"-----BEGIN PUBLIC KEY-----\n<"PUBLIC KEY">\n-----END PUBLIC
 KEY-----",
  "Data":[
{
```

```
  "HsmId":<"HSM ID">,
  "Digest":<"DIGEST">,
  "Signature":””
}
]
}
```

g. Signieren Sie die Token-Datei mit dem folgenden Befehl:

```
openssl pkeyutl -sign -in digest1.bin -inkey private_key.pem -pkeyopt
 digest:sha256 | base64

Output Expected:
<"THE SIGNATURE">
```

> **Note**
>
> Verwenden Sie, wie im obigen Befehl gezeigt, openssl pkeyutl statt openssl dgst zum Signieren.

h. Fügen Sie den signierten Digest in die Authdata-Datei im Feld „Signatur“ ein.

```
vi authdata
```

```
{
    "Version": "1.0",
    "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
    "Data": [
        {
            "HsmId": <"HSM ID">,
            "Digest": <"DIGEST">,
            "Signature": "Kkdl ... rkrvJ6Q=="
        },
        {
            "HsmId": <"HSM ID">,
            "Digest": <"DIGEST">,
            "Signature": "K1hxy ... Q261Q=="
        }
    ]
}
```

3. Gehen Sie zurück zum ersten Terminal und drücken Sie **`Enter`**:

```
Generate Base64-encoded signatures for SHA256 digests in the authentication
 datafile. To generate the signatures, use the RSA private key,
which is the second factor ofauthentication for this user. Paste the signatures and
 the corresponding public keyinto the authentication data file and provide the file
 path below. Leave this field blank to use the path initially provided.
Enter filename: >>>>> Press Enter here

createUser success on server 0(10.0.1.11)
```

2FA für HSM-Benutzer verwalten

Verwenden Sie „Passwort ändern“, um das Passwort für einen 2FA-Benutzer zu ändern oder um 2FA zu aktivieren oder zu deaktivieren oder um den 2FA-Schlüssel zu rotieren. Jedes Mal, wenn Sie 2FA aktivieren, müssen Sie einen öffentlichen Schlüssel für 2FA-Logins angeben.

Change Password führt eines der folgenden Szenarien aus:

- Ändern Sie das Passwort für einen 2FA-Benutzer
- Ändern Sie das Passwort für einen Nicht-2FA-Benutzer
- 2FA zu einem Nicht-2FA-Benutzer hinzufügen
- 2FA von einem 2FA-Benutzer entfernen
- Den Schlüssel für einen 2FA-Benutzer rotieren

Sie können Aufgaben auch kombinieren. Sie können beispielsweise 2FA von einem Benutzer entfernen und gleichzeitig das Passwort ändern, oder Sie können den 2FA-Schlüssel wechseln und das Benutzerpasswort ändern.

Um Passwörter zu ändern oder Schlüssel für CO-Benutzer mit aktivierter 2FA zu rotieren

1. Verwenden Sie CMU, um sich als CO mit aktivierter 2FA am HSM anzumelden.
2. Verwenden Sie changePswd, um das Passwort zu ändern oder den Schlüssel von CO-Benutzern mit aktivierter 2FA zu wechseln. Verwenden Sie den `-2fa`-Parameter und geben Sie einen Speicherort im Dateisystem an, an dem das System die `authdata`-Datei schreiben kann. Diese Datei enthält einen Digest für jedes HSM im Cluster.

```
aws-cloudhsm>changePswd CO example-user <new-password> -2fa /path/to/authdata
```

CMU fordert Sie auf, den privaten Schlüssel zu verwenden, um die Digests in der `authdata`-Datei zu signieren und die Signaturen mit dem öffentlichen Schlüssel zurückzugeben.

3. Verwenden Sie den privaten Schlüssel, um die Digests in der `authdata`-Datei zu signieren, fügen Sie die Signaturen und den öffentlichen Schlüssel zur `authdata`-Datei im JSON-Format hinzu und teilen Sie CMU dann den Speicherort der `authdata`-Datei mit. Weitere Informationen finden Sie unter the section called “Konfigurationsreferenz”.

> **Note**
>
> Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für 2FA. Wenn Sie die Quorumauthentifizierung verwenden oder planen, die Quorumauthentifizierung zu verwenden, finden Sie weitere Informationen unter the section called “Quorum-Authentifizierung und 2FA”.

So deaktivieren Sie 2FA für CO-Benutzer mit aktivierter 2FA

1. Verwenden Sie CMU, um sich als CO mit aktivierter 2FA am HSM anzumelden.
2. Verwenden Sie changePswd, um 2FA von CO-Benutzern zu entfernen, bei denen 2FA aktiviert ist.

```
aws-cloudhsm>changePswd CO example-user <new password>
```

CMU fordert Sie auf, den Vorgang zum Ändern des Passworts zu bestätigen.

> **Note**
>
> Wenn Sie die 2FA-Anforderung entfernen oder das Passwort für einen 2FA-Benutzer ändern, der auch ein Quorum-Authentifizierungsbenutzer ist, entfernen Sie auch die Registrierung des Quorumbenutzers als MoFN-Benutzer. Weitere Informationen über Quorum-Benutzer und 2FA finden Sie unter the section called “Quorum-Authentifizierung und 2FA”.

3. Typ **y**.

Die CMU bestätigt den Vorgang zum Ändern des Passworts.

Konfigurationsreferenz

Im Folgenden finden Sie ein Beispiel für die 2FA-Eigenschaften in der `authdata`-Datei sowohl für die von der CMU generierte Anfrage als auch für Ihre Antworten.

```
{
    "Version": "1.0",
    "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
    "Data": [
        {
            "HsmId": "hsm-lgavqitns2a",
            "Digest": "k5O1p3f6foQRVQH7S8Rrjcau6h3TYqsSdr16A54+qG8=",
            "Signature": "Kkdl ... rkrvJ6Q=="
        },
        {
            "HsmId": "hsm-lgavqitns2a",
            "Digest": "IyBcx4I5Vyx1jztwvXinCBQd9lDx8oQe7iRrWjBAi1w=",
            "Signature": "K1hxy ... Q261Q=="
        }
    ]
}
```

Daten

Oberster Knoten Enthält einen untergeordneten Knoten für jedes HSM im Cluster. Erscheint in Anfragen und Antworten für alle 2FA-Befehle.

Digest

Dies müssen Sie unterschreiben, um den zweiten Authentifizierungsfaktor bereitzustellen. CMU wurde in Anfragen für alle 2FA-Befehle generiert.

HsmId

Die ID Ihres HSM. Erscheint in Anfragen und Antworten für alle 2FA-Befehle.

PublicKey

Der öffentliche Schlüsselteil des Schlüsselpaar, das Sie generiert haben, wurde als Zeichenfolge im PEM-Format eingefügt. Sie geben dies in Antworten für createUser und changePswd ein.

Signature

Der mit Base 64 kodierte signierte Digest. Sie geben dies in Antworten für alle 2FA-Befehle ein.

Version

Die Version der JSON-formatierten Datei mit den Authentifizierungsdaten Erscheint in Anfragen und Antworten für alle 2FA-Befehle.

Verwendung von CloudHSM Management Utility (CMU) zur Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle)

Die HSMs in Ihrem AWS CloudHSM-Cluster unterstützen die Quorum-Authentifizierung (auch M of N-Zugriffskontrolle genannt). Mit der Quorum-Authentifizierung kann kein einzelner Benutzer im HSM quorum-kontrollierte Operationen im HSM durchführen. Stattdessen muss eine Mindestanzahl von HSM-Benutzern (mindestens 2) zusammenarbeiten, um diese Operationen durchzuführen. Mit der Quorum-Authentifizierung können Sie eine zusätzliche Schutzebene hinzufügen, indem Sie Genehmigungen von mehr als einem HSM-Benutzer erfordern.

Die Quorum-Authentifizierung kann die folgenden Vorgänge steuern:

- HSM-Benutzerverwaltung durch Verschlüsselungsverantwortliche (Crypto Officers, COs) Anlegen und Löschen von HSM-Benutzern und Ändern des Passworts eines anderen HSM-Benutzers. Weitere Informationen finden Sie unter Verwenden der Quorum-Authentifizierung für Verschlüsselungsverantwortliche.

Die folgenden Themen stellen weitere Informationen zur Quorum-Authentifizierung in AWS CloudHSM zur Verfügung.

Themen

- Überblick über die Quorum-Authentifizierung
- Weitere Details zur Quorum-Authentifizierung
- Verwenden der Quorum-Authentifizierung für Verschlüsselungsverantwortliche: Erstmalige Einrichtung
- Verwenden der Quorum-Authentifizierung für Verschlüsselungsverantwortliche
- Ändern des Quorum-Mindestwerts für Verschlüsselungsverantwortliche

Überblick über die Quorum-Authentifizierung

Die folgenden Schritte fassen die Verfahren zur Quorum-Authentifizierung zusammen. Die exakten Schritte und Tools finden Sie unter Verwenden der Quorum-Authentifizierung für Verschlüsselungsverantwortliche.

1. Jeder HSM-Benutzer erstellt einen asymmetrischen Schlüssel zum Signieren. Er oder sie tut dies außerhalb des HSMs und achtet darauf, den Schlüssel angemessen zu schützen.
2. Jeder HSM-Benutzer meldet sich beim HSM an und registriert den öffentlichen Teil seines Signierungsschlüssels (den öffentlichen Schlüssel) beim HSM.
3. Wenn ein HSM-Benutzer eine Quorum-kontrollierte Operation durchführen möchte, meldet er sich beim HSM an und erhält ein Quorum-Token.
4. Der HSM-Benutzer gibt das Quorum-Token an einen oder mehrere andere HSM-Benutzer weiter und bittet um deren Zustimmung.
5. Die anderen HSM-Benutzer genehmigen, indem sie mit ihren Schlüsseln das Quorum-Token kryptographisch signieren. Dies geschieht außerhalb des HSMs.
6. Wenn der HSM-Benutzer über die erforderliche Anzahl von Genehmigungen verfügt, meldet er sich beim HSM an und gibt das Quorum-Token und die Genehmigungen (Signaturen) an das HSM weiter.
7. Das HSM verwendet die registrierten, öffentlichen Schlüssel jedes Unterzeichners, um die Signaturen zu verifizieren. Wenn die Signaturen gültig sind, genehmigt das HSM das Token.
8. Der HSM-Benutzer kann nun eine quorum-kontrollierte Operation durchführen.

Weitere Details zur Quorum-Authentifizierung

Beachten Sie die folgenden, zusätzlichen Informationen zur Verwendung der Quorum-Authentifizierung in AWS CloudHSM.

- Ein HSM-Benutzer kann sein eigenes Quorum-Token signieren, d. h. der anfordernde Benutzer kann eine der erforderlichen Genehmigungen für die Quorum-Authentifizierung bereitstellen.
- Sie wählen die Mindestzahl der Quorum-Genehmiger für die Quorum-kontrollierten Operationen aus. Die kleinste Zahl, die Sie wählen können, ist zwei (2), und die größte Zahl, die Sie wählen können, ist acht (8).
- Das HSM kann bis zu 1024 Quorum-Token speichern. Wenn das HSM beim Versuch, ein neues zu erstellen, bereits 1024 Token hat, löscht das HSM eines der abgelaufenen Token. Standardmäßig verfallen Token zehn Minuten nach ihrer Erstellung.

- Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für die Zwei-Faktor-Authentifizierung (2FA). Weitere Informationen zur Verwendung von Quorum-Authentifizierung und 2FA finden Sie unter Quorum-Authentifizierung und 2FA.

Verwenden der Quorum-Authentifizierung für Verschlüsselungsverantwortliche: Erstmalige Einrichtung

In den folgenden Themen werden die Schritte beschrieben, die Sie zum Konfigurieren Ihres Hardware-Sicherheitsmoduls (HSM) durchführen müssen, damit Verschlüsselungsverantwortliche (COs, Crypto Officers) Quorum-Authentifizierung verwenden können. Sie müssen diese Schritte nur einmal ausführen, wenn Sie die Quorum-Authentifizierung für COs zum ersten Mal konfigurieren. Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Verwenden der Quorum-Authentifizierung für Verschlüsselungsverantwortliche fort.

Themen

- Voraussetzungen
- Erstellen und Registrieren eines Schlüssels für das Signieren
- Setzten des Quorum-Mindestwerts für das HSM

Voraussetzungen

Um dieses Beispiel zu verstehen, sollten Sie mit dem cloudhsm_mgmt_util-Kommandozeilen-Tool (CMU) vertraut sein. In diesem Beispiel verfügt der AWS CloudHSM-Cluster über zwei HSMs, die jeweils die gleichen COs aufweisen, wie die folgende Ausgabe des Befehls listUsers zeigt. Weitere Informationen zum Erstellen von Benutzern finden Sie unter Verwalten von HSM-Benutzern.

```
aws-cloudhsm>listUsers
Users on server 0(10.0.2.14):
Number of users found:7

    User Id             User Type       User Name                      MofnPubKey
  LoginFailureCnt         2FA
        1               PRECO           admin                               NO
          0               NO
        2               AU              app_user                            NO
          0               NO
        3               CO              officer1                            NO
          0               NO
```

```
         4               CO              officer2                                NO
           0                NO
         5               CO              officer3                                NO
           0                NO
         6               CO              officer4                                NO
           0                NO
         7               CO              officer5                                NO
           0                NO
Users on server 1(10.0.1.4):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1               PRECO           admin                                   NO
           0                NO
         2               AU              app_user                                NO
           0                NO
         3               CO              officer1                                NO
           0                NO
         4               CO              officer2                                NO
           0                NO
         5               CO              officer3                                NO
           0                NO
         6               CO              officer4                                NO
           0                NO
         7               CO              officer5                                NO
           0                NO
```

Erstellen und Registrieren eines Schlüssels für das Signieren

Um die Quorumauthentifizierung zu verwenden, muss jeder CO alle der folgenden Schritte ausführen:

Themen

- Erstellen eines RSA-Schlüsselpaares
- Erstellen und signieren Sie ein Registrierungstoken
- Registrieren Sie den öffentlichen Schlüssel beim HSM

Erstellen eines RSA-Schlüsselpaares

Es gibt viele verschiedene Möglichkeiten, ein Schlüsselpaar zu erstellen und zu schützen. In den folgenden Beispielen wird gezeigt, wie dies mit OpenSSL durchgeführt wird.

Example – Erstellen eines privaten Schlüssels mit OpenSSL

Im folgenden Beispiel wird gezeigt, wie OpenSSL verwendet wird, um einen 2048-Bit-RSA-Schlüssel zu erstellen, der durch eine Pass-Phrase geschützt ist. Ersetzen Sie, um dieses Beispiel zu verwenden, *officer1.key* durch den Namen der Datei, in der Sie den Schlüssel speichern möchten.

```
$ openssl genrsa -out officer1.key -aes256 2048
Generating RSA private key, 2048 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
Enter pass phrase for officer1.key:
Verifying - Enter pass phrase for officer1.key:
```

Generieren eines öffentlichen Schlüssels mit dem privaten Schlüssel, den Sie gerade erstellt haben.

Example – Erstellen Sie einen öffentlichen Schlüssel mit OpenSSL

Das folgende Beispiel zeigt, wie Sie OpenSSL verwenden, um einen öffentlichen Schlüssel aus dem privaten Schlüssel zu erstellen, den Sie gerade erstellt haben.

```
$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
Enter pass phrase for officer1.key:
writing RSA key
```

Erstellen und signieren Sie ein Registrierungstoken

Sie erstellen ein Token und signieren es mit dem privaten Schlüssel, den Sie gerade im vorherigen Schritt generiert haben.

Example – Erstellen Sie ein Token

Das Registrierungstoken ist nur eine Datei mit beliebigen zufälligen Daten, die die maximale Größe von 245 Byte nicht überschreiten. Sie signieren das Token mit dem privaten Schlüssel, um nachzuweisen, dass Sie Zugriff auf den privaten Schlüssel haben. Der folgende Befehl verwendet Echo, um eine Zeichenfolge in eine Datei umzuleiten.

```
$ echo "token to be signed" > officer1.token
```

Signieren Sie das Token und speichern Sie es in einer Signaturdatei. Sie benötigen das signierte Token, das unsignierte Token und den öffentlichen Schlüssel, um den CO als MoFN-Benutzer beim HSM zu registrieren.

Example – Signieren Sie das Token

Verwenden Sie OpenSSL und den privaten Schlüssel, um das Registrierungstoken zu signieren und die Signaturdatei zu erstellen.

```
$ openssl dgst -sha256 \
    -sign officer1.key \
    -out officer1.token.sig officer1.token
```

Registrieren Sie den öffentlichen Schlüssel beim HSM

Nach dem Erstellen eines Schlüssels muss der CO den öffentlichen Teil des Schlüssels (den öffentlichen Schlüssel) beim HSM registrieren.

So registrieren Sie einen öffentlichen Schlüssel bei dem HSM

1. Starten Sie das Befehlszeilen-Toolcloudhsm_mgmt_util mit folgendem Befehl.

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. Verwenden Sie den Befehl loginHSM, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter ???.
3. Verwenden Sie den Befehl registerQuorumPubKey, um den öffentlichen Schlüssel zu registrieren. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl help registerQuorumPubKey ausführen.

Example – Registrieren eines öffentlichen Schlüssels beim HSM

Das folgende Beispiel zeigt, wie Sie mit dem Befehl registerQuorumPubKey im cloudhsm_mgmt_util-Befehlszeilen-Tool den öffentlichen Schlüssel eines COs beim HSM registrieren. Der CO muss bei dem HSM angemeldet sein, um diesen Befehl zu verwenden. Ersetzen Sie diese Werte durch Ihre eigenen Werte:

```
aws-cloudhsm> registerQuorumPubKey CO <officer1> <officer1.token> <officer1.token.sig>
 <officer1.pub>
```

```
************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?y
registerQuorumPubKey success on server 0(10.0.2.14)
```

<officer1.token>

Der Pfad zu einer Datei, die ein unsigniertes Registrierungstoken enthält. Kann beliebige Daten mit einer maximalen Dateigröße von 245 Byte enthalten.

Erforderlich: Ja

<officer1.token.sig>

Der Pfad zu einer Datei, die den mit dem SHA256_PKCS-Mechanismus signierten Hash-Wert des Registrierungs-Tokens enthält.

Erforderlich: Ja

<officer1.pub>

Der Pfad zu der Datei, die den öffentlichen Schlüssel eines asymmetrischen RSA-2048-Schlüsselpaars enthält. Verwenden Sie den privaten Schlüssel, um das Registrierungstoken zu signieren.

Erforderlich: Ja

Nachdem alle COs ihre öffentlichen Schlüssel registriert haben, zeigt die Ausgabe des listUsers-Befehls dies in der `MofnPubKey`-Spalte an, wie im folgenden Beispiel veranschaulicht.

```
aws-cloudhsm>listUsers
Users on server 0(10.0.2.14):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
```

```
         1              PRECO           admin                                NO
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              officer1                            YES
           0               NO
         4              CO              officer2                            YES
           0               NO
         5              CO              officer3                            YES
           0               NO
         6              CO              officer4                            YES
           0               NO
         7              CO              officer5                            YES
           0               NO
Users on server 1(10.0.1.4):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                NO
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              officer1                            YES
           0               NO
         4              CO              officer2                            YES
           0               NO
         5              CO              officer3                            YES
           0               NO
         6              CO              officer4                            YES
           0               NO
         7              CO              officer5                            YES
           0               NO
```

Setzten des Quorum-Mindestwerts für das HSM

Zum Verwenden der Quorum-Authentifizierung für COs muss sich ein CO bei dem HSM anmelden und dann den Quorum-Mindestwert, der auch als m_value bezeichnet wird, festlegen. Dies ist die Mindestanzahl von CO-Genehmigungen, die erforderlich sind, um HSM-Benutzermanagement-Vorgänge auszuführen. Alle COs im HSM können den Quorum-Mindestwert festlegen, einschließlich COs, die keinen Schlüssel für das Signieren registriert haben. Sie können den Quorum-Mindestwert jederzeit ändern. Weitere Informationen finden Sie unter Ändern Sie den Mindestwert.

So legen Sie den Quorum-Mindestwert im HSM fest

1. Starten Sie das Befehlszeilen-Toolcloudhsm_mgmt_util mit folgendem Befehl.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. Verwenden Sie den Befehl loginHSM, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter ???.
3. Verwenden Sie den setMValue-Befehl, um den Quorum-Mindestwert festzulegen. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl help setMValue ausführen.

Example – Setzen des Quorum-Mindestwerts für das HSM

In diesem Beispiel wird ein Quorum-Mindestwert von zwei verwendet. Sie können einen Wert zwischen zwei (2) und acht (8) wählen, bis zur Gesamtzahl der COs auf dem HSM. In diesem Beispiel hat das HSM sechs COs, also ist der maximal mögliche Wert sechs.

Um den folgenden Beispielbefehl zu verwenden, ersetzen Sie die letzte Zahl (*2*) durch den bevorzugten Quorum-Mindestwert.

```
aws-cloudhsm>setMValue 3 2
************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)?y
Setting M Value(2) for 3 on 2 nodes
```

Im vorhergehenden Beispiel bestimmt die erste Zahl (3) den HSM-Service, dessen Quorum-Mindestwert Sie festlegen.

In der folgenden Tabelle sind die HSM-Dienstkennungen zusammen mit ihren Namen, Beschreibungen und den Befehlen aufgeführt, die im Dienst enthalten sind.

Dienstkennung	Service-Name	Service description (Service-Beschreibung)	HSM-Befehle
3	USER_MGMT	HSM-Benutzerverwaltung	• createUser • deleteUser • changePswd (gilt nur, wenn das Passwort eines anderen HSM-Benutzers geändert wird)
4	MISC_CO	Diverser CO-Service	• setMValue

Um den Quorum-Mindestwert für einen Service abzurufen, verwenden Sie den getMValue-Befehl wie im folgenden Beispiel.

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

Die Ausgabe aus dem vorausgehenden Befehl getMValue zeigt, dass der Quorum-Mindestwert für HSM-Benutzermanagement-Vorgänge (Service 3) jetzt zwei ist.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Verwenden der Quorum-Authentifizierung für Verschlüsselungsverantwortliche fort.

Verwenden der Quorum-Authentifizierung für Verschlüsselungsverantwortliche

Ein Verschlüsselungsverantwortlicher (CO, Crypto Officer) kann eine Quorum-Authentifizierung für die folgenden Operationen auf dem HSM konfigurieren:

- Erstellen von HSM-Benutzern
- Löschen von HSM-Benutzern
- Ändern des Passworts eines anderen HSM-Benutzers

Nachdem das HSM für die Quorum-Authentifizierung konfiguriert wurde, können COs selbständig HSM-Benutzermanagement-Vorgänge durchführen. Das folgende Beispiel zeigt die Ausgabe, die angezeigt wird, wenn ein CO versucht, einen neuen Benutzer auf dem HSM anzulegen. Der Befehl schlägt mit einem `RET_MXN_AUTH_FAILED`-Fehler fehl. Dies weist darauf hin, dass die Quorum-Authentifizierung nicht möglich war.

```
aws-cloudhsm>createUser CU user1 password
**********CAUTION**********
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
************************************************************

Do you want to continue(y/n)?y
Creating User user1(CU) on 2 nodes
createUser failed: RET_MXN_AUTH_FAILED
creating user on server 0(10.0.2.14) failed

Retry/Ignore/Abort?(R/I/A):A
```

Zum Ausführen eines HSM-Benutzermanagement-Vorgangs muss ein CO folgende Aufgaben erledigen:

1. Abrufen eines Quorum-Tokens.
2. Erhalten von Genehmigungen (Signaturen) von anderen COs.
3. Genehmigen des Tokens im HSM.
4. Durchführen des HSM-Benutzermanagement-Vorgangs.

Wenn Sie das HSM noch nicht für die Quorum-Authentifizierung für COs konfiguriert haben, holen Sie dies jetzt nach. Weitere Informationen finden Sie unter Erstmalige Einrichtung.

Abrufen eines Quorum-Tokens

Zunächst muss der CO das cloudhsm_mgmt_util-Befehlszeilen-Tool verwenden, um ein Quorum-Token anzufordern.

So fordern Sie ein Quorum-Token an

1. Starten Sie das Befehlszeilen-Toolcloudhsm_mgmt_util mit folgendem Befehl.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. Verwenden Sie den Befehl loginHSM, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter ???.
3. Verwenden Sie zum Abrufen eines Quorum-Tokens den Befehl getToken. Weitere Informationen finden Sie im folgenden Beispiel. Alternativ können Sie auch den Befehl help getToken ausführen.

Example – Abrufen eines Quorum-Tokens

In diesem Beispiel wird ein Quorum-Token für den CO mit dem Benutzernamen „officer1“ abgerufen und in einer Datei namens `officer1.token` gespeichert. Zum Verwenden des Beispielbefehls ersetzen Sie diese Werte durch Ihre eigenen:

- *`officer1`* – Der Name des COs, der das Token erhält. Dies muss derselbe CO sein, der am HSM angemeldet ist und diesen Befehl ausführt.
- *`officer1.token`* – Der Name der Datei, in der das Quorum-Token gespeichert wird.

Beim folgenden Befehl identifiziert `3` den Service, für den Sie das Token verwenden können, das Sie abrufen. In diesem Fall ist das Token für die HSM-Benutzermanagement-Vorgänge (Service 3). Weitere Informationen finden Sie unter Setzten des Quorum-Mindestwerts für das HSM.

```
aws-cloudhsm>getToken 3 officer1 officer1.token
getToken success on server 0(10.0.2.14)
Token:
Id:1
Service:3
Node:1
Key Handle:0
User:officer1
getToken success on server 1(10.0.1.4)
Token:
Id:1
Service:3
Node:0
Key Handle:0
User:officer1
```

Erhalten der Signaturen von genehmigenden COs

Ein CO mit einem Quorum-Token muss sich das Token von anderen COs genehmigen lassen. Für die Genehmigung nutzen die anderen COs Ihren Signaturschlüssel zur kryptografischen Token-Signierung. Dies geschieht außerhalb des HSMs.

Es gibt viele verschiedene Möglichkeiten, ein Token zu signieren. Im folgenden Beispiel wird gezeigt, wie dies mit OpenSSL durchgeführt wird. Wenn Sie ein anderes Signatur-Tool verwenden, müssen Sie sicherstellen, dass dieses den privaten Schlüssel (Signaturschlüssel) des COs zur Signierung des SHA-256-Digests des Tokens verwendet.

Example – Erhalten der Signaturen von den genehmigenden COs

In diesem Beispiel benötigt der CO mit dem Token (officer1) mindestens zwei Genehmigungen. Das folgende Beispiel zeigt, wie zwei COs OpenSSL zur kryptografischen Signierung des Tokens einsetzen können.

Beim ersten Befehl signiert „officer1“ das eigene Token. Wenn Sie die folgenden Beispielbefehle verwenden, ersetzen Sie diese Werte durch Ihre eigenen:

- *officer1.key* und *officer2.key* – Der Name der Datei, die den Signaturschlüssel des COs enthält.
- *officer1.token.sig1* und *officer1.token.sig2* – Der Name der Datei, in der die Signatur gespeichert wird. Stellen Sie sicher, dass jede Signatur in einer anderen Datei gespeichert wird.
- *officer1.token* – Der Name der Datei, die das Token enthält, das vom CO signiert wird.

```
$ openssl dgst -sha256 -sign officer1.key -out officer1.token.sig1 officer1.token
Enter pass phrase for officer1.key:
```

Beim folgenden Befehl signiert „officer2“ dasselbe Token.

```
$ openssl dgst -sha256 -sign officer2.key -out officer1.token.sig2 officer1.token
Enter pass phrase for officer2.key:
```

Genehmigen des signierten Tokens auf dem HSM

Nachdem ein CO die minimale Anzahl an Genehmigungen (Signaturen) von anderen COs erhalten hat, muss er das signierte Token auf dem HSM genehmigen.

So genehmigen Sie das signierte Token auf dem HSM

1. Erstellen Sie eine Token-Genehmigungs-Datei. Weitere Informationen finden Sie im folgenden Beispiel.
2. Starten Sie das Befehlszeilen-Toolcloudhsm_mgmt_util mit folgendem Befehl.

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

3. Verwenden Sie den Befehl loginHSM, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter ???.
4. Verwenden Sie den Befehl approveToken zum Genehmigen des signierten Tokens, indem Sie die Token-Genehmigungs-Datei übergeben. Weitere Informationen finden Sie im folgenden Beispiel.

Example – Erstellen einer Token-Genehmigungs-Datei und Genehmigen des signierten Tokens auf dem HSM

Die Token-Genehmigungs-Datei ist eine Textdatei in einem Format, das für das HSM erforderlich ist. Die Datei enthält Informationen über das Token, die Genehmigenden und dessen Signaturen. Im Folgenden wird ein Beispiel einer Token-Genehmigungs-Datei gezeigt.

```
# For "Multi Token File Path", type the path to the file that contains
# the token. You can type the same value for "Token File Path", but
# that's not required. The "Token File Path" line is required in any
# case, regardless of whether you type a value.
Multi Token File Path = officer1.token;
Token File Path = ;

# Total number of approvals
Number of Approvals = 2;

# Approver 1
# Type the approver's type, name, and the path to the file that
# contains the approver's signature.
Approver Type = 2; # 2 for CO, 1 for CU
Approver Name = officer1;
Approval File = officer1.token.sig1;

# Approver 2
# Type the approver's type, name, and the path to the file that
```

```
# contains the approver's signature.
Approver Type = 2; # 2 for CO, 1 for CU
Approver Name = officer2;
Approval File = officer1.token.sig2;
```

Nach dem Erstellen der Token-Genehmigungsdatei verwendet der CO das cloudhsm_mgmt_util-Befehlszeilen-Tool für die Anmeldung am HSM. Anschließend genehmigt der CO das Token mit dem Befehl approveToken, wie im folgenden Beispiel gezeigt. Ersetzen Sie *approval.txt* durch den Namen der Token-Genehmigungs-Datei.

```
aws-cloudhsm>approveToken approval.txt
approveToken success on server 0(10.0.2.14)
approveToken success on server 1(10.0.1.4)
```

Wurde der Befehl erfolgreich ausgeführt, hat das HSM das Quorum-Token genehmigt. Zum Prüfen des Token-Status nutzen Sie den Befehl listTokens, wie im folgenden Beispiel gezeigt. Die Befehlsausgabe zeigt, dass das Token über die erforderliche Anzahl von Genehmigungen verfügt.

Die Token-Gültigkeitsdauer gibt an, wie lange das Token auf dem HSM erhalten bleibt. Sie können das Token auch nach Ablauf der Token-Gültigkeitsdauer (null Sekunden) noch verwenden.

```
aws-cloudhsm>listTokens

====================
    Server 0(10.0.2.14)
====================
-------- Token - 0 ----------
Token:
Id:1
Service:3
Node:1
Key Handle:0
User:officer1
Token Validity: 506 sec
Required num of approvers : 2
Current num of approvals : 2
Approver-0: officer1
Approver-1: officer2
Num of tokens = 1

====================
    Server 1(10.0.1.4)
```

```
====================
-------- Token - 0 ----------
Token:
Id:1
Service:3
Node:0
Key Handle:0
User:officer1
Token Validity: 506 sec
Required num of approvers : 2
Current num of approvals : 2
Approver-0: officer1
Approver-1: officer2
Num of tokens = 1

listTokens success
```

Verwenden des Tokens für Benutzermanagement-Vorgänge

Nachdem ein CO über ein Token mit ausreichender Anzahl an Genehmigungen verfügt (wie im vorherigen Abschnitt gezeigt), kann er eine der folgenden HSM-Benutzermanagement-Vorgänge ausführen:

- Erstellen eines HSM-Benutzers mit dem Befehl createUser
- Löschen eines HSM-Benutzers mit dem Befehl deleteUser
- Ändern des Passworts eines anderen HSM-Benutzers mit dem Befehl changePswd

Weitere Informationen zur Verwendung dieser Befehle finden Sie unter Verwalten von HSM-Benutzern.

Der CO kann das Token nur für einen Vorgang nutzen. Wurde dieser erfolgreich ausgeführt, verliert das Token seine Gültigkeit. Um einen weiteren HSM-Benutzermanagement-Vorgang auszuführen, benötigt der CO ein neues Quorum-Token sowie neue Signaturen von den Genehmigenden. Zudem muss er das neue Token auf dem HSM genehmigen.

> **Note**
>
> Das MoFN-Token ist nur gültig, solange Ihre aktuelle Anmeldesitzung geöffnet ist. Wenn Sie sich von cloudhsm_mgmt_util abmelden oder die Netzwerkverbindung unterbrochen wird, ist das Token nicht mehr gültig. Ebenso kann ein autorisiertes Token nur innerhalb von

cloudhsm_mgmt_util verwendet werden, es kann nicht zur Authentifizierung in einer anderen Anwendung verwendet werden.

Beim folgenden Beispielbefehl erstellt der CO einen neuen Benutzer auf dem HSM.

```
aws-cloudhsm>createUser CU user1 password
*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
Creating User user1(CU) on 2 nodes
```

Nachdem der vorherige Befehl erfolgreich durchgeführt wurde, zeigt der nachfolgende Befehl listUsers den neuen Benutzer an.

```
aws-cloudhsm>listUsers
Users on server 0(10.0.2.14):
Number of users found:8

    User Id             User Type       User Name                          MofnPubKey
  LoginFailureCnt         2FA
        1               PCO             admin                               NO
          0               NO
        2               AU              app_user                            NO
          0               NO
        3               CO              officer1                           YES
          0               NO
        4               CO              officer2                           YES
          0               NO
        5               CO              officer3                           YES
          0               NO
        6               CO              officer4                           YES
          0               NO
        7               CO              officer5                           YES
          0               NO
        8               CU              user1                               NO
          0               NO
```

```
Users on server 1(10.0.1.4):
Number of users found:8

    User Id             User Type       User Name                          MofnPubKey
  LoginFailureCnt         2FA
        1               PCO             admin                               NO
          0               NO
        2               AU              app_user                            NO
          0               NO
        3               CO              officer1                           YES
          0               NO
        4               CO              officer2                           YES
          0               NO
        5               CO              officer3                           YES
          0               NO
        6               CO              officer4                           YES
          0               NO
        7               CO              officer5                           YES
          0               NO
        8               CU              user1                               NO
          0               NO
```

Wenn der CO versucht, einen weiteren HSM-Benutzermanagement-Vorgang durchzuführen, schlägt dieser aufgrund eines Quorum-Authentifizierungsfehlers fehl, wie im folgenden Beispiel gezeigt.

```
aws-cloudhsm>deleteUser CU user1
Deleting user user1(CU) on 2 nodes
deleteUser failed: RET_MXN_AUTH_FAILED
deleteUser failed on server 0(10.0.2.14)

Retry/rollBack/Ignore?(R/B/I):I
deleteUser failed: RET_MXN_AUTH_FAILED
deleteUser failed on server 1(10.0.1.4)

Retry/rollBack/Ignore?(R/B/I):I
```

Der Befehl listTokens zeigt an, dass der CO nicht über genehmigte Tokens verfügt, wie im folgenden Beispiel gezeigt. Um einen weiteren HSM-Benutzermanagement-Vorgang auszuführen, benötigt der CO ein neues Quorum-Token sowie neue Signaturen von den Genehmigenden. Zudem muss er das neue Token auf dem HSM genehmigen.

```
aws-cloudhsm>listTokens
```

```
====================
    Server 0(10.0.2.14)
====================
Num of tokens = 0

====================
    Server 1(10.0.1.4)
====================
Num of tokens = 0

listTokens success
```

Ändern des Quorum-Mindestwerts für Verschlüsselungsverantwortliche

Nachdem Sie den Quorum-Mindestwert festgelegt haben, sodass Verschlüsselungsverantwortliche (COs) die Quorum-Authentifizierung verwenden können, können Sie den Quorum-Mindestwert ändern. Das HSM lässt nur eine Änderung des Quorum-Mindestwerts zu, wenn die Anzahl der Genehmiger gleich oder höher als der aktuelle Quorum-Mindestwert ist. Beispiel: Wenn der Quorum-Mindestwert 2 beträgt, genehmigen mindestens zwei COs die Änderung des Quorum-Mindestwerts.

Für die Quorum-Genehmigung zum Ändern des Quorum-Mindestwerts benötigen Sie ein Quorum-Token für den Befehl setMValue (Service 4). Um ein Quorum-Token für den Befehl setMValue (Service 4) zu erhalten, muss der Quorum-Mindestwert für Service 4 größer als 1 sein. Das bedeutet, dass Sie vor dem Ändern des Quorum-Mindestwerts für COs (Service 3) möglicherweise den Quorum-Mindestwert für Service 4 ändern müssen.

In der folgenden Tabelle sind die HSM-Dienstkennungen zusammen mit ihren Namen, Beschreibungen und den Befehlen aufgeführt, die im Dienst enthalten sind.

Dienstkennung	Service-Name	Service description (Service-Beschreibung)	HSM-Befehle
3	USER_MGMT	HSM-Benutzerverwaltung	• createUser • deleteUser • changePswd (gilt nur, wenn das Passwort eines

Dienstkennung	Service-Name	Service description (Service-Beschreibung)	HSM-Befehle
			anderen HSM-Benutzers geändert wird)
4	`MISC_CO`	Diverser CO-Service	• setMValue

So ändern Sie den Quorum-Mindestwert für Verschlüsselungsverantwortliche

1. Starten Sie das Befehlszeilen-Toolcloudhsm_mgmt_util mit folgendem Befehl.

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. Verwenden Sie den Befehl loginHSM, um sich bei den HSMs als CO anzumelden. Weitere Informationen finden Sie unter ???.
3. Verwenden Sie den Befehl getMValue, um den Quorum-Mindestwert für Service 3 abzurufen. Weitere Informationen finden Sie im folgenden Beispiel.
4. Verwenden Sie den Befehl getMValue, um den Quorum-Mindestwert für Service 4 abzurufen. Weitere Informationen finden Sie im folgenden Beispiel.
5. Wenn der Quorum-Mindestwert für Service 4 niedriger ist als der Wert für Service 3, verwenden Sie den Befehl setMValue, um den Wert für Service 4 zu ändern. Ändern Sie den Wert für Service 4 in einen Wert, der gleich oder höher als der Wert für Service 3 ist. Weitere Informationen finden Sie im folgenden Beispiel.
6. Rufen Sie ein Quorum-Token ab und geben Sie Service 4 als den Service an, für den Sie das Token verwenden können.
7. Erhalten von Genehmigungen (Signaturen) von anderen COs.
8. Genehmigen des Tokens im HSM.
9. Verwenden Sie den Befehl setMValue, um den Quorum-Mindestwert für Service 3 zu ändern (Benutzerverwaltungs-Vorgänge, die von COs ausgeführt werden).

Example – Abrufen von Quorum-Mindestwerten und Ändern des Werts für Service 4

Das folgende Beispiel zeigt, dass der Quorum-Mindestwert für Service 3 derzeit 2 ist.

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

Das folgende Beispiel zeigt, dass der Quorum-Mindestwert für Service 4 derzeit 1 ist.

```
aws-cloudhsm>getMValue 4
MValue of service 4[MISC_CO] on server 0 : [1]
MValue of service 4[MISC_CO] on server 1 : [1]
```

Um den Quorum-Mindestwert für Service 4 zu ändern, verwenden Sie den Befehl setMValue und legen Sie einen Wert fest, der gleich oder höher als der Wert für Service 3 ist. Im folgenden Beispiel wird der Quorum-Mindestwert für Service 4 auf 2 festgelegt. Dies ist der gleiche Wert wie für Service 3.

```
aws-cloudhsm>setMValue 4 2
************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?y
Setting M Value(2) for 4 on 2 nodes
```

Die folgenden Befehle zeigen, dass der Quorum-Mindestwert für Service 3 und Service 4 jetzt 2 ist.

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

```
aws-cloudhsm>getMValue 4
MValue of service 4[MISC_CO] on server 0 : [2]
MValue of service 4[MISC_CO] on server 1 : [2]
```

Verwalten von Schlüsseln in AWS CloudHSM

Verwenden Sie in AWS CloudHSM eine der folgenden Optionen, um Schlüssel auf den HSMs in Ihrem Cluster zu verwalten:

- PKCS #11-Bibliothek
- JCE-Anbieter
- CNG- und KSP-Anbieter
- CloudHSM-CLI

Bevor Sie Schlüssel verwalten können, müssen Sie sich mit dem Benutzernamen und dem Passwort eines Crypto-Benutzers (CU) am HSM anmelden. Nur ein CU kann einen Schlüssel erstellen. Der CU, die einen Schlüssel erstellt, besitzt und verwaltet diesen Schlüssel.

Themen

Schlüsselsynchronisations- und Haltbarkeitseinstellungen in AWS CloudHSM

In diesem Thema werden Einstellungen in AWS CloudHSM für die Schlüsselsynchronisierung, allgemeine Probleme, mit denen Kunden bei der Arbeit mit Schlüsseln in einem Cluster konfrontiert sind, sowie Strategien zur Erhöhung der Haltbarkeit von Schlüsseln beschrieben.

Themen

Konzepte

Token-Schlüssel

Persistente Schlüssel, die Sie beim Generieren, Importieren oder Entpacken von Schlüsseln erstellen. AWS CloudHSM synchronisiert Token-Schlüssel in einem Cluster.

Sitzungsschlüssel

Kurzlebige Schlüssel, die nur auf einem Hardwaresicherheitsmodul (HSM) im Cluster vorhanden sind. AWS CloudHSM synchronisiert keine Sitzungsschlüssel innerhalb eines Clusters.

Clientseitige Schlüsselsynchronisierung

Ein clientseitiger Prozess, der Tokenschlüssel klont, die Sie beim Generieren, Importieren oder Entpacken von Schlüsseln erstellen. Sie können Token-Schlüssel langlebiger machen, indem Sie einen Cluster mit mindestens zwei HSMs ausführen.

Serverseitige Schlüsselsynchronisierung

Klont in regelmäßigen Abständen Schlüssel für jedes HSM im Cluster. Erfordert keine Verwaltung.

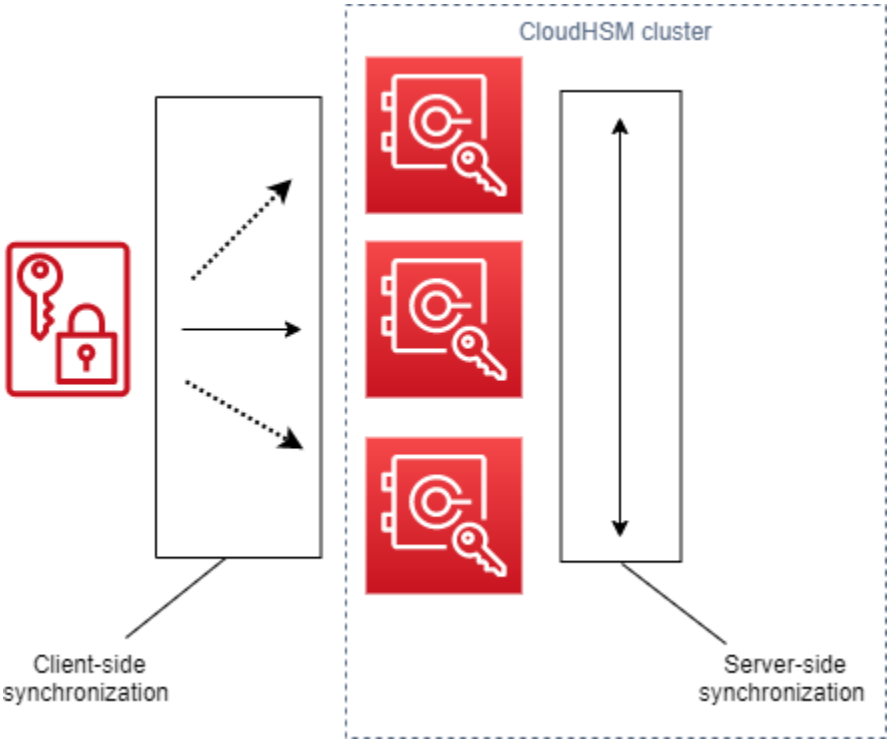
Einstellungen für die Haltbarkeit von Client-Schlüsseln

Einstellungen, die Sie auf dem Client konfigurieren und die sich auf die Haltbarkeit von Schlüsseln auswirken. Diese Einstellungen funktionieren in Client-SDK 5 und Client-SDK 3 unterschiedlich.

- Verwenden Sie im Client-SDK 5 diese Einstellung, um einen einzelnen HSM-Cluster auszuführen.
- Verwenden Sie im Client-SDK 3 diese Einstellung, um die Anzahl der HSMs anzugeben, die für eine erfolgreiche Schlüsselerstellung erforderlich sind.

Grundlegendes zur Schlüsselsynchronisierung

AWS CloudHSM verwendet die Schlüsselsynchronisierung, um Tokenschlüssel für alle HSMs in einem Cluster zu klonen. Sie erstellen Token-Schlüssel als persistente Schlüssel bei der Schlüsselgenerierung, beim Import oder beim Entpacken. Um diese Schlüssel über den Cluster zu verteilen, bietet CloudHSM die clientseitige und serverseitige Schlüssel-Synchronisierung.



Das Ziel der Schlüsselsynchronisierung – sowohl serverseitig als auch clientseitig – besteht darin, neue Schlüssel so schnell wie möglich im Cluster zu verteilen, nachdem Sie sie erstellt haben. Dies ist wichtig, da die nachfolgenden Aufrufe, die Sie zur Verwendung neuer Schlüssel tätigen, an jedes verfügbare HSM im Cluster weitergeleitet werden können. Wenn der Aufruf, den Sie tätigen, ohne den Schlüssel an ein HSM weitergeleitet wird, schlägt der Aufruf fehl. Sie können diese Art von Fehlern beheben, indem Sie angeben, dass Ihre Anwendungen nachfolgende Aufrufe wiederholen, die nach der Schlüsselerstellung ausgeführt wurden. Die für die Synchronisierung benötigte Zeit kann je nach der Arbeitslast Ihres Clusters und anderen immateriellen Vermögenswerten variieren. Verwenden Sie CloudWatch Metriken, um zu bestimmen, wann Ihre Anwendung in dieser Typsituation verwenden soll. Weitere Informationen finden Sie unter CloudWatch Metriken.

Die Herausforderung bei der Schlüsselsynchronisierung in einer Cloud-Umgebung ist die Haltbarkeit der Schlüssel. Sie erstellen Schlüssel auf einem einzigen HSM und beginnen oft sofort, diese Schlüssel zu verwenden. Wenn das HSM, auf dem Sie Schlüssel erstellen, ausfällt, bevor die Schlüssel auf ein anderes HSM im Cluster geklont wurden, verlieren Sie die Schlüssel und den Zugriff auf alles, was mit den Schlüsseln verschlüsselt wurde. Um dieses Risiko zu minimieren, bieten wir Client-Synchronisierung an. Die clientseitige Synchronisation ist ein clientseitiger Prozess, bei dem die Schlüssel geklont werden, die Sie beim Generieren, Importieren oder Entpacken von Schlüsseln erstellen. Durch das Klonen von Schlüsseln während der Erstellung werden Schlüssel langlebiger. Natürlich können Sie Schlüssel in einem Cluster nicht mit einem einzigen HSM klonen. Um Schlüssel langlebiger zu machen, empfehlen wir Ihnen außerdem, Ihren Cluster so zu

konfigurieren, dass mindestens zwei HSMs verwendet werden. Mit der clientseitigen Synchronisation und einem Cluster mit zwei HSMs können Sie die Herausforderung der Schlüsselbeständigkeit in einer Cloud-Umgebung bewältigen.

Arbeiten mit den Einstellungen für die Haltbarkeit von Client-Schlüsseln

Die Schlüsselsynchronisierung ist größtenteils ein automatischer Prozess, aber Sie können die Einstellungen für die Haltbarkeit von Schlüsseln auf der Clientseite verwalten. Die clientseitigen Einstellungen für die Haltbarkeit von Schlüsseln funktionieren in Client-SDK 5 und Client-SDK 3 unterschiedlich.

- In Client-SDK 5 führen wir das Konzept der Quoren für die Schlüsselverfügbarkeit ein, bei dem Sie Cluster mit mindestens zwei HSMs ausführen müssen. Sie können die clientseitigen Einstellungen für die Haltbarkeit von Schlüsseln verwenden, um die beiden HSM-Anforderungen abzulehnen. Weitere Informationen zu Quora finden Sie unter the section called “Konzepte des Client-SDK 5”.
- Im Client-SDK 3 verwenden Sie die clientseitigen Einstellungen für die Haltbarkeit von Schlüsseln, um die Anzahl der HSMs anzugeben, auf denen die Schlüsselerstellung erfolgreich sein muss, damit der gesamte Vorgang als erfolgreich gewertet wird.

Client-SDK 5: Einstellungen für die Haltbarkeit von Client-Schlüsseln

In Client-SDK 5 ist die Schlüsselsynchronisierung ein vollautomatischer Prozess. Beim Schlüsselverfügbarkeitsquorum müssen neu erstellte Schlüssel auf zwei HSMs im Cluster vorhanden sein, bevor Ihre Anwendung den Schlüssel verwenden kann. Um das Schlüsselverfügbarkeitsquorum verwenden zu können, muss Ihr Cluster über mindestens zwei HSMs verfügen.

Wenn Ihre Clusterkonfiguration die Anforderungen an die Haltbarkeit von Schlüsseln nicht erfüllt, schlägt jeder Versuch, einen Token-Schlüssel zu erstellen oder zu verwenden, fehl und die folgende Fehlermeldung wird in den Protokollen angezeigt:

```
Key <key handle> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

Sie können die Client-Konfigurationseinstellungen verwenden, um das Quorum für die Schlüsselverfügbarkeit zu deaktivieren. Möglicherweise möchten Sie sich abmelden, um beispielsweise einen Cluster mit einem einzigen HSM auszuführen.

Konzepte des Client-SDK 5

Quorum für Schlüsselverfügbarkeit

AWS CloudHSM gibt die Anzahl der HSMs in einem Cluster an, auf denen Schlüssel vorhanden sein müssen, bevor Ihre Anwendung den Schlüssel verwenden kann. Benötigt Cluster mit mindestens zwei HSMs.

Verwaltung der Einstellungen für die Haltbarkeit von Client-Schlüsseln

Um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu verwalten, müssen Sie das Configure-Tool für Client-SDK 5 verwenden.

PKCS #11 library

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-
availability-check
```

Client-SDK 3: Einstellungen für die Haltbarkeit von Client-Schlüsseln

In Client-SDK 3 ist die Schlüsselsynchronisierung größtenteils ein automatischer Prozess, aber Sie können die Einstellungen für die Haltbarkeit von Schlüsseln des Clients verwenden, um die Haltbarkeit von Schlüsseln zu erhöhen. Sie geben die Anzahl der HSMs an, auf denen die Schlüsselerstellung erfolgreich sein muss, damit der gesamte Vorgang als erfolgreich gewertet wird. Bei der clientseitigen Synchronisation wird stets versucht, Schlüssel für jedes HSM im Cluster zu klonen, unabhängig davon, welche Einstellung Sie wählen. Ihre Einstellung erzwingt die Schlüsselerstellung für die von Ihnen angegebene Anzahl von HSMs. Wenn Sie einen Wert angeben und das System den Schlüssel nicht auf diese Anzahl von HSMs replizieren kann, entfernt das System automatisch unerwünschtes Schlüsselmaterial, und Sie können es erneut versuchen.

> **Important**
>
> Wenn Sie keine Einstellungen für die Haltbarkeit von Client-Schlüsseln festlegen (oder den Standardwert 1 verwenden), besteht die Gefahr, dass Ihre Schlüssel verloren gehen. Wenn Ihr aktuelles HSM ausfällt, bevor der serverseitige Dienst diesen Schlüssel auf ein anderes HSM geklont hat, verlieren Sie das Schlüsselmaterial.

Um die Haltbarkeit von Schlüsseln zu maximieren, sollten Sie erwägen, mindestens zwei HSMs für die clientseitige Synchronisation anzugeben. Denken Sie daran, dass unabhängig davon, wie

viele HSMs Sie angeben, die Arbeitslast auf Ihrem Cluster gleich bleibt. Bei der clientseitigen Synchronisation wird stets nach besten Kräften versucht, Schlüssel für jedes HSM im Cluster zu klonen.

Empfehlungen

- Minimum: Zwei HSMs pro Cluster
- Maximum: Eins weniger als die Gesamtzahl der HSMs in Ihrem Cluster

Wenn die clientseitige Synchronisation fehlschlägt, bereinigt der Client-Dienst alle unerwünschten Schlüssel, die möglicherweise erstellt wurden und nun unerwünscht sind. Bei dieser Bereinigung handelt es sich um eine bestmögliche Lösung, die möglicherweise nicht immer funktioniert. Wenn die Bereinigung fehlschlägt, müssen Sie möglicherweise unerwünschtes Schlüsselmaterial löschen. Weitere Informationen finden Sie unter Schlüssel-Synchronisierungsfehler.

Einrichtung der Konfigurationsdatei für die Haltbarkeit von Client-Schlüsseln

Um die Einstellungen für die Haltbarkeit von Client-Schlüsseln festzulegen, müssen Sie `cloudhsm_client.cfg` bearbeiten.

Bearbeiten der Clientkonfigurationsdatei

1. Öffnen Sie `cloudhsm_client.cfg`.

   Linux:

   ```
   /opt/cloudhsm/etc/cloudhsm_client.cfg
   ```

   Windows:

   ```
   C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
   ```

2. Fügen Sie im `client`-Knoten der Datei `create_object_minimum_nodes` hinzu und geben Sie einen Wert für die Mindestanzahl der HSMs an, auf denen AWS CloudHSM Schlüssel erfolgreich erstellen müssen, damit die Schlüsselerstellungsoperationen erfolgreich sind.

   ```
   "create_object_minimum_nodes" : 2
   ```

> **Note**
>
> Das Befehlszeilentool key_mgmt_util (KMU) bietet eine zusätzliche Einstellung für die Haltbarkeit von Client-Schlüsseln. Weitere Informationen finden Sie unter the section called “KMU- und clientseitige Schlüsselsynchronisierung”.

Konfigurationsreferenz

Dies sind die clientseitigen Synchronisierungseigenschaften, die in einem Auszug aus der `cloudhsm_client.cfg`:

```
{
    "client": {
        "create_object_minimum_nodes" : 2,
        ...
    },

    ...
}
```

create_object_minimum_nodes

Gibt die Mindestanzahl von HSMs an, die erforderlich sind, um die Schlüsselgenerierung, den Schlüsselimport oder das Entpacken von Schlüsseln als erfolgreich zu betrachten. Falls eingestellt, ist die Voreinstellung 1. Das bedeutet, dass der clientseitige Dienst bei jedem Vorgang zur Schlüsselerstellung versucht, Schlüssel für jedes HSM im Cluster zu erstellen. Um jedoch einen Erfolg zurückzugeben, muss nur ein einziger Schlüssel auf einem HSM im Cluster erstellt werden.

KMU- und clientseitige Schlüsselsynchronisierung

Wenn Sie Schlüssel mit dem Befehlszeilentool key_mgmt_util (KMU) erstellen, verwenden Sie einen optionalen Befehlszeilenparameter (`-min_srv`), um die Anzahl der HSMs zu begrenzen, auf denen Schlüssel geklont werden sollen. Wenn Sie den Befehlszeilenparameter und einen Wert in der Konfigurationsdatei angeben, wird der GRÖSSERE der beiden Werte berücksichtigt. AWS CloudHSM

Weitere Informationen finden Sie unter den folgenden Themen:

- genDSAKeyPair
- genECCKeyPair
- genRSAKeyPair
- genSymKey
- importPrivateKey
- importPubKey
- imSymKey
- insertMaskedObject
- unWrapKey

Synchronisieren von Schlüsseln zwischen geklonten Clustern

Die clientseitige und serverseitige Synchronisation dient nur der Synchronisierung von Schlüsseln innerhalb desselben Clusters. Wenn Sie eine Sicherung eines Clusters in eine andere Region kopieren, können Sie den SyncKey-Befehl der cloudhsm_mgmt_util (CMU) verwenden, um Schlüssel zwischen Clustern zu synchronisieren. Sie können geklonte Cluster für regionsübergreifende Redundanz oder zur Vereinfachung Ihres Disaster Recovery-Prozesses verwenden. Weitere Informationen finden Sie unter syncKey.

AES-Schlüssel einpacken AWS CloudHSM

In diesem Thema werden die Optionen für das Umschließen von AES-Schlüsseln beschrieben AWS CloudHSM. Die AES Key Wrapping verwendet einen AES-Schlüssel (den Verschlüsselungsschlüssel), um einen anderen Schlüssel eines beliebigen Typs (den Zielschlüssel) zu verschlüsseln. Sie verwenden die Verschlüsselung, um gespeicherte Schlüssel zu schützen oder Schlüssel über unsichere Netzwerke zu übermitteln.

Themen

- Unterstützte Algorithmen
- Verwenden Sie den AES-Schlüsselumbruch AWS CloudHSM

Unterstützte Algorithmen

AWS CloudHSM bietet drei Optionen für das Umschließen von AES-Schlüsseln, die jeweils darauf basieren, wie der Zielschlüssel vor dem Umbruch aufgefüllt wird. Das Padding erfolgt automatisch

in Übereinstimmung mit dem von Ihnen verwendeten Algorithmus, wenn Sie die Verschlüsselung aufrufen. In der folgenden Tabelle sind die unterstützten Algorithmen und die zugehörigen Details aufgeführt, um Ihnen bei der Auswahl eines geeigneten Verschlüsselungsmechanismus für Ihre Anwendung zu helfen.

AES-Verschlüsselungsalgorithmus	Spezifikation	Unterstützte Zielschlüsseltypen	Padding-Schema	AWS CloudHSM Verfügbarkeit des Clients
AES-Verschlüsselung mit Zero Padding	RFC 5649 und SP 800 - 38F	Alle	Fügt zur Blockausrichtung Schlüssel-Bits nach Bedarf Nullen hinzu	SDK 3.1 und höher
AES-Verschlüsselung ohne Padding	RFC 3394 und SP 800 - 38F	Blockausgerichtete Schlüssel wie AES und 3DES	None	SDK 3.1 und höher
AES-Verschlüsselung mit PKCS #5 Padding	None	Alle	Mindestens 8 Bytes werden gemäß PKCS #5 Padding-Schema zur Blockausrichtung hinzugefügt	Alle

Weitere Informationen zur Verwendung der AES-Verschlüsselungsalgorithmen aus der vorangegangenen Tabelle in Ihrer Anwendung finden Sie unter Verwenden von AES Key Wrap in AWS CloudHSM.

Grundlegendes zu Initialisierungsvektoren in der AES-Verschlüsselung

Vor dem Verschlüsseln hängt CloudHSM einen Initialisierungsvektor (IV) an den Zielschlüssel für die Datenintegrität an. Jeder Verschlüsselungsalgorithmus weist spezifische Einschränkungen auf, welcher IV-Typ zulässig ist. Um die IV einzustellen AWS CloudHSM, haben Sie zwei Möglichkeiten:

- Implizit: Setzen Sie den IV auf NULL und CloudHSM verwendet für diesen Algorithmus den Standardwert für Ver- und Entschlüsselungsvorgänge (empfohlen)
- Explizit: Legen Sie den IV fest, indem Sie den Standard-IV-Wert an die Verschlüsselungsfunktion übergeben.

> **Important**
>
> Sie müssen wissen, welchen IV Sie in Ihrer Anwendung verwenden. Um den Schlüssel zu entschlüsseln, müssen Sie denselben IV angeben, den Sie zum Verschlüsseln des Schlüssels verwendet haben. Wenn Sie einen impliziten IV zum Verschlüsseln verwenden, wenden Sie zum Entschlüsseln einen impliziten IV an. Bei einem impliziten IV verwendet CloudHSM zum Entschlüsseln den Standardwert.

Die folgende Tabelle beschreibt zulässige Werte für IVs, die der Verschlüsselungsalgorithmus spezifiziert.

AES-Verschlüsselungsalgorithmus	Implizite IV	Expliziter IV
AES-Verschlüsselung mit Zero Padding	Erforderlich Standardwert: (intern berechneter IV basierend auf der Spezifikation)	Nicht zulässig
AES-Verschlüsselung ohne Padding	Zulässig (empfohlen) Standardwert: `0xA6A6A6A6A6A6A6A6`	Zulässig Nur dieser Wert akzeptiert: `0xA6A6A6A6A6A6A6A6`
AES-Verschlüsselung mit PKCS #5 Padding	Zulässig (empfohlen) Standardwert: `0xA6A6A6A6A6A6A6A6`	Zulässig Nur dieser Wert akzeptiert: `0xA6A6A6A6A6A6A6A6`

Verwenden Sie den AES-Schlüsselumbruch AWS CloudHSM

Sie packen und entpacken Schlüssel wie folgt:

- Wählen Sie wie in der folgenden Tabelle dargestellt in der PCS #11-Bibliothek den entsprechenden Mechanismus für die Funktionen `C_WrapKey` und `C_UnWrapKey` aus.
- Wählen Sie wie in der folgenden Tabelle dargestellt im JCE-Anbieter den entsprechenden Algorithmus, den Modus und die Padding-Kombination zur Implementierung der Cipher-Methoden `Cipher.WRAP_MODE` und `Cipher.UNWRAP_MODE` aus.
- Wählen Sie in der CloudHSM-CLI den entsprechenden Algorithmus aus der Liste der unterstützten Schlüssel entpacken Algorithmen Schlüsselumbruch und Algorithmen aus, wie in der folgenden Tabelle dargestellt.
- Verwenden Sie wie in der folgenden Tabelle dargestellt in key_mgmt_util (KMU) die Befehle wrapKey und unWrapKey mit entsprechenden m-Werten.

AES-Verschlüsselungsalgorithmus	PKCS #11-Mechanismus	Java-Methode	CLI-Unterbefehl	Argument des Key Management Utility (KMU)
AES-Verschlüsselung mit Zero Padding	• `CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD` (Anbieter definierter Mechanismus)	`AESWrap/ECB/ZeroPadding`	aes-zero-pad	m = 6
AES-Verschlüsselung ohne Padding	• `CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD` (Anbieter definierter Mechanismus)	`AESWrap/ECB/NoPadding`	aes-no-pad	m = 5

AES-Verschlüsselungsalgorithmus	PKCS #11-Mechanismus	Java-Methode	CLI-Unterbefehl	Argument des Key Management Utility (KMU)
AES-Verschlüsselung mit PKCS #5 Padding	• `CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD` (Anbieter definierter Mechanismus)	`AESWrap/ECB/PKCS5Padding`	aes-pkcs5-pad	m = 4

Verwenden von vertrauenswürdigen Schlüsseln in AWS CloudHSM

AWS CloudHSM unterstützt Trusted Key Wrapping, um Datenschlüssel vor Insider-Bedrohungen zu schützen. In diesem Thema wird beschrieben, wie Vertrauenswürdige Schlüssel zum Schutz von Daten erstellt werden.

Themen

Grundlegendes zu vertrauenswürdigen Schlüsseln

Ein vertrauenswürdiger Schlüssel ist ein Schlüssel, der zum Umschließen anderer Schlüssel verwendet wird und den Administratoren und Cryptographic Officers (COs) anhand dieses Attributs `CKA_TRUSTED` ausdrücklich als vertrauenswürdig identifizieren. Darüber hinaus geben Administratoren und Cryptographic Officers (COs) anhand von `CKA_UNWRAP_TEMPLATE` und verwandter Attribute an, welche Aktionen Datenschlüssel ausführen können, wenn sie durch einen vertrauenswürdigen Schlüssel entpackt wurden. Datenschlüssel, die durch den vertrauenswürdigen Schlüssel entpackt wurden, müssen auch diese Attribute enthalten, damit der Entpackvorgang erfolgreich ist. Dadurch wird sichergestellt, dass entpackte Datenschlüssel nur für die von Ihnen beabsichtigte Verwendung zulässig sind.

Verwenden Sie das Attribut CKA_WRAP_WITH_TRUSTED, um alle Datenschlüssel zu identifizieren, die Sie mit vertrauenswürdigen Schlüsseln umschließen möchten. Auf diese Weise können Sie Datenschlüssel einschränken, sodass Anwendungen nur vertrauenswürdige Schlüssel verwenden können, um sie zu entpacken. Sobald Sie dieses Attribut für die Datenschlüssel festgelegt haben, ist das Attribut schreibgeschützt und kann nicht mehr geändert werden. Wenn diese Attribute vorhanden sind, können Anwendungen Ihre Datenschlüssel nur mit den Schlüsseln entpacken, denen Sie vertrauen, und Entpackungen führen immer zu Datenschlüsseln mit Attributen, die einschränken, wie diese Schlüssel verwendet werden können.

Vertrauenswürdige Schlüsselattribute

Mit den folgenden Attributen können Sie einen Schlüssel als vertrauenswürdig markieren, angeben, dass ein Datenschlüssel nur mit einem vertrauenswürdigen Schlüssel ein- und ausgepackt werden kann, und steuern, was ein Datenschlüssel nach dem Entpacken tun kann:

- CKA_TRUSTED: Wenden Sie dieses Attribut (zusätzlich zu CKA_UNWRAP_TEMPLATE) auf den Schlüssel an, der Datenschlüssel umhüllt, um anzugeben, dass ein Administrator oder Crypto Officer (CO) die erforderliche Sorgfalt walten lässt und diesem Schlüssel vertraut. Nur ein Administrator oder CO kann CKA_TRUSTED einstellen. Der Crypto-Benutzer (CU) besitzt den Schlüssel, aber nur ein CO kann sein CKA_TRUSTED-Attribut festlegen.
- CKA_WRAP_WITH_TRUSTED: Wenden Sie dieses Attribut auf einen exportierbaren Datenschlüssel an, um anzugeben, dass Sie diesen Schlüssel nur mit Schlüsseln umschließen können, die als CKA_TRUSTED markiert sind. Sobald Sie CKA_WRAP_WITH_TRUSTED auf wahr setzen, wird das Attribut schreibgeschützt und Sie können das Attribut nicht mehr ändern oder entfernen.
- CKA_UNWRAP_TEMPLATE: Wenden Sie dieses Attribut (zusätzlich zu CKA_TRUSTED) auf den Wrapping-Schlüssel an, um anzugeben, welche Attributnamen und -werte der Service automatisch auf Datenschlüssel anwenden muss, die der Dienst entpackt. Wenn eine Anwendung einen Schlüssel zum Entpacken einreicht, kann sie auch ihre eigene Entpackungsvorlage bereitstellen. Wenn Sie eine Unwrap-Vorlage angeben und die Anwendung eine eigene Unwrap-Vorlage bereitstellt, verwendet das HSM beide Vorlagen, um Attributnamen und -werte auf den Schlüssel anzuwenden. Wenn jedoch ein Wert in der CKA_UNWRAP_TEMPLATE für den Wrapping-Schlüssel mit einem Attribut kollidiert, das von der Anwendung während der Unwrap-Anfrage bereitgestellt wird, schlägt die Unwrap-Anfrage fehl.

Weitere Informationen über Attribute finden Sie in den folgenden Themen:

- PKCS #11-Schlüsselattribute

- JCE-Schlüsselattribute
- CloudHSM-CLI-Schlüsselattribute

Wie verwendet man vertrauenswürdige Schlüssel, um Datenschlüssel zu umschließen

Um einen vertrauenswürdigen Schlüssel zum Umschließen eines Datenschlüssels zu verwenden, müssen Sie drei grundlegende Schritte ausführen:

1. Für den Datenschlüssel, den Sie mit einem vertrauenswürdigen Schlüssel umschließen möchten, setzen Sie dessen `CKA_WRAP_WITH_TRUSTED`-Attribut auf wahr.
2. Setzen Sie für den vertrauenswürdigen Schlüssel, mit dem Sie den Datenschlüssel umschließen möchten, dessen `CKA_TRUSTED`-Attribut auf wahr.
3. Verwenden Sie den vertrauenswürdigen Schlüssel, um den Datenschlüssel zu umschließen.

Schritt 1: Festlegen der Datenschlüssel **`CKA_WRAP_WITH_TRUSTED`** auf wahr

Wählen Sie für den Datenschlüssel, den Sie umschließen möchten, eine der folgenden Optionen, um das `CKA_WRAP_WITH_TRUSTED`-Schlüsselattribut auf wahr zu setzen. Dadurch wird der Datenschlüssel eingeschränkt, so dass Anwendungen nur vertrauenswürdige Schlüssel verwenden können, um ihn zu verschlüsseln.

Option 1: Wenn Sie einen neuen Schlüssel generieren, setzen Sie **`CKA_WRAP_WITH_TRUSTED`** auf wahr

Generieren Sie einen Schlüssel mit PKCS #11, JCE oder CloudHSM-CLI. Weitere Einzelheiten finden Sie in den folgenden Beispielen.

PKCS #11

Um einen Schlüssel mit PKCS #11 zu generieren, müssen Sie das `CKA_WRAP_WITH_TRUSTED`-Attribut des Schlüssels auf wahr setzen. Wie im folgenden Beispiel gezeigt, tun Sie dies, indem Sie dieses Attribut in die `CK_ATTRIBUTE template` des Schlüssels aufnehmen und das Attribut dann auf wahr setzen:

```
CK_BYTE_PTR label = "test_key";
CK_ATTRIBUTE template[] = {
        {CKA_WRAP_WITH_TRUSTED, &true_val,         sizeof(CK_BBOOL)},
        {CKA_LABEL,             label,             strlen(label)},
```

```
        ...
};
```

Weitere Informationen finden Sie in unseren öffentlichen Beispielen, die die Schlüsselgenerierung mit PKCS #11 demonstrieren.

JCE

Um einen Schlüssel mit JCE zu generieren, müssen Sie das `WRAP_WITH_TRUSTED`-Attribut des Schlüssels auf wahr setzen. Wie im folgenden Beispiel gezeigt, tun Sie dies, indem Sie dieses Attribut in die `KeyAttributesMap` des Schlüssels aufnehmen und das Attribut dann auf wahr setzen:

```
final String label = "test_key";
final KeyAttributesMap keySpec = new KeyAttributesMap();
keySpec.put(KeyAttribute.WRAP_WITH_TRUSTED, true);
keySpec.put(KeyAttribute.LABEL, label);
...
```

Weitere Informationen finden Sie in unseren öffentlichen Beispielen, die die Schlüsselgenerierung mit JCE demonstrieren.

CloudHSM CLI

Um einen Schlüssel mit CloudHSM-CLI zu generieren, müssen Sie das `wrap-with-trusted`-Attribut des Schlüssels auf wahr setzen. Fügen Sie dazu `wrap-with-trusted=true` in das entsprechende Argument für den Befehl zur Schlüsselgenerierung ein:

- Bei symmetrischen Schlüsseln fügen Sie `wrap-with-trusted` zum `attributes`-Argument hinzu.
- Bei öffentlichen Schlüsseln fügen Sie `wrap-with-trusted` zum `public-attributes`-Argument hinzu.
- Bei privaten Schlüsseln fügen Sie `wrap-with-trusted` zum `private-attributes`-Argument hinzu.

Weitere Informationen zur Generierung von Schlüsselpaaren finden Sie unter key generate-asymmetric-pair .

Weitere Informationen zur Generierung von symmetrischen Schlüsseln finden Sie unter key generate-symmetric .

Option 2: Wenn Sie einen vorhandenen Schlüssel verwenden, verwenden Sie die CloudHSM-CLI, um **`CKA_WRAP_WITH_TRUSTED`** auf wahr zu setzen

Gehen Sie wie folgt vor, um das `CKA_WRAP_WITH_TRUSTED`-Attribut eines vorhandenen Schlüssels auf wahr zu setzen:

1. Verwenden Sie den login-Befehl, um sich als Crypto-Benutzer (CU) anzumelden.
2. Verwenden Sie den key set-attribute-Befehl, um das `wrap-with-trusted`-Schlüsselattribut auf wahr zu setzen.

```
aws-cloudhsm >key set-attribute --filter attr.label=test_key --name wrap-with-
trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

Schritt 2: Festlegen von **`CKA_TRUSTED`** des vertrauenswürdigen Schlüssels auf wahr

Um einen Schlüssel zu einem vertrauenswürdigen Schlüssel zu machen, muss sein `CKA_TRUSTED`-Attribut auf wahr gesetzt werden. Sie können dazu entweder CloudHSM-CLI oder das CloudHSM Management Utility (CMU) verwenden.

- Wenn Sie die CloudHSM-CLI verwenden, um das `CKA_TRUSTED`-Attribut eines Schlüssels festzulegen, finden Sie weitere Informationen unter So markieren Sie einen Schlüssel mit CloudHSM-CLI als vertrauenswürdig.
- Wenn Sie die CMU verwenden, um das `CKA_TRUSTED`-Attribut eines Schlüssels festzulegen, finden Sie weitere Informationen unter So kennzeichnen Sie einen Schlüssel mit der CMU als vertrauenswürdig.

Schritt 3. Verwenden Sie den vertrauenswürdigen Schlüssel, um den Datenschlüssel zu umschließen

Codebeispiele finden Sie unter den folgenden Links, um den in Schritt 1 referenzierten Datenschlüssel mit dem vertrauenswürdigen Schlüssel zu verbinden, den Sie in Schritt 2 festgelegt haben. In jedem Beispiel wird gezeigt, wie Schlüssel umgebrochen werden.

- AWS CloudHSM-PKCS-#11-Beispiele

- AWS CloudHSM-JCE-Beispiele

Wie entpacke ich einen Datenschlüssel mit einem vertrauenswürdigen Schlüssel

Um einen Datenschlüssel zu entpacken, benötigen Sie einen vertrauenswürdigen Schlüssel, der `CKA_UNWRAP` auf wahr gesetzt ist. Um ein solcher Schlüssel zu sein, muss er auch folgende Kriterien erfüllen:

- Das `CKA_TRUSTED`-Attribut des Schlüssels muss auf wahr festgelegt werden.
- Der Schlüssel muss anhand von `CKA_UNWRAP_TEMPLATE` und verwandter Attribute angeben, welche Aktionen Datenschlüssel ausführen können, wenn sie entpackt wurden. Wenn Sie beispielsweise möchten, dass ein entpackter Schlüssel nicht exportierbar ist, legen Sie `CKA_EXPORTABLE = FALSE` als Teil von `CKA_UNWRAP_TEMPLATE` fest.

> **Note**
>
> `CKA_UNWRAP_TEMPLATE` ist nur mit PKCS #11 verfügbar.

Wenn eine Anwendung einen Schlüssel zum Entpacken einreicht, kann die Anwendung auch ihre eigene Unwrap-Vorlage bereitstellen. Wenn Sie eine Unwrap-Vorlage angeben und die Anwendung eine eigene Unwrap-Vorlage bereitstellt, verwendet das HSM beide Vorlagen, um Attributnamen und -werte auf den Schlüssel anzuwenden. Wenn jedoch während der Entpackungsanforderung ein Wert in `CKA_UNWRAP_TEMPLATE` des vertrauenswürdigen Schlüssels mit einem von der Anwendung bereitgestellten Attribut in Konflikt gerät, schlägt die Entpackungsanforderung fehl.

Ein Beispiel für das Entpacken eines Datenschlüssels mit einem vertrauenswürdigen Schlüssel finden Sie in diesem PKCS-#11-Beispiel.

Schlüssel mit CloudHSM-CLI verwalten

Wenn Sie die neueste SDK-Versionsserie verwenden, verwenden Sie die CloudHSM-CLI, um die Schlüssel in Ihrem AWS CloudHSM-Cluster zu verwalten. Weitere Informationen finden Sie in den folgenden Themen.

- Unter Verwendung vertrauenswürdiger Schlüssel wird beschrieben, wie Sie mithilfe von PKCS #11-Bibliotheksattributen und der CloudHSM-CLI vertrauenswürdige Schlüssel zur Sicherung von Daten erstellen.

- Das Generieren von Schlüsseln umfasst Anweisungen zum Erstellen von Schlüsseln, einschließlich symmetrischer Schlüssel, RSA-Schlüssel und EC-Schlüssel.
- Unter Schlüssel löschen wird beschrieben, wie Schlüsselbesitzer Schlüssel löschen.
- Unter Teilen und Aufheben der Freigabe von Schlüsseln wird beschrieben, wie Schlüsselinhaber Schlüssel teilen und deren Weitergabe rückgängig machen.
- Das Filtern von Schlüsseln enthält Richtlinien zur Verwendung von Filtern zum Auffinden von Schlüsseln.

Verwenden der CloudHSM-CLI zum Generieren von Schlüsseln

Bevor Sie einen Schlüssel generieren können, müssen Sie CloudHSM-CLI starten und sich als Crypto-Benutzer (CU) anmelden. Zum Generieren der Schlüssel auf dem HSM verwenden Sie den entsprechenden Befehl für die Art von Schlüssel, die Sie erstellen möchten.

Themen

- Generieren von symmetrischen Schlüsseln
- Generieren von asymmetrischen Schlüsseln
- Verwandte Themen

Generieren von symmetrischen Schlüsseln

Verwenden Sie die unter key generate-symmetric aufgeführten Befehle, um symmetrische Schlüssel zu generieren. Verwenden Sie den Befehl help key generate-symmetric, um alle verfügbaren Optionen anzuzeigen.

Generieren eines AES-Schlüssels

Verwenden Sie den key generate-symmetric aes-Befehl, um AES-Schlüssel zu generieren. Verwenden Sie den Befehl help key generate-symmetric aes, um alle verfügbaren Optionen anzuzeigen.

Example

Das folgende Beispiel generiert einen 32-Byte-AES-Schlüssel.

```
aws-cloudhsm > key generate-symmetric aes \
    --label aes-example \
```

```
    --key-length-bytes 32
```

Argumente

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den AES-Schlüssel an.

Erforderlich: Ja

<KEY-LENGTH-BYTES>

Gibt die Schlüssellänge in Byte an.

Zulässige Werte:

- 16, 24 und 32

Erforderlich: Ja

<KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten AES-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`)

Eine Liste der unterstützten AWS CloudHSM-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden. Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie key set-attribute.

Wenn Schlüssel generiert werden, handelt es sich standardmäßig um persistente/Token-Schlüssel. Die Verwendung von <SESSION> ändert dies und stellt sicher, dass ein mit diesem Argument erzeugter Schlüssel eine Sitzung/ephemisch ist

Erforderlich: Nein

Generieren Sie einen generischen geheimen Schlüssel

Verwenden Sie den key generate-symmetric generic-secret-Befehl, um generische geheime Schlüssel zu generieren. Verwenden Sie den Befehl help key generate-symmetric generic-secret, um alle verfügbaren Optionen anzuzeigen.

Example

Das folgende Beispiel generiert einen generischen 32-Byte-Schlüssel.

```
aws-cloudhsm > key generate-symmetric generic-secret \
    --label generic-secret-example \
    --key-length-bytes 32
```

Argumente

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den generischen geheimen Schlüssel an.

Erforderlich: Ja

<KEY-LENGTH-BYTES>

Gibt die Schlüssellänge in Byte an.

Zulässige Werte:

- 1 bis 800

Erforderlich: Ja

<KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten generischen geheimen Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`)

Eine Liste der unterstützten AWS CloudHSM-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden. Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie key set-attribute.

Wenn Schlüssel generiert werden, handelt es sich standardmäßig um persistente/Token-Schlüssel. Die Verwendung von <SESSION> ändert dies und stellt sicher, dass ein mit diesem Argument erzeugter Schlüssel eine Sitzung/ephemisch ist

Erforderlich: Nein

Generieren von asymmetrischen Schlüsseln

Verwenden Sie die in key generate-asymmetric-pair aufgeführten Befehle, um asymmetrische Schlüsselpaare zu erzeugen.

Generieren Sie einen RSA-Schlüssel

Verwenden Sie den key generate-asymmetric-pair rsa-Befehl, um ein RSA-Schlüsselpaar zu erzeugen. Verwenden Sie den Befehl help key generate-asymmetric-pair rsa, um alle verfügbaren Optionen anzuzeigen.

Example

Das folgende Beispiel generiert ein 2048-Bit-RSA-Schlüsselpaar.

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
    --public-exponent 65537 \
    --modulus-size-bits 2048 \
    --public-label rsa-public-example \
    --private-label rsa-private-example
```

Argumente

<PUBLIC_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an.

Erforderlich: Ja

<PRIVATE_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den private-key an.

Erforderlich: Ja

<MODULUS_SIZE_BITS>

Gibt die Länge des Moduls in Bits an. Der minimale Wert beträgt 2048.

Erforderlich: Ja

<PUBLIC_EXPONENT>

Gibt den öffentlichen Exponenten an. Bei diesem Wert muss es sich eine ungerade Zahl gleich oder größer als 65537 handeln.

Erforderlich: Ja

<PUBLIC_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten öffentlichen RSA-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`).

Eine Liste der unterstützten AWS CloudHSM-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden. Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie key set-attribute.

Wenn Schlüssel generiert werden, handelt es sich standardmäßig um persistente/Token-Schlüssel. Die Verwendung von <SESSION> ändert dies und stellt sicher, dass ein mit diesem Argument erzeugter Schlüssel eine Sitzung/ephemisch ist

Erforderlich: Nein

Generieren von ECC- (Elliptic Curve Cryptography-)Schlüsselpaaren

Verwenden Sie den key generate-asymmetric-pair ec-Befehl, um ein EC-Schlüsselpaar zu generieren. Wenn Sie alle verfügbaren Optionen, einschließlich einer Liste der unterstützten Ellipsenkurven anzeigen möchten, verwenden Sie den Befehl help key generate-asymmetric-pair ec.

Example

Das folgende Beispiel erzeugt ein EC-Schlüsselpaar unter Verwendung der elliptischen Kurve Secp384r1.

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp384r1 \
    --public-label ec-public-example \
    --private-label ec-private-example
```

Argumente

<PUBLIC_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an. Die maximal zulässige Größe für `label` beträgt 127 Zeichen für Client-SDK 5.11 und höher. Das Client-SDK 5.10 und früher hat ein Limit von 126 Zeichen.

Erforderlich: Ja

<PRIVATE_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den private-key an. Die maximal zulässige Größe für `label` beträgt 127 Zeichen für Client-SDK 5.11 und höher. Das Client-SDK 5.10 und früher hat ein Limit von 126 Zeichen.

Erforderlich: Ja

<CURVE>

Gibt die ID für die elliptische Kurve an.

Zulässige Werte:

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

Erforderlich: Ja

<PUBLIC_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten öffentlichen EC-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`).

Eine Liste der unterstützten AWS CloudHSM-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<PRIVATE_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten privaten EC-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`).

Eine Liste der unterstützten AWS CloudHSM-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden. Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel

verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie key set-attribute.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von <SESSION> ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI
- key generate-asymmetric-pair
- key generate-symmetric

Verwenden der CloudHSM-CLI zum Löschen von Schlüsseln

Verwenden Sie das Beispiel in diesem Thema, um einen Schlüssel mit der CloudHSM-CLI zu löschen. Nur Schlüsselinhaber können Schlüssel löschen.

Themen

- Beispiel: Löschen Sie einen Schlüssel
- Verwandte Themen

Beispiel: Löschen Sie einen Schlüssel

1. Führen Sie den key list-Befehl aus, um den Schlüssel zu identifizieren, den Sie löschen möchten

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000540011",
        "key-info": {
```

```
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_key_to_delete",
          "id": "",
          "check-value": "0x29bbd1",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
          "modulus":
"0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
```

```
  }
```

2. Nachdem Sie den Schlüssel identifiziert haben, führen Sie key delete mit dem eindeutigen `label`-Schlüsselattribut aus, um den Schlüssel zu löschen:

```
aws-cloudhsm > key delete --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. Führen Sie den key list-Befehl mit dem eindeutigen `label`-Schlüsselattribut aus und vergewissern Sie sich, dass der Schlüssel gelöscht wurde. Wie im folgenden Beispiel gezeigt, befindet sich kein Schlüssel mit dem Label `my_key_to_delete` im HSM-Cluster:

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "matched_keys": [],
    "total_key_count": 0,
    "returned_key_count": 0
  }
}
```

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI
- key delete

Verwenden der CloudHSM-CLI zum Teilen und Aufheben der Freigabe von Schlüsseln

Verwenden Sie die Befehle in diesem Thema, um Schlüssel in der CloudHSM-CLI zu teilen und deren Freigabe aufzuheben. In ist AWS CloudHSMder Crypto-Benutzer (CU), der den Schlüssel erstellt, Eigentümer des Schlüssels. Der Besitzer kann die Befehle key share und key unshare verwenden, um den Schlüssel mit anderen CUs zu teilen oder die gemeinsame Nutzung aufzuheben. Benutzer, mit denen der Schlüssel gemeinsam genutzt wird, können den Schlüssel

in kryptographischen Operationen verwenden. Sie können jedoch den Schlüssel nicht exportieren, löschen oder mit anderen Benutzern teilen.

Bevor Sie einen Schlüssel freigeben können, müssen Sie sich beim HSM als der Crypto-Benutzer (CU) anmelden, dem der Schlüssel gehört.

Themen

- Beispiel: Freigabe und Aufhebung der Freigabe eines Schlüssels
- Verwandte Themen

Beispiel: Freigabe und Aufhebung der Freigabe eines Schlüssels

Example

Das folgende Beispiel zeigt, wie Sie einen Schlüssel für den Crypto-Benutzer (CU) `alice` freigeben und die Freigabe aufheben. Neben den Befehlen key share und key unshare erfordern die Befehle zum Teilen und Aufheben der Freigabe auch einen bestimmten Schlüssel mithilfe von CloudHSM-CLI-Schlüsselfiltern und den spezifischen Benutzernamen des Benutzers, mit dem der Schlüssel geteilt oder nicht geteilt werden soll.

1. Führen Sie zunächst den key list-Befehl mit einem Filter aus, um einen bestimmten Schlüssel zurückzugeben und zu sehen, mit wem der Schlüssel bereits geteilt wurde.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
```

```
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
```

```
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

2. Sehen Sie sich die `shared-users`-Ausgabe an, um festzustellen, mit wem der Schlüssel derzeit geteilt wird.
3. Geben Sie den folgenden Befehl ein, um diesen Schlüssel mit dem Crypto-Benutzer (CU) `alice` zu teilen:

```
aws-cloudhsm > key share --filter attr.label="rsa_key_to_share" attr.class=private-
key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key shared successfully"
  }
}
```

   Beachten Sie, dass dieser Befehl zusammen mit dem key share-Befehl die eindeutige Bezeichnung des Schlüssels und den Namen des Benutzers verwendet, mit dem der Schlüssel geteilt wird.
4. Führen Sie den key list-Befehl aus, um zu bestätigen, dass der Schlüssel mit `alice` geteilt wurde:

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
```

```
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
```

```
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

5. Führen Sie den folgenden unshare-Befehl aus, um die gemeinsame Nutzung desselben Schlüssels mit `alice` rückgängig zu machen:

```
aws-cloudhsm > key unshare --filter attr.label="rsa_key_to_share"
 attr.class=private-key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
```

```
    "message": "Key unshared successfully"
  }
}
```

Beachten Sie, dass dieser Befehl zusammen mit dem key unshare-Befehl die eindeutige Bezeichnung des Schlüssels und den Namen des Benutzers verwendet, mit dem der Schlüssel geteilt wird.

6. Führen Sie den key list-Befehl erneut aus und vergewissern Sie sich, dass der Schlüssel nicht an den Crypto-Benutzer `alice` weitergegeben wurde:

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
```

```
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
```

```
  }
}
```

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI
- key share
- key unshare
- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln

Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln

Verwenden Sie die folgenden Tastenbefehle, um die standardisierten Schlüsselfiltermechanismen für CloudHSM-CLI zu nutzen.

- key list
- key delete
- key share
- key unshare
- key set-attribute

Um Schlüssel mit der CloudHSM-CLI auszuwählen und/oder zu filtern, verwenden Tastenbefehle einen standardisierten Filtermechanismus, der auf Schlüsselattribute für CloudHSM-CLI basiert. Ein Schlüssel oder eine Gruppe von Schlüsseln kann in Tastenbefehlen angegeben werden, indem ein oder mehrere AWS CloudHSM-Attribute verwendet werden, die einen einzelnen Schlüssel oder mehrere Schlüssel identifizieren können. Der Schlüsselfiltermechanismus funktioniert nur für Schlüssel, die der aktuell angemeldete Benutzer besitzt und gemeinsam nutzt, sowie für alle öffentlichen Schlüssel im AWS CloudHSM-Cluster.

Themen

- Voraussetzungen
- Filtern, um einen einzelnen Schlüssel zu finden
- Fehler bei der Filterung
- Verwandte Themen

Voraussetzungen

Um Schlüssel zu filtern, müssen Sie als Crypto-Benutzer (CU) angemeldet sein.

Filtern, um einen einzelnen Schlüssel zu finden

Bitte beachten Sie, dass in den folgenden Beispielen jedes Attribut, das als Filter verwendet wird, in der Form von `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` geschrieben werden muss. Wenn Sie beispielsweise nach dem Label-Attribut filtern möchten, schreiben Sie `attr.label=my_label`.

Example Verwenden Sie ein einzelnes Attribut, um einen einzelnen Schlüssel zu finden

Dieses Beispiel zeigt, wie mit nur einem einzigen identifizierenden Attribut nach einem einzigen eindeutigen Schlüssel gefiltert wird.

```
aws-cloudhsm > key list --filter attr.label="my_unique_key_label" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_unique_key_label",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
```

```
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254c8f5
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```


Example Verwenden Sie mehrere Attribute, um einen einzelnen Schlüssel zu finden

Das folgende Beispiel zeigt, wie Sie einen einzelnen Schlüssel mithilfe mehrerer Schlüsselattribute finden.

```
aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key attr.check-
value=0x29bbd1 --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
```

```
        "key-reference": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_crypto_user",
          "id": "",
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
          "modulus":
"0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990c2a7
```

```
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

Example Filtern, um einen Satz von Schlüsseln zu finden

Das folgende Beispiel zeigt, wie Sie nach einer Reihe von privaten RSA-Schlüsseln filtern.

```
aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
```

```
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254c8f5
          "modulus-size-bits": 2048
        }
      },
      {
        "key-reference": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
```

```
          "key-type": "rsa",
          "label": "my_test_key",
          "id": "",
          "check-value": "0x29bbd1",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
          "modulus":
 "0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990c2a7
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 2,
    "returned_key_count": 2
  }
}
```


Fehler bei der Filterung

Bestimmte Schlüsseloperationen können jeweils nur mit einem einzigen Schlüssel ausgeführt werden. Bei diesen Vorgängen gibt die CloudHSM-CLI einen Fehler aus, wenn die Filterkriterien nicht ausreichend verfeinert sind und mehrere Schlüssel den Kriterien entsprechen. Ein solches Beispiel ist unten mit dem Schlüssel delete dargestellt.

Example Filtrationsfehler beim Abgleichen von zu vielen Schlüsseln

```
aws-cloudhsm > key delete --filter attr.key-type=rsa
{
  "error_code": 1,
  "data": "Key selection criteria matched 48 keys. Refine selection criteria to select
 a single key."
}
```

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI

So markieren Sie einen Schlüssel mit CloudHSM-CLI als vertrauenswürdig

Der Inhalt dieses Abschnitts enthält Anweisungen zur Verwendung der CloudHSM-CLI, um einen Schlüssel als vertrauenswürdig zu kennzeichnen.

1. Melden Sie sich mit dem loginCloudHSM-CLI-Befehl als Crypto-Benutzer (CU) an.
2. Verwenden Sie den key list-Befehl, um die Schlüsselreferenz des Schlüssels zu identifizieren, den Sie als vertrauenswürdig markieren möchten. Das folgende Beispiel listet den Schlüssel mit dem Label `key_to_be_trusted` auf.

   ```
   aws-cloudhsm > key list --filter attr.label=test_aes_trusted
           {
     "error_code": 0,
     "data": {
       "matched_keys": [
         {
           "key-reference": "0x0000000000200333",
           "attributes": {
             "label": "test_aes_trusted"
           }
         }
       ],
       "total_key_count": 1,
       "returned_key_count": 1
     }
   }
   ```

3. Melden Sie sich mit dem logout-Befehl als Crypto-Benutzer (CU) ab.

4. Melden Sie sich mit dem Befehl als login-Administrator an.
5. Verwenden Sie den Befehl key set-attribute mit der Schlüsselreferenz, die Sie in Schritt 2 identifiziert haben, und setzen Sie den vertrauenswürdigen Wert des Schlüssels auf wahr:

```
aws-cloudhsm > key set-attribute --filter key-reference=<Key Reference> --name
 trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

Schlüssel mit der KMU und CMU verwalten

Wenn Sie die neueste SDK-Versionsserie verwenden, verwenden Sie die CloudHSM-CLI, um die Schlüssel in Ihrem AWS CloudHSM-Cluster zu verwalten.

Wenn Sie die vorherige SDK-Versionsserie verwenden, können Sie die Schlüssel auf den HSMs in Ihrem AWS CloudHSM-Cluster mit dem Befehlszeilentool key_mgmt_util verwalten. Bevor Sie die Schlüssel verwalten können, müssen Sie den AWS CloudHSM-Client starten, key_mgmt_util starten, und sich an den HSMs anmelden. Weitere Informationen finden Sie unter Erste Schritte mit key_mgmt_util.

- Unter Verwendung vertrauenswürdiger Schlüssel wird beschrieben, wie Sie mithilfe von PKCS #11-Bibliotheksattributen und der CMU vertrauenswürdige Schlüssel zur Sicherung von Daten erstellen.
- Das Generieren von Schlüsseln umfasst Anweisungen zum Generieren von Schlüsseln, einschließlich symmetrischer Schlüssel, RSA-Schlüssel und EC-Schlüssel.
- Beim Importieren von Schlüsseln wird detailliert beschrieben, wie Schlüsselbesitzer Schlüssel importieren.
- Exportieren von Schlüsseln enthält Einzelheiten darüber, wie Schlüsselbesitzer Schlüssel exportieren.
- Löschen von Schlüsseln enthält Einzelheiten darüber, wie Schlüsselbesitzer Schlüssel löschen.
- Unter Teilen und Aufheben der Freigabe von Schlüsseln wird beschrieben, wie Schlüsselinhaber Schlüssel teilen und deren Weitergabe rückgängig machen.

Generieren von Schlüsseln

Zum Generieren der Schlüssel auf dem HSM verwenden Sie den entsprechenden Befehl für die Art von Schlüssel, die Sie erstellen möchten.

Themen

- Generieren von symmetrischen Schlüsseln
- Generieren von RSA-Schlüsselpaaren
- Generieren von ECC- (Elliptic Curve Cryptography-)Schlüsselpaaren

Generieren von symmetrischen Schlüsseln

Verwenden Sie den Befehl genSymKey, um AES und andere symmetrische Schlüssel zu generieren. Verwenden Sie den Befehl genSymKey -h, um alle verfügbaren Optionen anzuzeigen.

In dem folgenden Beispiel wird ein 256-Bit-AES-Schlüssel erstellt.

```
Command: genSymKey -t 31 -s 32 -l aes256
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 524295

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Generieren von RSA-Schlüsselpaaren

Um ein RSA-Schlüsselpaar zu generieren, verwenden Sie den Befehl genRSAKeyPair. Verwenden Sie den Befehl genRSAKeyPair -h, um alle verfügbaren Optionen anzuzeigen.

Das folgende Beispiel generiert ein 2048-Bit-RSA-Schlüsselpaar.

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa2048
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 524294    private key handle: 524296
```

```
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Generieren von ECC- (Elliptic Curve Cryptography-)Schlüsselpaaren

Um ein ECC-Schlüsselpaar zu generieren, verwenden Sie den Befehl genECCKeyPair. Wenn Sie alle verfügbaren Optionen, einschließlich einer Liste der unterstützten Ellipsenkurven anzeigen möchten, verwenden Sie den Befehl genECCKeyPair -h.

Das folgende Beispiel erzeugt ein ECC-Schlüsselpaar unter Verwendung der elliptischen Kurve P-384, die in der NIST FIPS Publikation 186-4 definiert ist.

```
Command: genECCKeyPair -i 14 -l ecc-p384
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 524297    private key handle: 524298

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Importieren von Schlüsseln

Um geheime Schlüssel, d. h. symmetrische Schlüssel und asymmetrische private Schlüssel in das HSM zu importieren, müssen Sie zunächst einen Wrapping-Schlüssel im HSM erstellen. Öffentliche Schlüssel können direkt ohne Umhüllungsschlüssel importiert werden.

Themen

- Importieren geheimer Schlüssel
- Importieren von öffentlichen Schlüsseln

Importieren geheimer Schlüssel

Führen Sie die folgenden Schritte aus, um einen geheimen Schlüssel zu importieren. Wenn Sie einen geheimen Schlüssel importieren möchten, speichern Sie ihn zunächst in einer Datei. Speichern Sie symmetrische Schlüssel als Rohbytes und asymmetrische private Schlüssel im PEM-Format.

Dieses Beispiel zeigt, wie man einen geheimen Klartext-Schlüssel aus einer Datei in das HSM importiert. Um einen verschlüsselten Schlüssel aus einer Datei in das HSM zu importieren, verwenden Sie den Befehl unWrapKey.

So importieren Sie einen geheimen Schlüssel

1. Verwenden Sie den Befehl genSymKey, um einen Umhüllungsschlüssel zu erstellen. Mit dem folgenden Befehl wird ein 128-Bit-AES-Schlüssel erstellt, der nur für die aktuelle Sitzung gilt. Sie können einen Sitzungsschlüssel oder einen persistenten Schlüssel als Wrapping-Schlüssel verwenden.

```
Command: genSymKey -t 31 -s 16 -sess -l import-wrapping-key
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 524299

Cluster Error Status
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

2. Verwenden Sie, abhängig vom Typ des geheimen Schlüssels, den Sie importieren, einen der folgenden Befehle.

   - Um einem symmetrischen Schlüssel zu importieren, verwenden Sie den Befehl imSymKey. Mit dem folgenden Befehl wird ein AES-Schlüssel aus einer Datei mit dem Namen `aes256.key` importiert, wobei der im vorangehenden Schritt erstellte Umhüllungsschlüssel benötigt wird. Verwenden Sie den Befehl imSymKey -h, um alle verfügbaren Optionen anzuzeigen.

```
Command: imSymKey -f aes256.key -t 31 -l aes256-imported -w 524299
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Unwrapped.  Key Handle: 524300

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

- Um einen asymmetrischen privaten Schlüssel zu importieren, verwenden Sie den Befehl importPrivateKey. Mit dem folgenden Befehl wird ein privater Schlüssel aus einer Datei mit dem Namen `rsa2048.key` importiert, wobei der im vorangehenden Schritt erstellte Umhüllungsschlüssel benötigt wird. Verwenden Sie den Befehl importPrivateKey -h, um alle verfügbaren Optionen anzuzeigen.

```
Command: importPrivateKey -f rsa2048.key -l rsa2048-imported -w 524299
BER encoded key length is 1216

Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Private Key Unwrapped.  Key Handle: 524301

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Importieren von öffentlichen Schlüsseln

Um einen öffentlichen Schlüssel zu importieren, verwenden Sie den Befehl importPubKey. Verwenden Sie den Befehl importPubKey -h, um alle verfügbaren Optionen anzuzeigen.

Im folgenden Beispiel wird öffentlicher RSA-Schlüssel aus einer Datei mit dem Namen `rsa2048.pub` importiert.

```
Command: importPubKey -f rsa2048.pub -l rsa2048-public-imported
Cfm3CreatePublicKey returned: 0x00 : HSM Return: SUCCESS

Public Key Handle: 524302

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Exportieren von Schlüsseln

Um geheime Schlüssel, d. h. symmetrische Schlüssel und asymmetrische private Schlüsselaus dem HSM zu exportieren, müssen Sie zunächst einen Wrapping-Schlüssel erstellen. Öffentliche Schlüssel können direkt ohne Umhüllungsschlüssel exportiert werden.

Nur der Schlüsselbesitzer kann einen Schlüssel exportieren. Benutzer, mit denen der Schlüssel gemeinsam genutzt wird, können den Schlüssel in kryptographischen Operationen verwenden, aber ihn nicht exportieren. Wenn Sie dieses Beispiel ausführen, stellen Sie sicher, dass Sie einen von Ihnen erstellten Schlüssel exportieren.

> **Important**
>
> Der Befehl exSymKey schreibt eine Klartextkopie (unverschlüsselt) des geheimen Schlüssels in eine Datei. Der Exportvorgang erfordert einen Wrapping-Schlüssel. Der Schlüssel in der Datei jedoch ist kein Wrapping-Schlüssel. Um eine verpackte (verschlüsselte) Kopie eines Schlüssels zu exportieren, verwenden Sie den Befehl wrapKey.

Themen

Exportieren geheimer Schlüssel

Führen Sie die folgenden Schritte aus, um einen geheimen Schlüssel zu exportieren.

So exportieren Sie einen geheimen Schlüssel

1. Verwenden Sie den Befehl genSymKey, um einen Umhüllungsschlüssel zu erstellen. Mit dem folgenden Befehl wird ein 128-Bit-AES-Schlüssel erstellt, der nur für die aktuelle Sitzung gilt.

```
Command: genSymKey -t 31 -s 16 -sess -l export-wrapping-key
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 524304

Cluster Error Status
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

2. Verwenden Sie, abhängig vom Typ des geheimen Schlüssels, den Sie exportieren, einen der folgenden Befehle.

   - Um einem symmetrischen Schlüssel zu exportieren, verwenden Sie den Befehl exSymKey. Der folgende Befehl exportiert einen AES-Schlüssel in eine Datei mit dem Namen `aes256.key.exp`. Verwenden Sie den Befehl exSymKey -h, um alle verfügbaren Optionen anzuzeigen.

     ```
     Command: exSymKey -k 524295 -out aes256.key.exp -w 524304
     Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

     Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


     Wrapped Symmetric Key written to file "aes256.key.exp"
     ```

     > **Note**
     >
     > Die Ausgabe des Befehls besagt, dass ein „Wrapped Symmetric Key“ in die Ausgabedatei geschrieben wurde. Die Ausgabedatei enthält jedoch einen Klartext-Schlüssel (nicht verpackt). Um einen verpackten (verschlüsselten) Schlüssel in eine Datei zu exportieren, verwenden Sie den Befehl wrapKey.

   - Um einem privaten Schlüssel zu exportieren, verwenden Sie den Befehl exportPrivateKey. Der folgende Befehl exportiert einen privaten Schlüssel in eine Datei mit dem Namen `rsa2048.key.exp`. Verwenden Sie den Befehl exportPrivateKey -h, um alle verfügbaren Optionen anzuzeigen.

     ```
     Command: exportPrivateKey -k 524296 -out rsa2048.key.exp -w 524304
     Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

     Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS

     PEM formatted private key is written to rsa2048.key.exp
     ```

Exportieren von öffentlichen Schlüsseln

Um einen öffentlichen Schlüssel zu exportieren, verwenden Sie den Befehl exportPubKey. Verwenden Sie den Befehl exportPubKey -h, um alle verfügbaren Optionen anzuzeigen.

Im folgenden Beispiel wird öffentlicher RSA-Schlüssel in eine Datei mit dem Namen `rsa2048.pub.exp` exportiert.

```
Command: exportPubKey -k 524294 -out rsa2048.pub.exp
PEM formatted public key is written to rsa2048.pub.key

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

Löschen von Schlüsseln

Verwenden Sie zum Löschen eines Schlüssels den Befehl deleteKey wie in dem folgenden Beispiel. Nur der Schlüsselbesitzer kann einen Schlüssel löschen.

```
Command: deleteKey -k 524300
Cfm3DeleteKey returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Schlüssel teilen und das Teilen rückgängig machen

In AWS CloudHSM befindet sich der Schlüssel im Besitz des CU, der diesen erstellt hat. Der Besitzer verwaltet den Schlüssel, kann ihn exportieren und löschen und kann ihn in kryptographischen Operationen verwenden. Der Besitzer kann den Schlüssel außerdem mit anderen CU-Benutzern teilen. Benutzer, mit denen der Schlüssel gemeinsam genutzt wird, können den Schlüssel in kryptographischen Operationen verwenden. Sie können jedoch den Schlüssel nicht exportieren, löschen oder mit anderen Benutzern teilen.

Sie können Schlüssel beim Erstellen mit anderen CU-Benutzern teilen – z. B. durch Verwendung des Parameters `-u` für die Befehle genSymKey oder genRSAKeyPair. Um vorhandene Schlüssel für einen anderen HSM-Benutzer freizugeben, verwenden Sie das Befehlszeilentool cloudhsm_mgmt_util. Dies unterscheidet sich von den meisten der in diesem Abschnitt dokumentierten Aufgaben, die das Befehlszeilentool key_mgmt_util verwenden.

Um einen Schlüssel freigeben zu können, müssen Sie cloudhsm_mgmt_util starten, die End-to-End-Verschlüsselung aktivieren und sich an den HSMs anmelden. Um einen Schlüssel freizugeben, melden Sie sich am HSM als der Crypto-Benutzer (CU, Crypto User) an, dem der Schlüssel gehört. Nur der Eigentümer eines Schlüssels ist berechtigt, einen Schlüssel freizugeben.

Verwenden Sie den Befehl shareKey, um einen Schlüssel freizugeben oder die Freigabe aufzuheben. Dabei geben Sie den Handle des Schlüssels und die IDs des Benutzers oder der Benutzer an. Wenn Sie den Schlüssel für mehr als einen Benutzer freigeben oder die Freigabe aufheben, geben Sie die Benutzer als eine durch Komma getrennte Liste der Benutzer-IDs an. Um einen Schlüssel freizugeben, verwenden Sie den Befehl 1 als letzten Parameter, wie im folgenden Beispiel gezeigt. Um die Freigabe aufzuheben, verwenden Sie `0`.

```
aws-cloudhsm>shareKey 524295 4 1
************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
************************************************************

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.2.9)
shareKey success on server 1(10.0.3.11)
shareKey success on server 2(10.0.1.12)
```

Nachfolgend finden Sie die allgemeine Syntax für den Befehl shareKey.

```
aws-cloudhsm>shareKey <key handle> <user ID> <Boolean: 1 for share, 0 for unshare>
```

So kennzeichnen Sie einen Schlüssel mit der CMU als vertrauenswürdig

Der Inhalt dieses Abschnitts enthält Anweisungen zur Verwendung der CMU, um einen Schlüssel als vertrauenswürdig zu kennzeichnen.

1. Melden Sie sich mit dem Befehl LoginHSM als Crypto Officer (CO) an.
2. Verwenden Sie den setAttribute-Befehl, bei dem `OBJ_ATTR_TRUSTED` (Wert `134`) auf wahr (1) gesetzt ist.

   ```
   setAttribute <Key Handle> 134 1
   ```

Verwalten von geklonten Clustern

Verwenden Sie CloudHSM Management Utility (CMU), um einen Cluster in einer Remote-Region zu synchronisieren, wenn der Cluster in dieser Region ursprünglich aus der Sicherung eines Clusters

in einer anderen Region erstellt wurde. Nehmen wir an, Sie haben einen Cluster in eine andere Region (Ziel) kopiert und möchten später Änderungen aus dem ursprünglichen Cluster (Quelle) synchronisieren. In solchen Szenarien verwenden Sie CMU, um die Cluster zu synchronisieren. Dazu erstellen Sie eine neue CMU-Konfigurationsdatei, geben Hardware-Sicherheitsmodule (HSM) aus beiden Clustern in der neuen Datei an und verwenden dann CMU, um mit dieser Datei eine Verbindung zum Cluster herzustellen.

Um CMU für alle geklonten Cluster zu verwenden

1. Erstellen Sie eine Kopie Ihrer aktuellen Konfigurationsdatei und ändern Sie den Namen der Kopie in einen anderen Namen.

   Verwenden Sie beispielsweise die folgenden Dateispeicherorte, um eine Kopie Ihrer aktuellen Konfigurationsdatei zu suchen und zu erstellen, und ändern Sie dann den Namen der Kopie von `cloudhsm_mgmt_config.cfg` zu `syncConfig.cfg`.

   - Linux: `/opt/cloudhsm/etc/cloudhsm_mgmt_config.cfg`
   - Windows: `C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_config.cfg`

2. Fügen Sie in der umbenannten Kopie die Elastic-Network-Schnittstelle (ENI)-IP des Ziel-HSM hinzu (das HSM in der fremden Region, das synchronisiert werden muss). Wir empfehlen, dass Sie das Ziel-HSM unter dem Quell-HSM hinzufügen.

```
{
    ...
    "servers": [
        {
            ...
            "hostname": "<ENI Source IP>",
            ...
        },
        {
            ...
            "hostname": "<ENI Destination IP>",
            ...
        }
    ]
}
```

Weitere Informationen darüber, wie Sie die IP-Adresse erhalten, finden Sie unter the section called “Holen Sie sich eine IP-Adresse für ein HSM”.

3. Initialisieren Sie CMU mit der neuen Konfigurationsdatei:

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/userSync.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM>cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\userSync.cfg
   ```

4. Überprüfen Sie die zurückgegebenen Statusmeldungen, um sicherzustellen, dass das CMU mit allen gewünschten HSMs verbunden ist. Bestimmen Sie, welche der zurückgegebenen ENI-IPs jedem Cluster entspricht. Verwenden Sie SyncUser und SyncKey, um Benutzer und Schlüssel manuell zu synchronisieren. Weitere Informationen finden Sie unter syncUser und syncKey.

Holen Sie sich eine IP-Adresse für ein HSM

Verwenden Sie diesen Abschnitt, um eine IP-Adresse für ein HSM zu erhalten.

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.
4. Um die IP-Adresse abzurufen, wählen Sie auf der Registerkarte HSMs eine der IP-Adressen aus, die unter ENI-IP-Adresse aufgeführt sind.

Um eine IP-Adresse für ein HSM (CLI) zu erhalten

- Rufen Sie die IP-Adresse eines HSM mit dem describe-clusters Befehl von der CLI ab. In der Ausgabe des Befehls sind die IP-Adressen der HSMs die Werte von `EniIp`.

  ```
  $  aws cloudhsmv2 describe-clusters
  ```

```
{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

Verwandte Themen

- syncUser
- syncKey
- Regionsübergreifendes Kopieren von Sicherungen

AWS CloudHSM Befehlszeilentools

In diesem Thema werden die Befehlszeilentools beschrieben, die für die Verwaltung und Verwendung von AWS CloudHSM verfügbar sind.

Themen

Grundlegendes zu Befehlszeilen-Tools

Zusätzlich zur AWS-Befehlszeilenschnittstelle (CLI), die Sie für die Verwaltung Ihrer AWS-Ressourcen verwenden, AWS CloudHSM bietet es Befehlszeilentools zum Erstellen und Verwalten von HSM-Benutzern und Schlüsseln auf Ihren HSMs. In verwenden AWS CloudHSM Sie die vertraute CLI, um Ihren Cluster zu verwalten, und die CloudHSM-Befehlszeilentools, um Ihr HSM zu verwalten.

Dies sind die verschiedenen Befehlszeilen-Tools:

Verwalten von HSMs und Clustern

CloudHSMV2-Befehle in CLI und PowerShell HSM2-Cmdlets im Modul AWSPowerShell

- Mit diesen Tools werden Cluster und HSMs abgerufen, erstellt, gelöscht und markiert: AWS CloudHSM
- Um die Befehle in CloudHSMV2-Befehlen in der CLI zu verwenden, müssen Sie CLI installieren und konfigurieren.
- PowerShell HSM2-Cmdlets im AWSPowerShell Modul sind in einem PowerShell Windows-Modul und einem plattformübergreifenden Core-Modul verfügbar. PowerShell

So verwalten Sie HSM-Benutzer

CloudHSM-CLI

- Verwenden Sie die CloudHSM CLI, um Benutzer zu erstellen, zu löschen, aufzulisten, Benutzerpasswörter zu ändern, und die Benutzer-Multi-Faktor-Authentifizierung (MFA) zu aktualisieren. Es ist nicht in der AWS CloudHSM-Clientsoftware enthalten. Anleitungen zur Installation dieses Tools finden Sie unter CloudHSM CLI installieren und konfigurieren.

Hilfstools

Zwei Tools unterstützen Sie bei der Verwendung AWS CloudHSM von Tools und Softwarebibliotheken:

- Das Configure-Tool aktualisiert Ihre CloudHSM-Client-Konfigurationsdateien. Dies ermöglicht AWS CloudHSM die Synchronisation der HSMs in einem Cluster.

  AWS CloudHSM bietet zwei Hauptversionen, und Client SDK 5 ist die neueste Version. Es bietet eine Reihe von Vorteilen gegenüber Client-SDK 3 (der vorherigen Serie).
- Mit pkpspeed wird die Leistung der HSM-Hardware unabhängige von Software-Bibliotheken gemessen.

Tools für frühere SDKs

Verwenden Sie das Key Management Tool (KMU), um symmetrische Schlüssel und asymmetrische Schlüsselpaare zu erstellen, zu löschen, zu importieren und zu exportieren:

- key_mgmt_util. Dieses Tool ist Bestandteil der AWS CloudHSM -Clientsoftware.

Verwenden Sie das CloudHSM Management Tool (CMU), um HSM-Benutzer zu erstellen und zu löschen, einschließlich der Implementierung der Quorum-Authentifizierung von Benutzerverwaltungsaufgaben

- cloudhsm_mgmt_util. Dieses Tool ist Bestandteil der AWS CloudHSM -Clientsoftware.

Configure-Tool

AWS CloudHSM synchronisiert automatisch Daten zwischen allen Hardware-Sicherheitsmodulen (HSM) in einem Cluster. Das configure-Tool aktualisiert die HSM-Daten der von den

Synchronisationsmechanismen verwendeten Konfigurationsdateien. Nutzen Sie configure, um die HSM-Daten zu aktualisieren, bevor Sie die Befehlszeilen-Tools verwenden – insbesondere dann, wenn sich die HSMs im Cluster geändert haben.

AWS CloudHSM umfasst zwei Hauptversionen des Client-SDK:

- Client-SDK 5: Dies ist unser neuestes und standardmäßiges Client-SDK. Informationen zu den Vorteilen und Nutzen, die es bietet, finden Sie unter Vorteile von Client-SDK 5.
- Client-SDK 3: Dies ist unser älteres Client-SDK. Es enthält einen vollständigen Satz von Komponenten für die Kompatibilität von plattform- und sprachbasierten Anwendungen und Verwaltungstools.

Anweisungen zur Migration von Client SDK 3 auf Client SDK 5 finden Sie unterMigration von Client-SDK 3 zu Client-SDK 5.

Themen

Configure-Tool für das Client-SDK 5

Verwenden Sie das Client-SDK-5-Configure-Tool, um die clientseitigen Konfigurationsdateien zu aktualisieren.

Jede Komponente in Client-SDK 5 enthält ein Configure-Tool mit einer Bezeichnung der Komponente im Dateinamen des Configure-Tools. Die PKCS #11-Bibliothek für das Client-SDK 5 enthält beispielsweise ein Configure-Tool namens `configure-pkcs11` unter Linux oder `configure-pkcs11.exe` auf Windows.

Syntax

PKCS #11

```
configure-pkcs11[ .exe ]
              -a <ENI IP address>
              [--hsm-ca-cert <customerCA certificate file path>]
              [--cluster-id <cluster ID>]
              [--endpoint <endpoint>]
```

```
            [--region <region>]
            [--server-client-cert-file <client certificate file path>]
            [--server-client-key-file <client key file path>]
            [--log-level <error | warn | info | debug | trace>]
                 Default is <info>
            [--log-rotation <daily | weekly>]
                 Default is <daily>
            [--log-file <file name with path>]
                 Default is </opt/cloudhsm/run/cloudhsm-pkcs11.log>
                 Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-pkcs11.log>
            [--log-type <file | term>]
                 Default is <file>
            [-h | --help]
            [-V | --version]
            [--disable-key-availability-check]
            [--enable-key-availability-check]
            [--disable-validate-key-at-init]
            [--enable-validate-key-at-init]
                 This is the default for PKCS #11
```

OpenSSL

```
configure-dyn[ .exe ]
            -a <ENI IP address>
            [--hsm-ca-cert <customerCA certificate file path>]
            [--cluster-id <cluster ID>]
            [--endpoint <endpoint>]
            [--region <region>]
            [--server-client-cert-file <client certificate file path>]
            [--server-client-key-file <client key file path>]
            [--log-level <error | warn | info | debug | trace>]
                 Default is <error>
            [--log-type <file | term>]
                 Default is <term>
            [-h | --help]
            [-V | --version]
            [--disable-key-availability-check]
            [--enable-key-availability-check]
            [--disable-validate-key-at-init]
                 This is the default for OpenSSL
            [--enable-validate-key-at-init]
```

JCE

```
configure-jce[ .exe ]
             -a <ENI IP address>
             [--hsm-ca-cert <customerCA certificate file path>]
             [--cluster-id <cluster ID>]
             [--endpoint <endpoint>]
             [--region <region>]
             [--server-client-cert-file <client certificate file path>]
             [--server-client-key-file <client key file path>]
             [--log-level <error | warn | info | debug | trace>]
                  Default is <info>
             [--log-rotation <daily | weekly>]
                  Default is <daily>
             [--log-file <file name with path>]
                  Default is </opt/cloudhsm/run/cloudhsm-jce.log>
                  Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-jce.log>
             [--log-type <file | term>]
                  Default is <file>
             [-h | --help]
             [-V | --version]
             [--disable-key-availability-check]
             [--enable-key-availability-check]
             [--disable-validate-key-at-init]
                  This is the default for JCE
             [--enable-validate-key-at-init]
```

CloudHSM CLI

```
configure-cli[ .exe ]
             -a <ENI IP address>
             [--hsm-ca-cert <customerCA certificate file path>]
             [--cluster-id <cluster ID>]
             [--endpoint <endpoint>]
             [--region <region>]
             [--server-client-cert-file <client certificate file path>]
             [--server-client-key-file <client key file path>]
             [--log-level <error | warn | info | debug | trace>]
                  Default is <info>
             [--log-rotation <daily | weekly>]
                  Default is <daily>
             [--log-file <file name with path>]
```

```
                Default for Linux is </opt/cloudhsm/run/cloudhsm-cli.log>
                Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-cli.log>
            [--log-type <file | term>]
                Default setting is <file>
            [-h | --help]
            [-V | --version]
            [--disable-key-availability-check]
            [--enable-key-availability-check]
            [--disable-validate-key-at-init]
                This is the default for CloudHSM CLI
            [--enable-validate-key-at-init]
```

Erweiterte Konfigurationen

Eine Liste der erweiterten Konfigurationen, die speziell für das Client-SDK-5-Configure-Tool gelten, finden Sie unter Erweiterte Konfigurationen für das Client-SDK-5-Configure-Tool.

> **Important**
>
> Nachdem Sie Änderungen an Ihrer Konfiguration vorgenommen haben, müssen Sie Ihre Anwendung neu starten, damit die Änderungen wirksam werden.

Beispiele

Diese Beispiele zeigen, wie Sie das Configure-Tool für Client-SDK 5 verwenden.

Bootstrappen des Client-SDK 5

Example

In diesem Beispiel wird der `-a`-Parameter zur Aktualisierung der HSM-Daten für Client-SDK 5 verwendet. Um den `-a`-Parameter verwenden zu können, benötigen Sie die IP-Adresse für eines der HSMs in Ihrem Cluster.

PKCS #11 library

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 -a <HSM IP addresses>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" -a <HSM IP
 addresses>
```

OpenSSL Dynamic Engine

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn -a <HSM IP addresses>
```

JCE provider

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce -a <HSM IP addresses>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" -a <HSM IP addresses>
```

CloudHSM CLI

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IP addresses of the HSMs>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IP
 addresses of the HSMs>
```

> **Note**
>
> Sie können den `--cluster-id`-Parameter anstelle von `-a <HSM_IP_ADDRESSES>` verwenden. Informationen zu den Anforderungen für die Verwendung von `--cluster-id` finden Sie unter Configure-Tool für das Client-SDK 5.

Weitere Informationen zum Parameter `-a` erhalten Sie unter the section called “Parameter”.

Geben Sie den Cluster, die Region und den Endpunkt für Client-SDK 5 an

Example

In diesem Beispiel wird der `cluster-id`-Parameter verwendet, um das Client-SDK 5 durch einen `DescribeClusters`-Aufruf zu bootstrappen.

PKCS #11 library

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5 mit **`cluster-id`**

- Verwenden Sie die Cluster-ID`cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5 mit **`cluster-id`**

- Verwenden Sie die Cluster-ID`cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --cluster-
id cluster-1234567
```

OpenSSL Dynamic Engine

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5 mit **`cluster-id`**

- Verwenden Sie die Cluster-ID`cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567
```

JCE provider

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5 mit **`cluster-id`**

- Verwenden Sie die Cluster-ID`cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5 mit **`cluster-id`**

- Verwenden Sie die Cluster-ID`cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --cluster-
id cluster-1234567
```

CloudHSM CLI

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5 mit **`cluster-id`**

- Verwenden Sie die Cluster-ID`cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5 mit **`cluster-id`**

- Verwenden Sie die Cluster-ID`cluster-1234567`, um die IP-Adresse eines HSM in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-
id cluster-1234567
```

Sie können die Parameter `--region` und `--endpoint` in Kombination mit dem `cluster-id`-Parameter verwenden, um anzugeben, wie das System den `DescribeClusters`-Aufruf durchführt. Wenn sich die Region des Clusters beispielsweise von der Region unterscheidet, die als AWS-CLI-Standard konfiguriert ist, sollten Sie den `--region`-Parameter verwenden, um diese Region zu verwenden. Darüber hinaus haben Sie die Möglichkeit, den AWS CloudHSM API-Endpunkt anzugeben, der für den Anruf verwendet werden soll. Dies kann für verschiedene Netzwerkkonfigurationen erforderlich sein, z. B. für die Verwendung von VPC-Schnittstellenendpunkten, für die nicht den Standard-DNS-Hostnamen verwendet wird. AWS CloudHSM

PKCS #11 library

Um eine Linux-EC2-Instance mit einem benutzerdefinierten Endpunkt und einer benutzerdefinierten Region zu booten

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567 --
region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

So booten Sie eine Windows-EC2-Instance mit einem Endpunkt und einer Region

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe --cluster-
id cluster-1234567--region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

OpenSSL Dynamic Engine

Um eine Linux-EC2-Instance mit einem benutzerdefinierten Endpunkt und einer benutzerdefinierten Region zu booten

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567 --region us-
east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

JCE provider

Um eine Linux-EC2-Instance mit einem benutzerdefinierten Endpunkt und einer benutzerdefinierten Region zu booten

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567 --region us-
east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

So booten Sie eine Windows-EC2-Instance mit einem Endpunkt und einer Region

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

CloudHSM CLI

Um eine Linux-EC2-Instance mit einem benutzerdefinierten Endpunkt und einer benutzerdefinierten Region zu booten

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567 --region us-
east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

So booten Sie eine Windows-EC2-Instance mit einem Endpunkt und einer Region

- Verwenden Sie das Konfigurationstool, um die IP-Adresse eines HSM in Ihrem Cluster mit einer benutzerdefinierten Region und einem benutzerdefinierten Endpunkt anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

Weitere Hinweise zu den Parametern `--cluster-id`, `--region` und `--endpoint` finden Sie unter the section called “Parameter”.

Aktualisieren Sie das Client-Zertifikat und den Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung

Example

Dieses Beispiel zeigt, wie Sie die `--server-client-key-file` Parameter `server-client-cert-file` und verwenden, um SSL neu zu konfigurieren, indem Sie einen benutzerdefinierten Schlüssel und ein SSL-Zertifikat für angeben AWS CloudHSM

PKCS #11 library

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. Verwenden Sie mit einem PowerShell Interpreter das Konfigurationstool, um und anzugebenssl-client.crt. ssl-client.key

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
```

```
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

OpenSSL Dynamic Engine

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

JCE provider

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. Verwenden Sie bei einem PowerShell Interpreter das Konfigurationstool, um und anzugebenssl-client.crt. ssl-client.key

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

CloudHSM CLI

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Linux

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. Verwenden Sie das Configure-Tool, um `ssl-client.crt` und `ssl-client.key` anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

So verwenden Sie ein benutzerdefiniertes Zertifikat und einen Schlüssel für die gegenseitige TLS-Client-Server-Authentifizierung mit dem Client-SDK 5 unter Windows

1. Kopieren Sie Schlüssel und Zertifikat in das entsprechende Verzeichnis.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. Verwenden Sie bei einem PowerShell Interpreter das Konfigurationstool, um und anzugebenssl-client.crt. ssl-client.key

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

Weitere Hinweise zu den Parametern `server-client-cert-file` und `--server-client-key-file` finden Sie unter the section called “Parameter”.

Deaktivieren Sie die Einstellungen für die Haltbarkeit von Client-Schlüsseln

Example

In diesem Beispiel wird der `--disable-key-availability-check`-Parameter verwendet, um die Einstellungen für die Haltbarkeit von Clientschlüsseln zu deaktivieren. Um einen Cluster mit einem einzigen HSM auszuführen, müssen Sie die Einstellungen für die Haltbarkeit von Client-Schlüsseln deaktivieren.

PKCS #11 library

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Linux zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Um die Haltbarkeit von Client-Schlüsseln für Client-SDK 5 unter Windows zu deaktivieren

- Verwenden Sie das Configure-Tool, um die Einstellungen für die Haltbarkeit von Client-Schlüsseln zu deaktivieren.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-availability-check
```

Weitere Informationen zum Parameter `--disable-key-availability-check` erhalten Sie unter the section called “Parameter”.

Protokollierungsoptionen verwalten

Example

Das Client-SDK 5 verwendet die Parameter `log-file`, `log-level`, `log-rotation` und `log-type`, um die Protokollierung zu verwalten.

> **Note**
>
> Um Ihr SDK für serverlose Umgebungen wie AWS Fargate oder AWS Lambda zu konfigurieren, empfehlen wir Ihnen, Ihren AWS CloudHSM Protokolltyp auf zu konfigurieren. `term` Die Client-Protokolle werden in der für diese Umgebung konfigurierten Protokollgruppe CloudWatch Logs ausgegeben `stderr` und dort erfasst.

PKCS #11 library

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-pkcs11.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
  ```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-level info
  ```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type file --log-file <file name
   with path> --log-rotation daily --log-level info
  ```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type term --log-level info
  ```

OpenSSL Dynamic Engine

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:
  ```

Linux

```
stderr
```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-level info
  ```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type <file name> --log-file file --log-rotation daily --log-level info
  ```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
  ```

JCE provider

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
  ```
  ```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-level info
  ```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type file --log-file <file name> --
  log-rotation daily --log-level info
  ```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type term --log-level info
  ```

CloudHSM CLI

Standardspeicherort für Protokollspeicherort

- Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den folgenden Standardspeicherort:

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-cli.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
  ```

Um die Protokollierungsebene zu konfigurieren und andere Protokollierungsoptionen auf Standard zu setzen

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-level info
  ```

Um Optionen für die Dateiprotokollierung zu konfigurieren

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type file --log-file <file name> --log-rotation daily --log-level info
  ```

Um Optionen für die Terminalprotokollierung zu konfigurieren

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type term --log-level info
  ```

Weitere Informationen über die Parameter `log-file`, `log-level`, `log-rotation` und `log-type` finden Sie unter the section called “Parameter”.

Platzieren Sie das ausstellende Zertifikat für das Client-SDK 5

Example

In diesem Beispiel wird der `--hsm-ca-cert`-Parameter verwendet, um den Speicherort des ausstellenden Zertifikats für das Client-SDK 5 zu aktualisieren.

PKCS #11 library

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

  ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --hsm-ca-cert <customerCA certificate file>
  ```

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --hsm-ca-
cert <customerCA certificate file>
```

OpenSSL Dynamic Engine

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --hsm-ca-cert <customerCA certificate
 file>
```

JCE provider

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-jce --hsm-ca-cert <customerCA certificate
 file>
```

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --hsm-ca-cert <customerCA
 certificate file>
```

CloudHSM CLI

Um das ausstellende Zertifikat für das Client-SDK 5 auf Linux zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli --hsm-ca-cert <customerCA certificate
 file>
```

Um das ausstellende Zertifikat unter Windows für Client-SDK 5 zu platzieren

- Verwenden Sie das Configure-Tool, um einen Speicherort für das ausstellende Zertifikat anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --hsm-ca-cert <customerCA
 certificate file>
```

Weitere Informationen zum Parameter `--hsm-ca-cert` erhalten Sie unter the section called “Parameter”.

Parameter

-a ***<ENI IP address>***

Fügt die angegebene IP-Adresse den Client-SDK 5-Konfigurationsdateien hinzu. Geben Sie eine beliebige ENI-IP-Adresse eines HSM aus dem Cluster ein. Weitere Informationen über die Verwendung dieser Option finden Sie unter Bootstrap für Client-SDK 5.

Erforderlich: Ja

-- hsm-ca-cert <customerCA certificate file path>

Pfad zu dem Verzeichnis, in dem das Zertifikat der Zertifizierungsstelle (CA) gespeichert ist, mit dem EC2-Client-Instances mit dem Cluster verbunden werden. Sie erstellen diese Datei, wenn Sie

den Cluster initialisieren. Standardmäßig sucht das System am folgenden Speicherort nach dieser Datei:

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

Weitere Informationen zur Initialisierung des Clusters oder zum Platzieren des Zertifikats finden Sie unter ??? und ???.

Erforderlich: Nein

--cluster-id ***<cluster ID>***

Führt einen `DescribeClusters`-Aufruf aus, um alle IP-Adressen der HSM-Elastic-Network-Schnittstelle (ENI) im Cluster mit der Cluster-ID zu finden. Das System fügt die ENI-IP-Adressen zu den AWS CloudHSM Konfigurationsdateien hinzu.

> **Note**
>
> Wenn Sie den `--cluster-id` Parameter von einer EC2-Instance innerhalb einer VPC verwenden, die keinen Zugriff auf das öffentliche Internet hat, müssen Sie einen VPC-Schnittstellen-Endpunkt erstellen, mit dem Sie eine Verbindung herstellen können. AWS CloudHSM Weitere Informationen über VPC-Endpunkte finden Sie unter ???.

Erforderlich: Nein

--endpoint ***<endpoint>***

Geben Sie den AWS CloudHSM API-Endpunkt an, der für den Aufruf verwendet wird. `DescribeClusters` Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Erforderlich: Nein

--region ***<region>***

Geben Sie die Region Ihres Clusters an. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Wenn Sie den `--region`-Parameter nicht angeben, wählt das System die Region aus, indem es versucht, die Umgebungsvariablen `AWS_DEFAULT_REGION` oder `AWS_REGION` zu lesen. Wenn diese Variablen nicht festgelegt sind, überprüft das System die Region, die Ihrem Profil in Ihrer AWS-Config-Datei zugeordnet ist (normalerweise `~/.aws/config`), sofern Sie in der `AWS_CONFIG_FILE`-Umgebungsvariable keine andere Datei angegeben haben. Wenn keine der oben genannten Optionen festgelegt ist, verwendet das System standardmäßig die `us-east-1`-Region.

Erforderlich: Nein

-- server-client-cert-file <client certificate file path>

Pfad zum Client-Zertifikat, das für die gegenseitige TLS-Client-Server-Authentifizierung verwendet wird.

Verwenden Sie diese Option nur, wenn Sie nicht den Standardschlüssel und das SSL/TLS-Zertifikat verwenden möchten, die im Client-SDK 5 enthalten sind. Sie müssen diese Option in Kombination mit `--server-client-key-file` festlegen.

Erforderlich: Nein

-- server-client-key-file <client key file path>

Pfad zum Client-Schlüssel, der für die gegenseitige TLS-Client-Server-Authentifizierung verwendet wird.

Verwenden Sie diese Option nur, wenn Sie nicht den Standardschlüssel und das SSL/TLS-Zertifikat verwenden möchten, die im Client-SDK 5 enthalten sind. Sie müssen diese Option in Kombination mit `--server-client-cert-file` festlegen.

Erforderlich: Nein

--log-level ***<error | warn | info | debug | trace>***

Gibt die Mindestprotokollierungsebene an, die das System in die Protokolldatei schreiben soll. Jede Stufe umfasst die vorherigen Stufen, wobei Fehler die Mindeststufe und Trace die Höchststufe ist. Das heißt, wenn Sie Fehler angeben, schreibt das System nur Fehler in das Protokoll. Wenn Sie trace angeben, schreibt das System Fehler, Warnungen, Informations- (Info-) und Debugmeldungen in das Protokoll. Weitere Informationen finden Sie unter Client-SDK-5-Protokollierung.

Erforderlich: Nein

--log-rotation ***<daily | weekly>***

Gibt die Häufigkeit an, mit der das System Protokolle rotiert. Weitere Informationen finden Sie unter Client-SDK-5-Protokollierung.

Erforderlich: Nein

--log-file ***<file name with path>***

Gibt an, wo das System die Protokolldatei schreiben wird. Weitere Informationen finden Sie unter Client-SDK-5-Protokollierung.

Erforderlich: Nein

--log-type ***<term | file>***

Gibt an, ob das System das Protokoll in eine Datei oder ein Terminal schreibt. Weitere Informationen finden Sie unter Client-SDK-5-Protokollierung.

Erforderlich: Nein

-h | --help

Zeigt Hilfe an.

Erforderlich: Nein

-v | --version

Zeigt die Version an.

Erforderlich: Nein

--disable-key-availability-check

Markierung, um das Quorum für die Schlüsselverfügbarkeit zu deaktivieren. Verwenden Sie dieses Flag, um anzugeben, dass das Schlüsselverfügbarkeitsquorum deaktiviert werden AWS CloudHSM soll und dass Sie Schlüssel verwenden können, die nur auf einem HSM im Cluster vorhanden sind. Weitere Hinweise zur Verwendung dieses Flags zum Festlegen eines Quorums für Schlüsselverfügbarkeit finden Sie unter ???.

Erforderlich: Nein

--enable-key-availability-check

Markierung, um das Quorum für die Schlüsselverfügbarkeit zu aktivieren. Verwenden Sie dieses Flag, um anzugeben, dass das Schlüsselverfügbarkeitsquorum verwendet werden AWS

CloudHSM soll und dass Sie Schlüssel erst verwenden dürfen, wenn diese Schlüssel auf zwei HSMs im Cluster vorhanden sind. Weitere Hinweise zur Verwendung dieses Flags zum Festlegen eines Quorums für Schlüsselverfügbarkeit finden Sie unter ???.

Standardmäßig aktiviert.

Erforderlich: Nein

-- -init disable-validate-key-at

Verbessert die Leistung, indem angegeben wird, dass Sie einen Initialisierungsaufruf überspringen können, um die Berechtigungen für einen Schlüssel für nachfolgende Aufrufe zu überprüfen. Verwenden Sie es mit Bedacht.

Hintergrund: Einige Mechanismen in der PKCS #11-Bibliothek unterstützen mehrteilige Operationen, bei denen ein Initialisierungsaufruf überprüft, ob Sie den Schlüssel für nachfolgende Aufrufe verwenden können. Dies erfordert einen Bestätigungsaufruf an das HSM, was die Latenz des gesamten Vorgangs erhöht. Mit dieser Option können Sie den nachfolgenden Aufruf deaktivieren und möglicherweise die Leistung verbessern.

Erforderlich: Nein

-- -init enable-validate-key-at

Gibt an, dass Sie einen Initialisierungsaufruf verwenden sollten, um die Berechtigungen für einen Schlüssel für nachfolgende Aufrufe zu überprüfen. Dies ist die Standardoption. Verwenden Sie`enable-validate-key-at-init`, um diese Initialisierungsrufe wieder aufzunehmen, nachdem Sie mit `disable-validate-key-at-init` sie unterbrochen haben.

Erforderlich: Nein

Verwandte Themen

- DescribeClusters API-Operation
- describe-clusters AWS CLI
- Cmdlet Get-Hsm2Cluster PowerShell
- Bootstrappen des Client-SDK 5
- AWS CloudHSM VPC-Endpunkte
- Verwaltung der wichtigsten Haltbarkeitseinstellungen des Client-SDK 5
- Protokollierung im Client-SDK 5

Erweiterte Konfigurationen für das Client-SDK-5-Configure-Tool

Das Client-SDK-5-Configure-Tool umfasst erweiterte Konfigurationen, die nicht zu den allgemeinen Funktionen gehören, die die meisten Kunden verwenden. Erweiterte Konfigurationen bieten zusätzliche Funktionen.

- Erweiterte Konfigurationen für PKCS #11
  - Verbindung zu mehreren Steckplätzen mit PKCS #11
  - Wiederholungsbefehle für PKCS #11
- Erweiterte Konfigurationen für JCE
  - Verbindung zu mehreren Clustern mit dem JCE-Anbieter herstellen
  - Wiederholungsbefehle für JCE
  - Schlüsselextraktion mit JCE
- Erweiterte Konfigurationen für OpenSSL
  - Wiederholungsbefehle für OpenSSL
- Erweiterte Konfigurationen für die Befehlszeilenschnittstelle (CLI)
  - Mit CLI eine Verbindung zu mehreren Clustern herstellen

Configure-Tool für das Client-SDK 3

Verwenden Sie das Client-SDK-3-Configure-Tool, um den Client-Daemon zu booten und das CloudHSM Management Utility zu konfigurieren.

Syntax

```
configure -h | --help
          -a <ENI IP address>
          -m [-i <daemon_id>]
          --ssl --pkey <private key file> --cert <certificate file>
          --cmu <ENI IP address>
```

Beispiele

Diese Beispiele zeigen die Verwendung des configure-Tools.

Example : Aktualisieren Sie die HSM-Daten für den Client und key_mgmt_util AWS CloudHSM

In diesem Beispiel werden die `-a` Parameter von configure zur Aktualisierung der HSM-Daten für den Client und key_mgmt_util verwendet. AWS CloudHSM Um den `-a`-Parameter verwenden zu können, benötigen Sie die IP-Adresse für eines der HSMs in Ihrem Cluster. Verwenden Sie entweder die Konsole oder die AWS-CLI, um die IP-Adresse abzurufen.

Um eine IP-Adresse für ein HSM (Konsole) zu erhalten

1. Öffnen Sie die AWS CloudHSM Konsole unter https://console.aws.amazon.com/cloudhsm/home.
2. Um die AWS-Region zu ändern, verwenden Sie die Regionsauswahl in der oberen rechten Ecke der Seite.
3. Um die Cluster-Detailseite zu öffnen, wählen Sie in der Cluster-Tabelle die Cluster-ID aus.
4. Um die IP-Adresse abzurufen, wählen Sie auf der Registerkarte HSMs eine der IP-Adressen aus, die unter ENI-IP-Adresse aufgeführt sind.

Um eine IP-Adresse für ein HSM (CLI) zu erhalten

- Rufen Sie die IP-Adresse eines HSM mit dem describe-clusters Befehl von der CLI ab. In der Ausgabe des Befehls sind die IP-Adressen der HSMs die Werte von `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

So aktualisieren Sie die HSM-Daten

1. Stoppen Sie den AWS CloudHSM Client, bevor Sie den -a Parameter aktualisieren. Dies verhindert Konflikte, die bei der Bearbeitung der Client-Konfigurationsdatei durch configure auftreten können. Wenn der Client bereits angehalten ist, hat dieser Befehl keine Auswirkungen, sodass Sie ihn in einem Skript verwenden können.

   Amazon Linux

   ```
   $ sudo stop cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client stop
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client stop
   ```

   CentOS 8

   ```
   $ sudo service cloudhsm-client stop
   ```

   RHEL 7

   ```
   $ sudo service cloudhsm-client stop
   ```

   RHEL 8

   ```
   $ sudo service cloudhsm-client stop
   ```

   Ubuntu 16.04 LTS

   ```
   $ sudo service cloudhsm-client stop
   ```

   Ubuntu 18.04 LTS

   ```
   $ sudo service cloudhsm-client stop
   ```

Windows

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

  Verwenden Sie Strg + C in dem Befehlsfenster, in dem Sie den AWS CloudHSM Client gestartet haben.

2. Dieser Schritt verwendet den `-a`-Parameter von configure, um den Konfigurationsdateien die ENI-IP-Adresse `10.0.0.9` hinzuzufügen.

   Amazon Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
   ```

   Amazon Linux 2

   ```
   $ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
   ```

   CentOS 7

   ```
   $ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
   ```

   CentOS 8

   ```
   $ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
   ```

   RHEL 7

   ```
   $ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
   ```

   RHEL 8

   ```
   $ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
   ```

Ubuntu 16.04 LTS

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Ubuntu 18.04 LTS

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -a 10.0.0.9
```

3. Starten Sie als Nächstes den AWS CloudHSM Client neu. Wenn der -Client startet, wird die ENI IP-Adresse aus der Konfigurationsdatei des Clients verwendet, um den Cluster abzufragen. Dann schreibt er die ENI-IP-Adressen aller HSMs im Cluster in die Datei `cluster.info`.

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 8

   ```
   $ sudo service cloudhsm-client start
   ```

   RHEL 7

   ```
   $ sudo service cloudhsm-client start
   ```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Wenn der Befehl abgeschlossen ist, sind die HSM-Daten, die der AWS CloudHSM Client und key_mgmt_util verwenden, vollständig und korrekt.

Example : Aktualisieren Sie die HSM-Daten für CMU aus dem Client-SDK 3.2.1 und früher

In diesem Beispiel wird der `-m` configure-Befehl verwendet, um die aktualisierten HSM-Daten aus der `cluster.info`-Datei in die `cloudhsm_mgmt_util.cfg`-Datei zu kopieren, die cloudhsm_mgmt_util verwendet. Verwenden Sie dies mit der CMU, die im Client-SDK 3.2.1 und früher enthalten ist.

- Bevor Sie den ausführen`-m`, beenden Sie den AWS CloudHSM Client, führen Sie den `-a` Befehl aus und starten Sie den AWS CloudHSM Client neu, wie im vorherigen Beispiel gezeigt. Dadurch wird sichergestellt, dass die Daten, die aus der Datei `cloudhsm_mgmt_util.cfg` in die Datei `cluster.info` kopiert werden, vollständig und korrekt sind.

Linux

```
$ sudo /opt/cloudhsm/bin/configure -m
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -m
```

Example : Aktualisieren Sie die HSM-Daten für CMU aus dem Client-SDK 3.3.0 und höher

In diesem Beispiel wird der `--cmu`-Parameter des configure-Befehls zur Aktualisierung der HSM-Daten für CMU verwendet. Verwenden Sie dies mit der CMU, die im Client-SDK 3.3.0 und höher enthalten ist. Weitere Informationen zur Verwendung von CMU finden Sie unter Verwendung des CloudHSM Management Utility (CMU) zum Verwalten von Benutzern und Verwendung des CMU mit Client-SDK 3.2.1 und früher.

- Verwenden Sie den `--cmu`-Parameter, um die IP-Adresse eines HSM in Ihrem Cluster zu übergeben.

  Linux

  ```
  $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
  ```

Parameter

-h | --help

Zeigt die Befehlssyntax an.

Erforderlich: Ja

-a *<ENI IP address>*

Fügt die angegebene IP-Adresse der HSM-Elastic Network-Schnittstelle (ENI) zu den Konfigurationsdateien von AWS CloudHSM hinzu. Geben Sie die ENI-IP-Adresse eines beliebigen HSMs im Cluster ein. Dabei spielt es keine Rolle, welche Sie auswählen.

Um die ENI-IP-Adressen der HSMs in Ihrem Cluster abzurufen, verwenden Sie den DescribeClustersVorgang, den CLI-Befehl describe-clusters oder das Cmdlet Get-Hsm2Cluster. PowerShell

> **Note**
>
> Stoppen Sie den Client, bevor Sie den Befehl ausführen. `-a` configure AWS CloudHSM Wenn der `-a` Befehl abgeschlossen ist, starten Sie den AWS CloudHSM Client neu. Details finden Sie in den Beispielen.

Dieser Parameter bearbeitet die folgenden Konfigurationsdateien:

- `/opt/cloudhsm/etc/cloudhsm_client.cfg`: Wird von AWS CloudHSM client und key_mgmt_util verwendet.
- `/opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg`: Wird von cloudhsm_mgmt_util verwendet.

Wenn der AWS CloudHSM Client gestartet wird, verwendet er die ENI-IP-Adresse in seiner Konfigurationsdatei, um den Cluster abzufragen und die `cluster.info` Datei (`/opt/cloudhsm/daemon/1/cluster.info`) mit den richtigen ENI-IP-Adressen für alle HSMs im Cluster zu aktualisieren.

Erforderlich: Ja

-m

Aktualisiert die HSM-ENI-IP-Adressen in der von CMU verwendeten Konfigurationsdatei.

> **Note**
>
> Der `-m`-Parameter ist für die Verwendung mit CMU aus Client-SDK 3.2.1 und früheren Versionen vorgesehen. Informationen zu CMU aus dem Client-SDK 3.3.0 und höher finden Sie unter `--cmu`-Parameter, der das Aktualisieren von HSM-Daten für CMU vereinfacht.

Wenn Sie den `-a` Parameter von aktualisieren configure und dann den AWS CloudHSM Client starten, fragt der Client-Daemon den Cluster ab und aktualisiert die `cluster.info` Dateien mit den richtigen HSM-IP-Adressen für alle HSMs im Cluster. Durch das Ausführen des Befehls `-m` configure wird die Aktualisierung abgeschlossen, da die HSM-IP-Adressen aus der Datei `cluster.info` in die von cloudhsm_mgmt_util verwendete `cloudhsm_mgmt_util.cfg`-Konfigurationsdatei kopiert werden.

Stellen Sie sicher, dass Sie den `-a` configure Befehl ausführen und den AWS CloudHSM Client neu starten, bevor Sie den Befehl ausführen. `-m` Dadurch wird sichergestellt, dass die Daten, die aus `cloudhsm_mgmt_util.cfg` in `cluster.info` kopiert werden, vollständig und korrekt sind.

Erforderlich: Ja

-i

Gibt einen alternativen Client-Daemon an. Der Standardwert stellt den AWS CloudHSM -Client dar.

Standard: 1

Erforderlich: Nein

--ssl

Ersetzt den SSL-Schlüssel und das Zertifikat für den Cluster durch den angegebenen privaten Schlüssel und das Zertifikat. Wenn Sie diesen Parameter verwenden, müssen die Parameter `--pkey` und `--cert` verwendet werden.

Erforderlich: Nein

--pkey

Gibt den neuen privaten Schlüssel an. Geben Sie den Pfad und den Namen der Datei an, die den privaten Schlüssel enthält.

Erforderlich: Ja, wenn --ssl angegeben ist. Andernfalls sollte dieser Wert nicht verwendet werden.

--cert

Gibt das neue Zertifikat an. Geben Sie den Pfad und den Namen der Datei an, die das Zertifikat enthält. Das Zertifikat sollte mit dem Zertifikat `customerCA.crt`, dem selbstsignierten Zertifikat, das zum Initialisieren des Clusters verwendet wurde, verkettet sein. Weitere Informationen finden Sie unter Initialisieren des Clusters.

Erforderlich: Ja, wenn --ssl angegeben ist. Andernfalls sollte dieser Wert nicht verwendet werden.

--cmu ***<ENI IP address>***

Kombiniert die Parameter `-a` und `-m` zu einem Parameter. Fügt den AWS CloudHSM Konfigurationsdateien die angegebene HSM elastic network interface (ENI) -IP-Adresse hinzu und aktualisiert dann die CMU-Konfigurationsdatei. Geben Sie eine IP-Adresse von einem beliebigen HSM im Cluster ein. Informationen zum Client-SDK 3.2.1 und früher finden Sie unter Verwenden von CMU mit Client-SDK 3.2.1 und früher.

Erforderlich: Ja

Verwandte Themen

- Einrichten von key_mgmt_util

CloudHSM-Befehlszeilenschnittstelle (CLI)

CloudHSM CLI unterstützt Administratoren bei der Verwaltung von Benutzern und Krypto-Benutzern bei der Verwaltung von Schlüsseln in ihrem Cluster. Es enthält Tools, mit denen Benutzer erstellt, gelöscht und aufgelistet, Benutzerkennwörter geändert und die Benutzer-Multifaktor-Authentifizierung (MFA) aktualisiert werden können. Es enthält auch Befehle zum Generieren, Löschen, Importieren und Exportieren von Schlüsseln, zum Abrufen und Festlegen von Attributen, zum Suchen von Schlüsseln und zum Ausführen kryptografischer Operationen.

Eine definierte Liste von CloudHSM-CLI-Benutzern finden Sie unter HSM-Benutzer mit CloudHSM CLI verwalten . Eine definierte Liste von Schlüsselattributen für die CloudHSM-CLI finden Sie unter. Schlüsselattribute für CloudHSM-CLI Informationen zur Verwendung der CloudHSM-CLI zur Verwaltung von Schlüsseln finden Sie unter. Schlüssel mit CloudHSM-CLI verwalten

Informationen zum schnellen Einstieg finden Sie unter Erste Schritte mit der CloudHSM-Befehlszeilenschnittstelle (CLI). Detaillierte Informationen über die CloudHSM-CLI-Befehle und Beispiele zu ihrer Verwendung finden Sie unter Referenz für CloudHSM-CLI-Befehle.

Themen

- Von der CloudHSM-Befehlszeilenschnittstelle (CLI) unterstützte Plattformen
- Erste Schritte mit der CloudHSM-Befehlszeilenschnittstelle (CLI)
- Interaktive Modi und Einzelbefehlsmodi

- Schlüsselattribute für CloudHSM-CLI
- Migrieren von Client-SDK 3 CMU und KMU zu Client-SDK 5 CloudHSM CLI
- Erweiterte Konfigurationen für CLI
- Referenz für CloudHSM-CLI-Befehle

Von der CloudHSM-Befehlszeilenschnittstelle (CLI) unterstützte Plattformen

Linux-Support

Unterstützte Plattformen	X86_64-Architektur	ARM-Architektur
Amazon Linux 2	Ja	Ja
Amazon Linux 2023	Ja	Ja
CentOS 7 (7,8+)	Ja	Nein
RedHat Enterprise Linux 7 (7.8+)	Ja	Nein
RedHat Enterprise Linux 8 (8,3 und höher)	Ja	Nein
RedHat Enterprise Linux 9 (9,2 und höher)	Ja	Ja
Ubuntu 20.04 LTS	Ja	Nein
Ubuntu 22.04 LTS	Ja	Ja

Hinweis: SDK 5.4.2 war die letzte Version, die CentOS 8-Plattformunterstützung bot. Weitere Informationen finden Sie auf der CentOS-Website.

Windows-Unterstützung

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Erste Schritte mit der CloudHSM-Befehlszeilenschnittstelle (CLI)

Mit der CloudHSM-Befehlszeilenschnittstelle (CLI) können Sie Benutzer in Ihrem AWS CloudHSM Cluster verwalten. Erfahren Sie in diesem Thema, um mit grundlegenden HSM-Benutzerverwaltungsaufgaben wie dem Erstellen von Benutzern, dem Auflisten von Benutzern und dem Verbinden der CloudHSM-CLI mit dem Cluster zu beginnen.

Installieren Sie die CloudHSM-CLI

Verwenden Sie die folgenden Befehle, um die CloudHSM-CLI herunterzuladen und zu installieren.

Amazon Linux 2

Amazon Linux 2 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

Amazon Linux 2 auf ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.aarch64.rpm
```

Amazon Linux 2023

Amazon Linux 2023 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

Amazon Linux 2023 auf ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

CentOS 7 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-
latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

RHEL 7 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-
latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

RHEL 8 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-
latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

RHEL 9 auf x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-
latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.x86_64.rpm
```

RHEL 9 auf ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

Ubuntu 20.04 LTS auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-cli_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_amd64.deb
```

Ubuntu 22.04 LTS auf ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_arm64.deb
```

Windows Server 2016

Öffnen Sie Windows Server 2016 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

Öffnen Sie Windows Server 2019 auf einer x86_64-Architektur PowerShell als Administrator und führen Sie den folgenden Befehl aus:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

Verwenden Sie die folgenden Befehle, um CloudHSM-CLI zu konfigurieren.

So bootstrappen Sie eine Linux-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IP addresses of the HSMs>
```

So bootstrappen Sie eine Windows-EC2-Instance für Client-SDK 5

- Verwenden Sie das Configure-Tool, um die IP-Adresse der HSM(s) in Ihrem Cluster anzugeben.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IP addresses
 of the HSMs>
```

Verwenden der CloudHSM-CLI

1. Verwenden Sie den folgenden Befehl, um die CloudHSM-CLI zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. Verwenden Sie den login-Befehl, um sich beim Cluster anzumelden. Alle Benutzer können diesen Befehl verwenden.

   Mit dem Befehl im folgenden Beispiel wird admin angemeldet, das Standard-Admin-Konto. Sie legen das Passwort für diesen Benutzer nach der Aktivierung des Clusters fest.

   ```
   aws-cloudhsm > login --username admin --role admin
   ```

   Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Sie geben das Passwort ein und die Ausgabe zeigt, dass der Befehl erfolgreich war.

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

3. Führen Sie den user list-Befehl aus, um alle Benutzer im Cluster aufzulisten.

   ```
   aws-cloudhsm > user list
   {
     "error_code": 0,
     "data": {
       "users": [
         {
           "username": "admin",
   ```

```
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

4. Verwenden Sie user create, um einen CU-Benutzer mit dem Namen **example_user** zu erstellen.

   Sie können CUs erstellen, weil Sie sich in einem vorherigen Schritt als Admin-Benutzer angemeldet haben. Nur Admin-Benutzer können Benutzerverwaltungsaufgaben wie das Erstellen und Löschen von Benutzern und das Ändern der Passwörter anderer Benutzer ausführen.

```
aws-cloudhsm > user create --username example_user --role crypto-user
Enter password:
Confirm password:
{
 "error_code": 0,
 "data": {
   "username": "example_user",
   "role": "crypto-user"
 }
}
```

5. Verwenden Sie user list, um alle Benutzer im Cluster aufzulisten.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
```

```
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "example_user",
        "role": "crypto_user",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

6. Verwenden Sie den logout Befehl, um sich vom Cluster abzumelden. AWS CloudHSM

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

7. Verwenden Sie den quit-Befehl, um die CLI zu stoppen.

```
aws-cloudhsm > quit
```

Interaktive Modi und Einzelbefehlsmodi

In der CloudHSM-CLI können Sie Befehle auf zwei verschiedene Arten ausführen: im Einzelbefehlsmodus und im interaktiven Modus. Der interaktive Modus ist für Benutzer konzipiert, und der Einzelbefehlsmodus ist für Skripts konzipiert.

> **Note**
>
> Alle Befehle funktionieren im interaktiven Modus und im Einzelbefehlsmodus.

interaktiver Modus

Verwenden Sie die folgenden Befehle, um den interaktiven CloudHSM-CLI-Modus zu starten

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

Wenn Sie die CLI im interaktiven Modus verwenden, können Sie sich mit dem login-Befehl bei einem Benutzerkonto anmelden.

Um alle CloudHSM-CLI-Befehle auszuführen, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm > help
```

Um die Syntax für einen CloudHSM-CLI-Befehl anzufordern, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm > help <command-name>
```

Wenn Sie eine Liste von Benutzern der HSMs abrufen möchten, geben Sie user list ein.

```
aws-cloudhsm > user list
```

Um Ihre CloudHSM-CLI-Sitzung zu beenden, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm > quit
```

Einzelbefehlsmodus

Wenn Sie die CloudHSM-CLI im Einzelbefehlsmodus ausführen, müssen Sie zwei Umgebungsvariablen setzen, um die Anmeldedaten bereitzustellen: CLOUDHSM_PIN und CLOUDHSM_ROLE:

```
$ export CLOUDHSM_ROLE=admin
```

```
$ export CLOUDHSM_PIN=admin_username:admin_password
```

Danach können Sie Befehle mit den in Ihrer Umgebung gespeicherten Anmeldeinformationen ausführen.

```
$ cloudhsm-cli user change-password --username alice --role crypto-user
Enter password:
Confirm password:
{
    "error_code": 0,
    "data": {
      "username": "alice",
      "role": "crypto-user"
    }
}
```

Schlüsselattribute für CloudHSM-CLI

In diesem Thema wird beschrieben, wie Sie die CloudHSM-CLI zum Festlegen von Schlüsselattributen verwenden. Ein Schlüsselattribut in der CloudHSM-CLI kann den Typ eines Schlüssels definieren, wie ein Schlüssel funktionieren kann oder wie ein Schlüssel beschriftet wird. Einige Attribute definieren eindeutige Merkmale (z. B. den Typ eines Schlüssels). Andere Attribute können auf wahr oder falsch gesetzt werden. Wenn Sie sie ändern, wird entweder ein Teil der Funktionalität des Schlüssels aktiviert oder deaktiviert.

Beispiele zur Verwendung von Schlüsselattributen finden Sie in den Befehlen, die unter dem übergeordneten Befehl Schlüssel aufgeführt sind.

Unterstützte Attribute

Als bewährte Methode legen Sie nur Werte für Attribute fest, die Sie einschränken möchten. Wenn Sie keinen Wert angeben, verwendet CloudHSM-CLI den in der folgenden Tabelle angegebenen Standardwert.

In der folgenden Tabelle sind die wichtigsten Attribute, möglichen Werte, Standardwerte und zugehörige Hinweise aufgeführt. Eine leere Zelle in der Spalte Wert gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
`always-sensitive`	Der Wert ist `True`, wenn `sensitive` immer auf `True` gesetzt wurde und sich nie geändert hat.	Nein	Nein
`check-value`	Der Prüfwert des Schlüssels. Weitere Informationen finden Sie unter Weitere Details.	Nein	Nein
`class`	Die möglichen Werte lauten: `secret-key` , `public-key` und `private-key` .	Nein	Ja
`curve`	Elliptische Kurve, die zur Generierung des EC-Schlüsselpaars verwendet wird. Gültige Werte: `secp224r1` , `secp256r1` ,	Nein	Einstellbar mit RSA, nicht einstellbar mit EC

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
	`prime256v1` , `secp384r1` , `secp256k1` und `secp521r1`		
`decrypt`	Standard: `False`	Ja	Ja
`derive`	Standard: `False`	Ja	Ja
`destroyable`	Standard: `True`	Ja	Ja
`ec-point`	Für EC-Schlüssel: DER-Kodierung des ANSI X9.62-ECPoint-Werts „Q“ in einem Hexadezimalformat. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.	Nein	Nein
`encrypt`	Standard: `False`	Ja	Ja
`extractable`	Standard: `True`	Nein	Ja
`id`	Standard: leer	Nein	Ja
`key-length-bytes`	Erforderlich für die Generierung eines AES-Schlüssels. Gültige Werte: 16, 24 und 32 Byte.	Nein	Nein

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
`key-type`	Die möglichen Werte lauten: `aes`, `rsa` und `ec`.	Nein	Ja
`label`	Standard: leer	Ja	Ja
`local`	Standard: `True` für Schlüssel, die im HSM generiert wurden, `False` für Schlüssel, die in das HSM importiert wurden.	Nein	Nein
`modifiable`	Standard: `True`	Nein	Nein
`modulus`	Der Modulus, der zum Generieren eines RSA-Schlüsselpaares verwendet wurde. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.	Nein	Nein
`modulus-size-bits`	Erforderlich für die Generierung eines RSA-Schlüsselpaars. Der Mindestwert ist `2048`.	Nein	Einstellbar mit RSA, nicht einstellbar mit EC

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
`never-extractable`	Der Wert ist `True`, wenn extrahierbar noch nie auf `False` gesetzt wurde. Der Wert ist `False`, wenn extrahierbar jemals auf `True` gesetzt wurde.	Nein	Nein
`private`	Standard: `True`	Nein	Ja
`public-exponent`	Erforderlich für die Generierung eines RSA-Schlüsselpaars. Gültige Werte: Bei diesem Wert muss es sich eine ungerade Zahl gleich oder größer als 65537 handeln.	Nein	Einstellbar mit RSA, nicht einstellbar mit EC
`sensitive`	Standard: • Der Wert ist `True` für AES-Schlüssel und private EC- und RSA-Schlüssel. • Der Wert ist `False` für öffentliche EC- und RSA-Schlüssel.	Nein	Einstellbar mit privaten Schlüsseln, nicht einstellbar mit öffentlichen Schlüsseln.

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
`sign`	Standard: • Der Wert ist `True` für AES-Schlüssel. • Der Wert ist `False` für RSA- und EC-Schlüssel.	Ja	Ja
`token`	Standard: `False`	Nein	Ja
`trusted`	Standard: `False`	Ja	Nein
`unwrap`	Standard: `False`	Ja	Ja
`unwrap-template`	Für die Werte sollte die Attributv orlage verwendet werden, die auf jeden Schlüssel angewende t wird, der mit diesem Schlüssel zum Packen entpackt wird.	Ja	Nein
`verify`	Standard: • Der Wert ist `True` für AES-Schlüssel. • Der Wert ist `False` für RSA- und EC-Schlüssel.	Ja	Ja
`wrap`	Standard: `False`	Ja	Ja

CloudHSM-CLI-Attribut	Wert	Modifizierbar mit key set-attribute	Kann bei der Schlüsselerstellung eingestellt werden
`wrap-template`	Für die Werte sollte die Attributvorlage verwendet werden, die dem Schlüssel entspricht, der mit diesem Schlüssel zum Packen gepackt wurde.	Ja	Nein
`wrap-with-trusted`	Standard: `False`	Ja	Ja

Weitere Details

Prüfwert

Der Prüfwert ist ein 3-Byte-Hash oder eine Prüfsumme eines Schlüssels, der generiert wird, wenn das HSM einen Schlüssel importiert oder generiert. Sie können einen Prüfwert auch außerhalb des HSM berechnen, z. B. nachdem Sie einen Schlüssel exportiert haben. Anschließend können Sie die Prüfwerte vergleichen, um die Identität und Integrität des Schlüssels zu bestätigen. Um den Prüfwert eines Schlüssels zu ermitteln, verwenden Sie die Schlüsselliste mit dem verbose-Flag.

AWS CloudHSM verwendet die folgenden Standardmethoden, um einen Prüfwert zu generieren:

- Symmetrische Schlüssel: Die ersten 3 Byte des Ergebnisses der Verschlüsselung eines Nullblocks mit dem Schlüssel.
- Asymmetrische Schlüsselpaare: Die ersten 3 Byte des SHA-1-Hashs des öffentlichen Schlüssels.
- HMAC-Schlüssel: KCV für HMAC-Schlüssel wird derzeit nicht unterstützt.

Verwandte Themen

- Schlüssel
- Referenz für CloudHSM-CLI-Befehle

Migrieren von Client-SDK 3 CMU und KMU zu Client-SDK 5 CloudHSM CLI

Verwenden Sie dieses Thema, um Workflows zu migrieren, die die Befehlszeilen-Tools des Client-SDK 3, das CloudHSM Management Utility (CMU) und das Key Management Utility (KMU) verwenden, um stattdessen das Befehlszeilen-Tool des Client-SDK 5, CloudHSM CLI, zu verwenden.

In führen AWS CloudHSMKundenanwendungen kryptografische Operationen mit dem AWS CloudHSM Client Software Development Kit (SDK) aus. Client-SDK 5 ist das primäre SDK, dem weiterhin neue Funktionen und Plattformunterstützung hinzugefügt werden. Dieses Thema enthält Details zur Migration von Client-SDK 3 zu Client-SDK 5 für Befehlszeilen-Tools.

Client-SDK 3 enthält zwei separate Befehlszeilen-Tools: die CMU für die Verwaltung von Benutzern und die KMU für die Verwaltung von Schlüsseln und das Ausführen von Operationen mit Schlüsseln. Client-SDK 5 konsolidiert die Funktionen der CMU und KMU (Tools, die mit Client-SDK 3 angeboten wurden) in einem einzigen Tool, dem CloudHSM-Befehlszeilenschnittstelle (CLI). Benutzerverwaltungsvorgänge finden Sie unter den Unterbefehlen user und quorum. Schlüsselverwaltungsoperationen finden Sie unter dem Unterbefehl Schlüssel und kryptografische Operationen unter dem Unterbefehl Crypto . Eine vollständige Liste der Befehle Referenz für CloudHSM-CLI-Befehle finden Sie unter .

> **Note**
>
> Wenn Sie sich in Client-SDK 3 auf die syncUser Funktionen syncKey und für die Cluster-übergreifende Synchronisation verlassen haben, verwenden Sie weiterhin die CMU. CloudHSM-CLI in Client-SDK 5 unterstützt diese Funktionalität derzeit nicht.

Anweisungen zur Migration zum Client-SDK 5 finden Sie unter Migration von Client-SDK 3 zu Client-SDK 5. Vorteile der Migration finden Sie unter Vorteile von Client-SDK 5.

Erweiterte Konfigurationen für CLI

Die AWS CloudHSM Befehlszeilenschnittstelle (CLI) umfasst die folgende erweiterte Konfiguration, die nicht Teil der allgemeinen Konfigurationen ist, die die meisten Kunden verwenden. Diese Konfigurationen bieten zusätzliche Funktionen.

- Verbindung zu mehreren Clustern herstellen

Mit CLI eine Verbindung zu mehreren Clustern herstellen

Mit Client SDK 5 können Sie AWS CloudHSM CLI so konfigurieren, dass Verbindungen zu mehreren CloudHSM-Clustern von einer einzigen CLI-Instanz aus möglich sind.

Verwenden Sie die Anweisungen in diesem Thema, um mithilfe der AWS CloudHSM Befehlszeilenschnittstelle (CLI) die Multi-Cluster-Funktionalität zu verwenden, um eine Verbindung mit mehreren Clustern herzustellen.

Themen

- Voraussetzungen für mehrere Cluster
- CLI für Multi-Cluster-Funktionalität konfigurieren
- configure-cli Add-Cluster
- configure-cli remove-cluster
- Verwenden mehrerer Cluster

Voraussetzungen für mehrere Cluster

- Zwei oder mehr AWS CloudHSM Cluster, zu denen Sie eine Verbindung herstellen möchten, zusammen mit ihren Cluster-Zertifikaten.
- Eine EC2-Instance mit Sicherheitsgruppen, die korrekt konfiguriert sind, um eine Verbindung zu allen oben genannten Clustern herzustellen. Weitere Informationen zum Einrichten eines Clusters und der Client-Instanz finden Sie unter Erste Schritte mit AWS CloudHSM.
- Um die Multi-Cluster-Funktionalität einzurichten, müssen Sie die AWS CloudHSM CLI bereits heruntergeladen und installiert haben. Wenn Sie dies noch nicht getan haben, lesen Sie die Anweisungen unter ???.

- Sie können nicht auf einen Cluster zugreifen, der mit konfiguriert ist, `./configure-cli[.exe] -a` da er nicht mit einem `cluster-id` verknüpft ist. Sie können ihn neu konfigurieren, indem Sie `config-cli add-cluster` wie in diesem Handbuch beschrieben vorgehen.

CLI für Multi-Cluster-Funktionalität konfigurieren

Gehen Sie folgendermaßen vor, um Ihre AWS CloudHSM CLI für Multi-Cluster-Funktionalität zu konfigurieren:

1. Identifizieren Sie die Cluster, zu denen Sie eine Verbindung herstellen möchten.
2. Fügen Sie diese Cluster mit dem Unterbefehl AWS CloudHSM configure-cli `add-cluster` wie unten beschrieben zu Ihrer CLI-Konfiguration hinzu.
3. Starten Sie alle AWS CloudHSM CLI-Prozesse neu, damit die neue Konfiguration wirksam wird.

configure-cli Add-Cluster

Wenn Sie eine Verbindung zu mehreren Clustern herstellen, verwenden Sie den `configure-cli add-cluster` Befehl, um Ihrer Konfiguration einen Cluster hinzuzufügen.

Syntax

```
configure-cli add-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [--region <REGION>]
        [--endpoint <ENDPOINT>]
        [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
        [--server-client-cert-file <CLIENT CERTIFICATE FILE>]
        [--server-client-key-file <CLIENT KEY FILE>]
        [-h, --help]
```

Beispiele

Fügen Sie mithilfe des **`cluster-id`**-Parameters einen Cluster hinzu

Example

Verwenden Sie den `configure-cli add-cluster` zusammen mit dem `cluster-id`-Parameter, um Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) hinzuzufügen.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-cli.exe add-cluster --cluster-
id cluster-1234567
```

> **Tip**
>
> Wenn die Verwendung von `configure-cli add-cluster` mit dem `cluster-id`-Parameter nicht dazu führt, dass der Cluster hinzugefügt wird, finden Sie im folgenden Beispiel eine längere Version dieses Befehls, die auch die Parameter `--region` und `--endpoint` zur Identifizierung des hinzugefügten Clusters erfordert. Wenn zum Beispiel die Region des Clusters eine andere ist als die, die als Standard für Ihre AWS CLI konfiguriert ist, sollten Sie den `--region`-Parameter verwenden, um die richtige Region zu verwenden. Darüber hinaus haben Sie die Möglichkeit, den AWS CloudHSM API-Endpunkt anzugeben, der für den Anruf verwendet werden soll. Dies kann für verschiedene Netzwerkkonfigurationen erforderlich sein, z. B. für die Verwendung von VPC-Schnittstellenendpunkten, für die nicht den Standard-DNS-Hostnamen verwendet wird. AWS CloudHSM

Fügen Sie einen Cluster mit den Parametern **`cluster-id`**, **`endpoint`**, und **`region`** hinzu

Example

Verwenden Sie `configure-cli add-cluster` zusammen mit den Parametern `cluster-id`, `endpoint` und `region`, um Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) hinzuzufügen.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id cluster-1234567 --
region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-cli.exe add-cluster --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

Weitere Hinweise zu den Parametern `--cluster-id`, `--region` und `--endpoint` finden Sie unter the section called “Parameter”.

Parameter

--cluster-id ***<Cluster ID>***

Führt einen `DescribeClusters`-Aufruf aus, um alle IP-Adressen der HSM-Elastic-Network-Schnittstelle (ENI) im Cluster mit der Cluster-ID zu finden. Das System fügt die ENI-IP-Adressen zu den Konfigurationsdateien hinzu. AWS CloudHSM

> **Note**
>
> Wenn Sie den `--cluster-id` Parameter von einer EC2-Instance innerhalb einer VPC verwenden, die keinen Zugriff auf das öffentliche Internet hat, müssen Sie einen VPC-Schnittstellen-Endpunkt erstellen, mit dem Sie eine Verbindung herstellen können. AWS CloudHSM Weitere Informationen über VPC-Endpunkte finden Sie unter ???.

Erforderlich: Ja

--endpoint ***<Endpoint>***

Geben Sie den AWS CloudHSM API-Endpunkt an, der für den Aufruf verwendet wird. `DescribeClusters` Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Erforderlich: Nein

-- hsm-ca-cert <HsmCA Certificate Filepath>

Gibt den Dateipfad zum HSM-CA-Zertifikat an.

Erforderlich: Nein

--region *<Region>*

Geben Sie die Region Ihres Clusters an. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Wenn Sie den `--region`-Parameter nicht angeben, wählt das System die Region aus, indem es versucht, die Umgebungsvariablen `AWS_DEFAULT_REGION` oder `AWS_REGION` zu lesen. Wenn diese Variablen nicht festgelegt sind, überprüft das System die Region, die Ihrem Profil in Ihrer AWS-Config-Datei zugeordnet ist (normalerweise `~/.aws/config`), sofern Sie in der `AWS_CONFIG_FILE`-Umgebungsvariable keine andere Datei angegeben haben. Wenn keine der oben genannten Optionen festgelegt ist, verwendet das System standardmäßig die `us-east-1`-Region.

Erforderlich: Nein

-- server-client-cert-file <Client Certificate Filepath>

Pfad zum Client-Zertifikat, das für die gegenseitige TLS-Client-Server-Authentifizierung verwendet wird.

Verwenden Sie diese Option nur, wenn Sie nicht den Standardschlüssel und das SSL/TLS-Zertifikat verwenden möchten, die im Client-SDK 5 enthalten sind. Sie müssen diese Option in Kombination mit `--server-client-key-file` festlegen.

Erforderlich: Nein

-- server-client-key-file <Client Key Filepath>

Pfad zum Client-Schlüssel, der für die gegenseitige TLS-Client-Server-Authentifizierung verwendet wird.

Verwenden Sie diese Option nur, wenn Sie nicht den Standardschlüssel und das SSL/TLS-Zertifikat verwenden möchten, die im Client-SDK 5 enthalten sind. Sie müssen diese Option in Kombination mit `--server-client-cert-file` festlegen.

Erforderlich: Nein

configure-cli remove-cluster

Wenn Sie mit CLI eine Verbindung zu mehreren Clustern herstellen, verwenden Sie den `configure-cli remove-cluster` Befehl, um einen Cluster aus Ihrer Konfiguration zu entfernen.

Syntax

```
configure-cli remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

Beispiele

Entfernen Sie mithilfe des **`cluster-id`**-Parameters einen Cluster

Example

Verwenden Sie den `configure-cli remove-cluster` zusammen mit dem `cluster-id`-Parameter, um aus Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) zu entfernen.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli remove-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-cli.exe remove-cluster --cluster-
id cluster-1234567
```

Weitere Informationen zum Parameter `--cluster-id` erhalten Sie unter the section called “Parameter”.

Parameter

--cluster-id ***<Cluster ID>***

Die ID des Clusters, der aus der Konfiguration entfernt werden soll.

Erforderlich: Ja

Verwenden mehrerer Cluster

Nachdem Sie mehrere Cluster mit AWS CloudHSM CLI konfiguriert haben, verwenden Sie den `cloudhsm-cli` Befehl, um mit ihnen zu interagieren.

Beispiele

Einstellung einer Standardeinstellung **`cluster-id`** bei Verwendung des interaktiven Modus

Example

Verwenden Sie den ??? zusammen mit dem `cluster-id` Parameter, um einen Standardcluster (mit der ID von`cluster-1234567`) aus Ihrer Konfiguration festzulegen.

Linux

```
$ cloudhsm-cli interactive --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\cloudhsm-cli.exe interactive --cluster-
id cluster-1234567
```

Einstellung von**`cluster-id`**, wenn ein einzelner Befehl ausgeführt wird

Example

Verwenden Sie den `cluster-id` Parameter, um den Cluster (mit der ID von`cluster-1234567`) festzulegen, von ??? dem aus Sie abrufen möchten.

Linux

```
$ cloudhsm-cli cluster hsm-info --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\cloudhsm-cli.exe cluster hsm-info --cluster-
id cluster-1234567
```

Referenz für CloudHSM-CLI-Befehle

CloudHSM CLI unterstützt Administratoren bei der Verwaltung von Benutzern in ihrem Cluster. AWS CloudHSM CloudHSM-CLI kann in zwei Modi ausgeführt werden: im interaktiven Modus und im

Einzelbefehlsmodus. Informationen zum schnellen Einstieg finden Sie unter Erste Schritte mit der CloudHSM-Befehlszeilenschnittstelle (CLI).

Um die meisten CloudHSM-CLI-Befehle auszuführen, müssen Sie die CloudHSM-CLI starten und sich beim HSM anmelden. Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CloudHSM-CLI. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Die folgenden Themen beschreiben Befehle in CloudHSM-CLI.

Befehl	Beschreibung	Benutzertyp
cluster activate	Aktiviert ein CloudHSM-Cluster und bestätigt, dass das Cluster neu ist. Dies muss geschehen, bevor andere Operationen ausgeführt werden können.	Nicht aktivierter Admin
cluster hsm-info	Listet die HSMs in Ihrem Cluster auf.	Alle [1], auch nicht authentifizierte Benutzer. Eine Anmeldung ist nicht erforderlich.
Kryptozeichen ecdsa	Generiert eine Signatur mithilfe eines privaten EC-Schlüssels und des ECDSA-Signaturmechanismus.	Crypto-Benutzer (Crypto User, CU)
Kryptozeichen rsa-pkcs	Generiert eine Signatur unter Verwendung eines privaten RSA-Schlüssels und des RSA-PKCS-Signaturmechanismus.	CU
Krypto-Zeichen rsa-pkcs-pss	Generiert eine Signatur unter Verwendung eines privaten RSA-Schlüssels und des RSA-	CU

Befehl	Beschreibung	Benutzertyp
	PKCS-PSS-Signaturmechanismus.	
ecdsa kryptoverifizieren	Bestätigt, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde. Überprüft, ob die Signatur mithilfe des ECDSA-Signaturmechanismus generiert wurde. Vergleicht eine signierte Datei mit einer Quelldatei und ermittelt anhand eines bestimmten öffentlichen ECDSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.	CU
kryptoverifizieren rsa-pkcs	Bestätigt, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde. Überprüft, ob die Signatur mithilfe des RSA-PKCS-Signaturmechanismus generiert wurde. Vergleicht eine signierte Datei mit einer Quelldatei und bestimmt anhand eines bestimmten öffentlichen RSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.	CU

Befehl	Beschreibung	Benutzertyp
kryptoverifizieren rsa-pkcs-pss	Bestätigt, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde. Überprüft, ob die Signatur mithilfe des RSA-PKCS-PSS-Signaturmechanismus generiert wurde. Vergleicht eine signierte Datei mit einer Quelldatei und bestimmt anhand eines bestimmten öffentlichen RSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.	CU
key delete	Löscht einen Schlüssel aus Ihrem Cluster. AWS CloudHSM	CU
key generate-file	Generiert eine Schlüsseldatei in Ihrem AWS CloudHSM Cluster.	CU
Schlüssel generate-asymmetric-pair rsa	Generiert ein asymmetrisches RSA-Schlüsselpaar in Ihrem AWS CloudHSM Cluster.	CU
Schlüssel ec generate-asymmetric-pair	Generiert ein asymmetrisches key pair mit elliptischen Kurven (EC) in Ihrem Cluster. AWS CloudHSM	CU
key generate-symmetric aes	Generiert einen symmetrischen AES-Schlüssel in Ihrem Cluster. AWS CloudHSM	CU

Befehl	Beschreibung	Benutzertyp
key generate-symmetric generic-secret	Generiert einen symmetrischen Generic Secret-Schlüssel in Ihrem AWS CloudHSM Cluster.	CU
Schlüssel importieren (PEM)	Importiert einen Schlüssel im PEM-Format in ein HSM. Sie können ihn verwenden, um öffentliche und außerhalb des HSM erstellte Schlüssel zu importieren.	CU
key list	Findet alle Schlüssel für den aktuellen Benutzer, der in Ihrem AWS CloudHSM Cluster vorhanden ist.	CU
Schlüssel replizieren	Replizieren Sie einen Schlüssel von einem Quellcluster auf einen geklonten Zielcluster.	CU
key set-attribute	Legt die Attribute der Schlüssel in Ihrem AWS CloudHSM Cluster fest.	CUs können diesen Befehl ausführen, Admins können das vertrauenswürdige Attribut festlegen.
key share	Teilt einen Schlüssel mit anderen CUs in Ihrem AWS CloudHSM Cluster.	CU
key unshare	Macht die gemeinsame Nutzung eines Schlüssels mit anderen CUs in Ihrem AWS CloudHSM Cluster rückgängig.	CU

Befehl	Beschreibung	Benutzertyp
Schlüssel auspacken aes-gcm	Entpackt mithilfe des AES-Wrapping-Schlüssels und des AES-GCM-Entpackungsmechanismus einen Payload-Schlüssel in den Cluster.	CU
Schlüssel auspacken aes-no-pad	Entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des AES-NO-PAD-Entpackungsmechanismus in den Cluster.	CU
Schlüssel entpacken aes-pkcs5-Pad	Entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des AES-Wrapping-Mechanismus (AES-PKCS5-PAD).	CU
Schlüssel auspacken aes-zero-pad	Entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des AES-ZERO-PAD-Entpackungsmechanismus in den Cluster.	CU
Schlüssel auspacken cloudhsm-aes-gcm	Entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des CLOUDHSM-AES-GCM-Entpackungsmechanismus in den Cluster.	CU

Befehl	Beschreibung	Benutzertyp
Schlüssel zum Auspacken von RSA-AES	Entpackt einen Payload-Schlüssel mithilfe eines privaten RSA-Schlüssels und des RSA-AES-Entpackungsmechanismus.	CU
key unwrap rsa-oaep	Entpackt einen Payload-Schlüssel mithilfe des privaten RSA-Schlüssels und des RSA-OAEP-Entpackungsmechanismus.	CU
Schlüssel unwrap rsa-pkcs	Entpackt einen Payload-Schlüssel mithilfe des privaten RSA-Schlüssels und des RSA-PKCS-Entpackungsmechanismus.	CU
Schlüsselhülle aes-gcm	Wrappt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des AES-GCM-Wrapping-Mechanismus ein.	CU
Schlüsselhülle aes-no-pad	Umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des AES-NO-PAD-Wrapping-Mechanismus.	CU
Schlüsselhülle aes-pkcs5-Pad	Umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des AES-PKCS5-PAD-Umschließungsmechanismus.	CU

Befehl	Beschreibung	Benutzertyp
Schlüsselhülle aes-zero-pad	Wrappt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des AES-ZERO-PAD-Wickelmechanismus ein.	CU
Schlüsselhülle cloudhsm-aes-gcm	Umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des CLOUDHSM-AES-GCM-Wrapping-Mechanismus.	CUs
Schlüsselumschlag rsa-aes	Umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und dem RSA-AES-Wrapping-Mechanismus.	CU
Schlüsselhülle rsa-oaep	Umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des RSA-OAEP-Wrapping-Mechanismus.	CU

Befehl	Beschreibung	Benutzertyp
Der key wrap rsa-pkcs Befehl umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des Wrapping-Mechanismus. `RSA-PKCS` Das `extractable` Attribut des Payload-Schlüssels muss auf gesetzt sein. `true` Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden. Um den key wrap rsa-pkcs Befehl verwenden zu können, benötigen Sie zunächst einen RSA-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können ein RSA-Schlüsselpaar generieren, indem Sie den key generate-asymmetric-pair Befehl verwenden und das `wrap` Attribut auf setzen. `true` Benutzertyp Die folgenden Benutzertypen können diesen Befehl ausführen. • Crypto-Benutzer (CUs)	Umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des RSA-PKCS-Wrapping-Mechanismus.	CU

Befehl	Beschreibung	Benutzertyp
login	Melden AWS CloudHSM Sie sich bei Ihrem Cluster an.	Administrator, Crypto-Benutzer (CU) und Appliance-Benutzer (AU)
logout	Melden Sie sich von Ihrem AWS CloudHSM Cluster ab.	Administrator, CU und Appliance-Benutzer (AU)
quorum token-sign delete	Löscht ein oder mehrere Token für einen vom Quorum autorisierten Dienst.	Admin.
quorum token-sign generate	Generiert ein Token für einen vom Quorum autorisierten Dienst.	Admin.
quorum token-sign list	Listet alle token-sign-Quorum-Token auf, die in Ihrem CloudHSM-Cluster vorhanden sind.	Alle [1], auch nicht authentifizierte Benutzer. Eine Anmeldung ist nicht erforderlich.
Quorum-Tokenzeichen list-quorum-values	Listet die in Ihrem CloudHSM-Cluster festgelegten Quorumwerte auf.	Alle [1], auch nicht authentifizierte Benutzer. Eine Anmeldung ist nicht erforderlich.
quorum token-sign list-timeouts	Ruft den Token-Timeout-Zeitraum in Sekunden für alle Tokentypen ab.	Admin und Crypto-Benutzer
Quorum-Tokenzeichen set-quorum-value	Legt einen neuen Quorumwert für einen vom Quorum autorisierten Dienst fest.	Admin.
quorum token-sign set-timeout	Legt den Token-Timeout-Zeitraum in Sekunden für jeden Tokentyp fest.	Admin.

Befehl	Beschreibung	Benutzertyp
user change-mfa	Ändert die Strategie zur Multi-Faktor-Authentifizierung (MFA) eines Benutzers.	Admin, CU
user change-password	Ändert die Passwörter von Benutzern der HSMs. Jeder Benutzer kann das eigene Passwort ändern. Admins können das Passwort jedes Benutzers ändern.	Admin, CU
user create	Erzeugt einen Benutzer in Ihrem Cluster. AWS CloudHSM	Admin.
user delete	Löscht einen Benutzer in Ihrem AWS CloudHSM Cluster.	Admin.
user list	Listet die Benutzer in Ihrem AWS CloudHSM Cluster auf.	Alle [1], auch nicht authentifizierte Benutzer. Eine Anmeldung ist nicht erforderlich.
user change-quorum token-sign register	Registriert die Quorumstrategie mit Quorum-token-sign für einen Benutzer.	Admin.

Anmerkungen

- [1] Alle Benutzer umfasst alle aufgelisteten Rollen und Benutzer, die nicht angemeldet sind.

Cluster

cluster ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen benutzerspezifischen Befehl erzeugen. Derzeit besteht die Benutzerkategorie aus den folgenden Befehlen:

- cluster activate
- cluster hsm-info

cluster activate

Verwenden Sie den cluster activate-Befehl in der CloudHSM-CLI, um ein neues Cluster zu aktivieren. Dieser Befehl muss ausgeführt werden, bevor der Cluster zum Ausführen von kryptografischen Operationen verwendet werden kann.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Nicht aktivierter Admin

Syntax

Dieser Befehl hat keine Parameter.

```
aws-cloudhsm > help cluster activate
Activate a cluster

This command will set the initial Admin password. This process will cause your CloudHSM
 cluster to
move into the ACTIVE state.

USAGE:
    cloudhsm-cli cluster activate [OPTIONS] [--password <PASSWORD>]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
      --password <PASSWORD>
          Optional: Plaintext activation password If you do not include this argument
 you will be prompted for it

  -h, --help
          Print help (see a summary with '-h')
```

Beispiel

Dieser Befehl aktiviert Ihr Cluster, indem er das anfängliche Passwort für Ihren Admin-Benutzer festlegt.

```
aws-cloudhsm > cluster activate
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": "Cluster activation successful"
}
```

Verwandte Themen

- user create
- user delete
- user change-password

cluster hsm-info

Verwenden Sie den cluster hsm-info-Befehl in der CloudHSM-CLI, um die HSMs in Ihrem Cluster aufzulisten. Sie müssen nicht bei CloudHSM-CLI angemeldet sein, um diesen Befehl auszuführen.

> **Note**
>
> Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien, die der AWS CloudHSM Client und die Befehlszeilentools verwenden. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

```
aws-cloudhsm > help cluster hsm-info
List info about each HSM in the cluster

Usage: cloudhsm-cli cluster hsm-info [OPTIONS]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

Beispiel

Dieser Befehl listet die in Ihrem AWS CloudHSM Cluster vorhandenen HSMs auf.

```
aws-cloudhsm > cluster hsm-info
{
  "error_code": 0,
  "data": {
    "hsms": [
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000590",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      },
      {
        "vendor": "Marvell Semiconductors, Inc.",
```

```
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000625",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      },
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000663",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      }
    ]
  }
}
```

Diese Ausgabe hat die folgenden Attribute:

- Anbieter: Der Name des Anbieters des HSM.
- Modell: Die Modellnummer des HSM.
- Seriennummer: Die Seriennummer des HSM. Dies kann sich aufgrund von Ersetzungen ändern.
- H ardware-version-major: Die wichtigste Hardwareversion.
- H ardware-version-minor: Die kleinere Hardwareversion.
- F irmware-version-major: Die Haupt-Firmware-Version.
- F irmware-version-minor: Die kleinere Firmware-Version.
- F irmware-build-number: Die Build-Nummer der Firmware.
- Firmware-id: Die Firmware-ID, die die Haupt- und Nebenversionen zusammen mit dem Build enthält.

Verwandte Themen

- cluster activate

crypto

crypto ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erstellen, der für kryptografische Operationen spezifisch ist. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- Crypto-Zeichen
  - Kryptozeichen ecdsa
  - Kryptozeichen rsa-pkcs
  - Krypto-Zeichen rsa-pkcs-pss
- Crypto-Verifizierung
  - ecdsa kryptoverifizieren
  - kryptoverifizieren rsa-pkcs
  - kryptoverifizieren rsa-pkcs-pss

Crypto-Zeichen

crypto sign ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen ausgewählten privaten Schlüssel in Ihrem AWS CloudHSM Cluster verwendet, um eine Signatur zu generieren. crypto sign hat die folgenden Unterbefehle:

- Kryptozeichen ecdsa
- Kryptozeichen rsa-pkcs
- Krypto-Zeichen rsa-pkcs-pss

Um verwenden zu könnencrypto sign, benötigen Sie einen privaten Schlüssel in Ihrem HSM. Sie können einen privaten Schlüssel mit den folgenden Befehlen generieren:

- Schlüssel- generate-asymmetric-pair EC
- Schlüssel- generate-asymmetric-pair RSA

Kryptozeichen ecdsa

Der crypto sign ecdsa Befehl generiert eine Signatur unter Verwendung eines privaten EC-Schlüssels und des ECDSA-Signaturmechanismus.

Um den crypto sign ecdsa Befehl verwenden zu können, benötigen Sie zunächst einen privaten EC-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können mit dem Schlüssel generate-asymmetric-pair ec Befehl, dessen `sign` Attribut auf gesetzt ist, einen privaten EC-Schlüssel generieren`true`.

> **Note**
>
> Signaturen können AWS CloudHSM mit Crypto-Verifizierung Unterbefehlen verifiziert werden.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help crypto sign ecdsa
Sign with the ECDSA mechanism

Usage: crypto sign ecdsa --key-filter [<KEY_FILTER>>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
```

```
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be signed
      --data <DATA>
          Base64 Encoded data to be signed
  -h, --help
          Print help
```

Beispiel

Diese Beispiele zeigen, wie eine Signatur mithilfe des ECDSA-Signaturmechanismus und SHA256 der Hash-Funktion generiert wird. crypto sign ecdsa Dieser Befehl verwendet einen privaten Schlüssel im HSM.

Example Beispiel: Generieren Sie eine Signatur für Base-64-kodierte Daten

```
aws-cloudhsm > crypto sign ecdsa --key-filter attr.label=ec-private --hash-function
 sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

Example Beispiel: Generieren Sie eine Signatur für eine Datendatei

```
aws-cloudhsm > crypto sign ecdsa --key-filter attr.label=ec-private --hash-function
 sha256 --data-path data.txt
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B. key-reference=0xabc) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE, um einen passenden Schlüssel auszuwählen.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

Verwandte Themen

- Crypto-Zeichen
- Crypto-Verifizierung

Kryptozeichen rsa-pkcs

Der crypto sign rsa-pkcs Befehl generiert eine Signatur mithilfe eines privaten RSA-Schlüssels und des RSA-PKCS-Signaturmechanismus.

Um den crypto sign rsa-pkcs Befehl verwenden zu können, benötigen Sie zunächst einen privaten RSA-Schlüssel in Ihrem Cluster. AWS CloudHSM Sie können mit dem Schlüssel generate-asymmetric-pair rsa Befehl, dessen `sign` Attribut auf gesetzt ist, einen privaten RSA-Schlüssel generieren. `true`

> **Note**
>
> Signaturen können AWS CloudHSM mit Crypto-Verifizierung Unterbefehlen verifiziert werden.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help crypto sign rsa-pkcs
Sign with the RSA-PKCS mechanism

Usage: crypto sign rsa-pkcs --key-filter [<KEY_FILTER>>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
```

```
        Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
     --key-filter [<KEY_FILTER>...]
         Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
     --hash-function <HASH_FUNCTION>
         [possible values: sha1, sha224, sha256, sha384, sha512]
     --data-path <DATA_PATH>
         The path to the file containing the data to be signed
     --data <DATA>
         Base64 Encoded data to be signed
  -h, --help
         Print help
```

Beispiel

Diese Beispiele zeigen, wie eine Signatur mithilfe des RSA-PKCS-Signaturmechanismus und der Hash-Funktion generiert wird. crypto sign rsa-pkcs SHA256 Dieser Befehl verwendet einen privaten Schlüssel im HSM.

Example Beispiel: Generieren Sie eine Signatur für Base-64-kodierte Daten

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function
 sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Example Beispiel: Generieren Sie eine Signatur für eine Datendatei

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function
 sha256 --data-path data.txt
```

```
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht in Form von Daten angegeben)

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384

- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

Verwandte Themen

- Crypto-Zeichen
- Crypto-Verifizierung

Krypto-Zeichen rsa-pkcs-pss

Der crypto sign rsa-pkcs-pss Befehl generiert eine Signatur mithilfe eines privaten RSA-Schlüssels und des `RSA-PKCS-PSS` Signaturmechanismus.

Um den crypto sign rsa-pkcs-pss Befehl verwenden zu können, müssen Sie zunächst über einen privaten RSA-Schlüssel in Ihrem AWS CloudHSM Cluster verfügen. Sie können mit dem Schlüssel generate-asymmetric-pair rsa Befehl, dessen `sign` Attribut auf gesetzt ist, einen privaten RSA-Schlüssel generieren. `true`

> **Note**
>
> Signaturen können AWS CloudHSM mit Crypto-Verifizierung Unterbefehlen verifiziert werden.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help crypto sign rsa-pkcs-pss
Sign with the RSA-PKCS-PSS mechanism

Usage: crypto sign rsa-pkcs-pss [OPTIONS] --key-filter [<KEY_FILTER>...] --
hash-function <HASH_FUNCTION> --mgf <MGF> --salt-length <SALT_LENGTH> <--data-
path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>        Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]   Key reference (e.g. key-
reference=0xabc) or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key
      --hash-function <HASH_FUNCTION>  [possible values: sha1, sha224, sha256, sha384,
 sha512]
      --data-path <DATA_PATH>          The path to the file containing the data to be
 signed
      --data <DATA>                    Base64 Encoded data to be signed
      --mgf <MGF>                      The mask generation function [possible values:
 mgf1-sha1, mgf1-sha224, mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --salt-length <SALT_LENGTH>      The salt length
  -h, --help                           Print help
```

Beispiel

Diese Beispiele zeigen, wie eine Signatur mithilfe des RSA-PKCS-PSS Signaturmechanismus und der SHA256 Hash-Funktion generiert wird. crypto sign rsa-pkcs-pss Dieser Befehl verwendet einen privaten Schlüssel im HSM.

Example Beispiel: Generieren Sie eine Signatur für Base-64-kodierte Daten

```
aws-cloudhsm > crypto sign rsa-pkcs-pss --key-filter attr.label=rsa-private --hash-
function sha256 --data YWJjMTIz --salt-length 10 --mgf mgf1-sha256
{
  "error_code": 0,
```

```
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

Example Beispiel: Generieren Sie eine Signatur für eine Datendatei

```
aws-cloudhsm > crypto sign rsa-pkcs-pss --key-filter attr.label=rsa-private --hash-
function sha256 --data-path data.txt --salt-length 10 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht in Form von Daten angegeben)

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<MGF>

Definiert die Funktion zur Maskengenerierung.

> **Note**
>
> Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<SALT_LENGTH>

Gibt die Salzlänge an.

Erforderlich: Ja

Verwandte Themen

- Crypto-Zeichen
- Crypto-Verifizierung

Verwandte Themen

- Crypto-Verifizierung

Crypto-Verifizierung

crypto verify ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie bestätigt, ob eine Datei mit einem bestimmten Schlüssel signiert wurde. crypto verify hat die folgenden Unterbefehle:

- Crypto-Verifizierungs-ECDSA
- -Krypto-Verifizierung rsa-pkcs
- Crypto-Verifizierung rsa-pkcs-pss

Der crypto verify Befehl vergleicht eine signierte Datei mit einer Quelldatei und analysiert anhand eines bestimmten öffentlichen Schlüssels und Signaturmechanismus, ob sie kryptografisch verwandt sind.

> **Note**
>
> Dateien können in AWS CloudHSM mit der Operation Crypto-Zeichen signiert werden.

ecdsa kryptoverifizieren

Der crypto verify ecdsa Befehl wird verwendet, um die folgenden Operationen abzuschließen:

- Bestätigen Sie, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde.
- Stellen Sie sicher, dass die Signatur mithilfe des ECDSA-Signaturmechanismus generiert wurde.
- Vergleichen Sie eine signierte Datei mit einer Quelldatei und stellen Sie anhand eines bestimmten öffentlichen ECDSA-Schlüssels und Signaturmechanismus fest, ob die beiden kryptografisch verwandt sind.

Um den crypto verify ecdsa Befehl verwenden zu können, müssen Sie zunächst über einen öffentlichen EC-Schlüssel in Ihrem Cluster verfügen. AWS CloudHSM Sie können einen öffentlichen EC-Schlüssel importieren, indem Sie den Schlüsselimportstift Befehl verwenden, dessen `verify` Attribut auf gesetzt ist`true`.

> **Note**
>
> Sie können in der CloudHSM-CLI eine Signatur mit Crypto-Zeichen Unterbefehlen generieren.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help crypto verify ecdsa
Verify with the ECDSA mechanism

Usage: crypto verify ecdsa --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-
path <SIGNATURE_PATH>|--signature <SIGNATURE>>

Options:
```

```
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
  -h, --help
          Print help
```

Beispiel

Diese Beispiele zeigen, wie eine Signatur verifiziert wirdcrypto verify ecdsa, die mithilfe des ECDSA-Signaturmechanismus und der Hash-Funktion generiert wurde. SHA256 Dieser Befehl verwendet einen öffentlichen Schlüssel im HSM.

Example Beispiel: Überprüfen Sie eine Base64-codierte Signatur mit Base64-codierten Daten

```
aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --key-filter attr.label=ec-
public --data YWJjMTIz --signature 4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Beispiel: Verifizieren Sie eine Signaturdatei mit einer Datendatei

```
aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --key-filter attr.label=ec-
public --data-path data.txt --signature-path signature-file
```

```
{
   "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Beispiel: Beweisen Sie die Beziehung zwischen falschen Signaturen

Mit diesem Befehl wird überprüft, ob die unter befindlichen Daten mit einem öffentlichen Schlüssel mit der Bezeichnung signiert `/home/data` wurden. Dabei wird der ECDSA-Signaturmechanismus `ecdsa-public` verwendet, um die Signatur zu erzeugen, die sich in befindet. `/home/signature` Da die angegebenen Argumente keine echte Signaturbeziehung bilden, gibt der Befehl eine Fehlermeldung zurück.

```
aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --
key-filter attr.label=ec-public --data aW52YWxpZA== --signature
 +ogk7M7S3iTqFg3SndJfd91dZFr5Qo6YixJl8JwcvqqVgsVuO6o+VKvTRjz0/V05kf3JJbBLr87Q
+wLWcMAJfA==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<SIGNATURE>

Base64-codierte Signatur.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

<SIGNATURE_PATH>

Gibt den Ort der Signatur an.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

Verwandte Themen

- Crypto-Zeichen
- Crypto-Verifizierung

kryptoverifizieren rsa-pkcs

Der crypto verify rsa-pkcs Befehl wird verwendet, um die folgenden Operationen abzuschließen:

- Bestätigen Sie, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde.
- Stellen Sie sicher, dass die Signatur mithilfe des `RSA-PKCS` Signaturmechanismus generiert wurde.
- Vergleicht eine signierte Datei mit einer Quelldatei und ermittelt anhand eines bestimmten öffentlichen RSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.

Um den crypto verify rsa-pkcs Befehl verwenden zu können, müssen Sie zunächst über einen öffentlichen RSA-Schlüssel in Ihrem Cluster verfügen. AWS CloudHSM

> **Note**
>
> Sie können mithilfe der CloudHSM-CLI mit den Unterbefehlen eine Signatur generieren. Crypto-Zeichen

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help crypto verify rsa-pkcs
Verify with the RSA-PKCS mechanism

Usage: crypto verify rsa-pkcs --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-
path <SIGNATURE_PATH>|--signature <SIGNATURE>>
```

```
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
  -h, --help
          Print help
```

Beispiel

Diese Beispiele zeigen, wie eine Signatur verifiziert wirdcrypto verify rsa-pkcs, die mithilfe des RSA-PKCS-Signaturmechanismus und der Hash-Funktion generiert wurde. SHA256 Dieser Befehl verwendet einen öffentlichen Schlüssel im HSM.

Example Beispiel: Überprüfen Sie eine Base64-codierte Signatur mit Base64-codierten Daten

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
 attr.label=rsa-public --data YWJjMTIz --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
```

```
}
```

Example Beispiel: Verifizieren Sie eine Signaturdatei mit einer Datendatei

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
 attr.label=rsa-public --data-path data.txt --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Beispiel: Nachweis einer falschen Signaturbeziehung

Mit diesem Befehl wird überprüft, ob die ungültigen Daten mit einem öffentlichen Schlüssel mit der Bezeichnung signiert wurden. Dabei wird der RSAKCS-Signaturmechanismus `rsa-public` verwendet, um die Signatur zu erzeugen, die sich in befindet. `/home/signature` Da die angegebenen Argumente keine echte Signaturbeziehung bilden, gibt der Befehl eine Fehlermeldung zurück.

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
 attr.label=rsa-public --data aW52YWxpZA== --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<SIGNATURE>

Base64-codierte Signatur.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

<SIGNATURE_PATH>

Gibt den Ort der Signatur an.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

Verwandte Themen

- Crypto-Zeichen
- Crypto-Verifizierung

kryptoverifizieren rsa-pkcs-pss

Der crypto sign rsa-pkcs-pss Befehl wird verwendet, um die folgenden Operationen abzuschließen.

- Bestätigen Sie, dass eine Datei im HSM mit einem bestimmten öffentlichen Schlüssel signiert wurde.
- Stellen Sie sicher, dass die Signatur mithilfe des RSA-PKCS-PSS-Signaturmechanismus generiert wurde.
- Vergleicht eine signierte Datei mit einer Quelldatei und ermittelt anhand eines bestimmten öffentlichen RSA-Schlüssels und Signaturmechanismus, ob die beiden kryptografisch verwandt sind.

Um den crypto verify rsa-pkcs-pss Befehl verwenden zu können, müssen Sie zunächst über einen öffentlichen RSA-Schlüssel in Ihrem Cluster verfügen. AWS CloudHSM Sie können einen öffentlichen RSA-Schlüssel mit dem Befehl key import pem (ADD UNWRAP LINK HERE) importieren, wobei das `verify` Attribut auf gesetzt ist. `true`

> **Note**
>
> Sie können mithilfe der CloudHSM-CLI mit den Unterbefehlen eine Signatur generieren. Crypto-Zeichen

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help crypto verify rsa-pkcs-pss
Verify with the RSA-PKCS-PSS mechanism

Usage: crypto verify rsa-pkcs-pss --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --salt-length >SALT_LENGTH< <--data-
path <DATA_PATH>|--data <DATA> <--signature-path <SIGNATURE_PATH>|--
signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
      --mgf <MGF>
          The mask generation function [possible values: mgf1-sha1, mgf1-sha224, mgf1-
sha256, mgf1-sha384, mgf1-sha512]
      --salt-length <SALT_LENGTH>
          The salt length
  -h, --help
```

```
          Print help
```

Beispiel

Diese Beispiele zeigen, wie eine Signatur verifiziert wirdcrypto verify rsa-pkcs-pss, die mit dem RSA-PKCS-PSS-Signiermechanismus und der Hash-Funktion generiert wurde. SHA256 Dieser Befehl verwendet einen öffentlichen Schlüssel im HSM.

Example Beispiel: Überprüfen Sie eine Base64-codierte Signatur mit Base64-codierten Daten

```
aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public
 --hash-function sha256 --data YWJjMTIz --salt-length 10 --mgf mgf1-sha256
 --signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Beispiel: Verifizieren Sie eine Signaturdatei mit einer Datendatei

```
aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public --hash-
function sha256 --data-path data.txt --salt-length 10 --mgf mgf1-sha256 --signature
 signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Beispiel: Nachweis einer falschen Signaturbeziehung

Mit diesem Befehl wird überprüft, ob die ungültigen Daten mit einem öffentlichen Schlüssel mit der Bezeichnung signiert wurden. Dabei wird der RSAPKCSPSS-Signaturmechanismus `rsa-public` verwendet, um die Signatur zu erzeugen, die sich in befindet. `/home/signature` Da die

angegebenen Argumente keine echte Signaturbeziehung bilden, gibt der Befehl eine Fehlermeldung zurück.

```
aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public
 --hash-function sha256 --data aW52YWxpZA== --salt-length 10 --mgf mgf1-sha256
 --signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<DATA>

Base64-kodierte Daten, die signiert werden sollen.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<DATA_PATH>

Gibt den Speicherort der zu signierenden Daten an.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384

- sha512

Erforderlich: Ja

<KEY_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM CLI.

Erforderlich: Ja

<MFG>

Definiert die Funktion zur Maskengenerierung.

> **Note**
>
> Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<SIGNATURE>

Base64-codierte Signatur.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

<SIGNATURE_PATH>

Gibt den Ort der Signatur an.

Erforderlich: Ja (sofern nicht über den Signaturpfad angegeben)

Verwandte Themen

- Crypto-Zeichen
- Crypto-Verifizierung

Schlüssel

key ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen schlüsselspezifischen Befehl erzeugen. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- key delete
- key generate-file
- Schlüssel generate-asymmetric-pair
  - Schlüssel generate-asymmetric-pair RSA
  - Schlüssel usw. generate-asymmetric-pair
- key generate-symmetric
  - key generate-symmetric aes
  - key generate-symmetric generic-secret
- Schlüsselimportstift
- key list
- Schlüsselreplikat
- key set-attribute
- key share
- key unshare
- Schlüssel entpacken
- Schlüsselumbruch

key delete

Verwenden Sie den key delete Befehl in der CloudHSM-CLI, um einen Schlüssel aus einem AWS CloudHSM Cluster zu löschen. Sie können nur jeweils einen Schlüssel löschen. Das Löschen eines

Schlüssels in einem Schlüsselpaar hat keine Auswirkungen auf den anderen Schlüssel in diesem Paar.

Nur der CU, der den Schlüssel erstellt hat und somit Eigentümer des Schlüssels ist, kann den Schlüssel löschen. Benutzer, die den Schlüssel gemeinsam nutzen, ihn aber nicht besitzen, können den Schlüssel für kryptografische Operationen verwenden, ihn aber nicht löschen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key delete
Delete a key in the HSM cluster

Usage: key delete [OPTIONS] --filter [<FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --filter [<FILTER>...]     Key reference (e.g. key-reference=0xabc)
 or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key for deletion
  -h, --help                     Print help
```

Beispiel

```
aws-cloudhsm > key delete --filter attr.label="ec-test-public-key"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
```

```
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI

Erforderlich: Ja

Verwandte Themen

- key list
- key generate-file
- key unshare
- Schlüsselattribute für CloudHSM-CLI
- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln

key generate-file

Der key generate-file Befehl exportiert einen asymmetrischen Schlüssel aus dem HSM. Wenn das Ziel ein privater Schlüssel ist, wird der Verweis auf den privaten Schlüssel im gefälschten PEM-Format exportiert. Wenn das Ziel ein öffentlicher Schlüssel ist, werden die Bytes des öffentlichen Schlüssels im PEM-Format exportiert.

Die gefälschte PEM-Datei, die nicht das eigentliche Material des privaten Schlüssels enthält, sondern stattdessen auf den privaten Schlüssel im HSM verweist, kann verwendet werden, um das SSL-/TLS-

Offloading von Ihrem Webserver zu einzurichten. AWS CloudHSM Weitere Informationen finden Sie unter SSL/TLS-Offloading.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key generate-file
Generate a key file from a key in the HSM cluster. This command does not export any
 private key data from the HSM

Usage: key generate-file --encoding <ENCODING> --path <PATH> --filter [<FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --encoding <ENCODING>
          Encoding format for the key file

          Possible values:
          - reference-pem: PEM formatted key reference (supports private keys)
          - pem:           PEM format (supports public keys)

      --path <PATH>
          Filepath where the key file will be written

      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key for file generation

  -h, --help
          Print help (see a summary with '-h')
```

Beispiel

Dieses Beispiel zeigt, wie Sie damit eine Schlüsseldatei key generate-file in Ihrem Cluster generieren können. AWS CloudHSM

Example

```
aws-cloudhsm > key generate-file --encoding reference-pem --path /tmp/ec-private-
key.pem --filter attr.label="ec-test-private-key"
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<ENCODING>

Gibt das Kodierungsformat für die Schlüsseldatei an

Erforderlich: Ja

<PATH>

Gibt den Dateipfad an, in den die Schlüsseldatei geschrieben wird

Erforderlich: Ja

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI
- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln
- key generate-asymmetric-pair
- key generate-symmetric

key generate-asymmetric-pair

key generate-asymmetric-pair ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erzeugen, der asymmetrische Schlüsselpaare generiert. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- key generate-asymmetric-pair ec
- key generate-asymmetric-pair rsa

Schlüssel generate-asymmetric-pair ec

Verwenden Sie den key asymmetric-pair ec Befehl in der CloudHSM-CLI, um ein asymmetrisches EC-Schlüsselpaar (Elliptic Curve) in Ihrem Cluster zu generieren. AWS CloudHSM

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key generate-asymmetric-pair ec
Generate an Elliptic-Curve Cryptography (ECC) key pair

Usage: key generate-asymmetric-pair ec [OPTIONS] --public-label <PUBLIC_LABEL> --
private-label <PRIVATE_LABEL> --curve <CURVE>

Options:
```

```
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
operation against. If not provided, will fall back to the value provided when
interactive mode was started, or error
      --public-label <PUBLIC_LABEL>
          Label for the public key
      --private-label <PRIVATE_LABEL>
          Label for the private key
      --session
          Creates a session key pair that exists only in the current session. The key
cannot be recovered after the session ends
      --curve <CURVE>
          Elliptic curve used to generate the key pair [possible values: prime256v1,
secp256r1, secp224r1, secp384r1, secp256k1, secp521r1]
      --public-attributes [<PUBLIC_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated EC public key
in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --private-attributes [<PRIVATE_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated EC private
key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie Sie mit diesem key generate-asymmetric-pair ec-Befehl ein EC-Schlüsselpaar erstellen.

Example Beispiel: Ein EC-Schlüsselpaar erstellen

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp224r1 \
    --public-label ec-public-key-example \
    --private-label ec-private-key-example
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
```

```
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-public-key-example",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
        "curve": "secp224r1"
      }
    },
"private_key": {
      "key-reference": "0x000000000012000c",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
```

```
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-private-key-example",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
        "curve": "secp224r1"
      }
    }
  }
}
```

Example Beispiel: Erstellen Sie ein EC-Schlüsselpaar mit optionalen Attributen

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp224r1 \
    --public-label ec-public-key-example \
    --private-label ec-private-key-example \
    --public-attributes token=true encrypt=true \
```

```
    --private-attributes token=true decrypt=true
{
 "error_code": 0,
 "data": {
   "public_key": {
     "key-reference": "0x00000000002806eb",
     "key-info": {
       "key-owners": [
         {
           "username": "cu1",
           "key-coverage": "full"
         }
       ],
       "shared-users": [],
       "cluster-coverage": "full"
     },
     "attributes": {
       "key-type": "ec",
       "label": "ec-public-key-example",
       "id": "",
       "check-value": "0xedef86",
       "class": "public-key",
       "encrypt": true,
       "decrypt": false,
       "token": true,
       "always-sensitive": false,
       "derive": false,
       "destroyable": true,
       "extractable": true,
       "local": true,
       "modifiable": true,
       "never-extractable": false,
       "private": true,
       "sensitive": false,
       "sign": false,
       "trusted": false,
       "unwrap": false,
       "verify": false,
       "wrap": false,
       "wrap-with-trusted": false,
       "key-length-bytes": 57,
       "ec-point":
"0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
       "curve": "secp224r1"
```

```
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280c82",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-private-key-example",
        "id": "",
        "check-value": "0xedef86",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
"0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
        "curve": "secp224r1"
      }
    }
```

```
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<CURVE>

Gibt die ID für die elliptische Kurve an.

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

Erforderlich: Ja

<PUBLIC_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten öffentlichen EC-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`)

Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<PUBLIC_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an. Die maximal zulässige Größe für Client SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<PRIVATE_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten privaten EC-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`)

Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<PRIVATE_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den private-key an. Die maximal zulässige Größe für Client-SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von <SESSION> ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI
- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln

Schlüssel generate-asymmetric-pair rsa

Verwenden Sie den key generate-asymmetric-pair rsa Befehl, um ein asymmetrisches RSA-Schlüsselpaar in Ihrem AWS CloudHSM Cluster zu generieren.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key generate-asymmetric-pair rsa
Generate an RSA key pair

Usage: key generate-asymmetric-pair rsa [OPTIONS] --public-label <PUBLIC_LABEL>
 --private-label <PRIVATE_LABEL> --modulus-size-bits <MODULUS_SIZE_BITS> --public-
exponent <PUBLIC_EXPONENT>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --public-label <PUBLIC_LABEL>
          Label for the public key
      --private-label <PRIVATE_LABEL>
          Label for the private key
      --session
          Creates a session key pair that exists only in the current session. The key
 cannot be recovered after the session ends
      --modulus-size-bits <MODULUS_SIZE_BITS>
          Modulus size in bits used to generate the RSA key pair
      --public-exponent <PUBLIC_EXPONENT>
          Public exponent used to generate the RSA key pair
      --public-attributes [<PUBLIC_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated RSA public
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
```

```
      --private-attributes [<PRIVATE_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated RSA private
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
  -h, --help
          Print help
```

Beispiele

Diese Beispiele verdeutlichen, wie mit `key generate-asymmetric-pair rsa` ein RSA-Schlüsselpaar erstellt wird.

Example Beispiel: Ein RSA-Schlüsselpaar erstellen

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000160010",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
        "check-value": "0x498e1f",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
```

```
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
e89a065e7d1a46ced96b46b909db2ab6be871ee700fd0a448b6e975bb64cae77c49008749212463e37a577baa57ce3e
bcebb7d20bd6df1948ae336ae23b52d73b7f3b6acc2543edb6358e08d326d280ce489571f4d34e316a2ea1904d513ca
        "modulus-size-bits": 2048
      }
    },
"private_key": {
      "key-reference": "0x0000000000160011",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-private-key-example",
        "id": "",
        "check-value": "0x498e1f",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
```

```
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
 "0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
        "modulus-size-bits": 2048
      }
    }
  }
}
```

Example Beispiel: Erstellen eines RSA-Schlüsselpaars mit optionalen Attributen

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes token=true encrypt=true \
--private-attributes token=true decrypt=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000280cc8",
      "key-info": {
        "key-owners": [
```

```
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "public-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
73a80fdb457aa7b20cd61e486c326e2cfd5e124a7f6a996437437812b542e3caf85928aa866f0298580f7967ee6aa01
f6e6296d6c116d5744c6d60d14d3bf3cb978fe6b75ac67b7089bafd50d8687213b31abc7dc1bad422780d29c851d510
133022653225bd129f8491101725e9ea33e1ded83fb57af35f847e532eb30cd7e726f23910d2671c6364092e834697e
ac3160f0ca9725d38318b7",
        "modulus-size-bits": 2048
      }
    },
    "private_key": {
```

```
      "key-reference": "0x0000000000280cc7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-private-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
        "modulus-size-bits": 2048
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<MODULUS_SIZE_BITS>

Gibt die Länge des Moduls in Bits an. Der minimale Wert beträgt 2048.

Erforderlich: Ja

<PRIVATE_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten privaten RSA-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`)

Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<PRIVATE_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den private-key an. Die maximal zulässige Größe für Client SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<PUBLIC_EXPONENT>

Gibt den öffentlichen Exponenten an. Bei diesem Wert muss es sich eine ungerade Zahl gleich oder größer als 65537 handeln.

Erforderlich: Ja

<PUBLIC_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten öffentlichen RSA-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`)

Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<PUBLIC_LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an. Die maximal zulässige Größe für Client-SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von <SESSION> ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI
- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln

key generate-symmetric

key generate-symmetric ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl erzeugen, der symmetrische Schlüssel generiert. Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- key generate-symmetric aes
- key generate-symmetric generic-secret

key generate-symmetric aes

Der key generate-symmetric aes Befehl generiert einen symmetrischen AES-Schlüssel in Ihrem AWS CloudHSM Cluster.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key generate-symmetric aes
Generate an AES key

Usage: key generate-symmetric aes [OPTIONS] --label <LABEL> --key-length-
bytes <KEY_LENGTH_BYTES>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --label <LABEL>
          Label for the key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
```

```
      --key-length-bytes <KEY_LENGTH_BYTES>
          Key length in bytes
      --attributes [<KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated AES key in
 the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie Sie den key generate-symmetric aes-Befehl zum Erstellen eines AES-Schlüssels verwenden.

Example Beispiel: Einen AES-Schlüssel erstellen

```
aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24
 {
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-aes",
        "id": "",
        "check-value": "0x9b94bd",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
```

```
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 24
      }
    }
  }
}
```

Example Beispiel: Erstellen eines AES-Schlüssels mit optionalen Attributen

```
aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24 \
--attributes decrypt=true encrypt=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "aes",
```

```
        "label": "example-aes",
        "id": "",
        "check-value": "0x9b94bd",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 24
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten AES-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`).

Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<KEY-LENGTH-BYTES>

Gibt die Schlüssellänge in Byte an.

Zulässige Werte:

- 16, 24 und 32

Erforderlich: Ja

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den AES-Schlüssel an. Die maximal zulässige Größe für Client SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von <SESSION> ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI
- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln

key generate-symmetric generic-secret

Der key generate-asymmetric-pair-Befehl generiert einen symmetrischen, generischen, geheimen Schlüssel in Ihrem AWS-CloudHSM-Cluster.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > key help generate-symmetric generic-secret
Generate a generic secret key

Usage: key generate-symmetric generic-secret [OPTIONS] --label <LABEL> --key-length-
bytes <KEY_LENGTH_BYTES>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --label <LABEL>
          Label for the key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --key-length-bytes <KEY_LENGTH_BYTES>
          Key length in bytes
      --attributes [<KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated generic
 secret key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie Sie den key generate-symmetric generic-secret-Befehl verwenden, um einen generischen geheimen Schlüssel zu erstellen.

Example Beispiel: Erstellen Sie einen generischen geheimen Schlüssel

```
aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256
 {
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
```

```
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 256
      }
    }
  }
}
```

Example Beispiel: Erstellen Sie einen generischen geheimen Schlüssel mit optionalen Attributen

```
aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256 \
--attributes token=true encrypt=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
```

```
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 256
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den generierten AES-Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B. `token=true`).

Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<KEY-LENGTH-BYTES>

Gibt die Schlüssellänge in Byte an.

Zulässige Werte:

- 1 bis 800

Erforderlich: Ja

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den generischen geheimen Schlüssel an. Die maximal zulässige Größe für Client SDK 5.11 und höher `label` beträgt 127 Zeichen. Das Client-SDK 5.10 und früher ist auf 126 Zeichen begrenzt.

Erforderlich: Ja

<SESSION>

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel zum Packen nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Standardmäßig handelt es sich bei den generierten Schlüsseln um persistente (Token-)Schlüssel. Das Übergeben von <SESSION> ändert dies und stellt sicher, dass es sich bei einem mit diesem Argument generierten Schlüssel um einen (kurzlebigen) Sitzungsschlüssel handelt.

Erforderlich: Nein

Verwandte Themen

- Schlüsselattribute für CloudHSM-CLI
- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln

Schlüsselimportstift

Der key import pem Befehl in AWS CloudHSM importiert einen Schlüssel im PEM-Format in ein HSM. Sie können ihn verwenden, um öffentliche und außerhalb des HSM erstellte Schlüssel zu importieren.

> **Note**
>
> Verwenden Sie den key generate-file Befehl, um eine Standard-PEM-Datei aus einem öffentlichen Schlüssel oder eine Referenz-PEM-Datei aus einem privaten Schlüssel zu erstellen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key import pem
Import key from a PEM file

Usage: key import pem [OPTIONS] --path <PATH> --label <LABEL> --key-type-
class <KEY_TYPE_CLASS>
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --path PATH>
          Path where the key is located in PEM format
      --label LABEL>
          Label for the imported key
      --key-type-class KEY_TYPE_CLASS>
          Key type and class of the imported key [possible values: ec-public, rsa-
public]
      --attributes [IMPORT_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the imported key
  -h, --help
          Print help
```

Beispiele

Dieses Beispiel zeigt, wie der key import pem Befehl verwendet wird, um einen öffentlichen RSA-Schlüssel aus einer Datei im PEM-Format zu importieren.

Example Beispiel: Importieren Sie einen öffentlichen RSA-Schlüssel

```
aws-cloudhsm > key import pem --path /home/example --label example-imported-key --key-
type-class rsa-public
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
```

```
        "public-exponent": "0x010001",
        "modulus":
 "0x8e9c172c37aa22ed1ce25f7c3a7c936dadc532201400128b044ebb4b96#··3e4930ab910df5a2896eaeb8853cfe
        "modulus-size-bits": 2048
      }
    },
    "message": "Successfully imported key"
  }
}
```

Example Beispiel: Importieren Sie einen öffentlichen RSA-Schlüssel mit optionalen Attributen

```
aws-cloudhsm > key import pem --path /home/example --label example-imported-key-with-
attributes --key-type-class rsa-public --attributes verify=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key-with-attributes",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
```

```
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0x8e9c172c37aa22ed1ce25f7c3a7c936dadc532201400128b044ebb4b96#··3e4930ab910df5a2896eaeb8853cfe
        "modulus-size-bits": 2048
      }
    },
    "message": "Successfully imported key"
  }
}
```


Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PATH>

Gibt den Dateipfad an, in dem sich die Schlüsseldatei befindet.

Erforderlich: Ja

<LABEL>

Gibt eine benutzerdefinierte Bezeichnung für den importierten Schlüssel an. Die maximal zulässige Größe für `label` beträgt 126 Zeichen.

Erforderlich: Ja

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des verpackten Schlüssels.

Mögliche Werte:

- ec-public
- rsa-öffentlich

Erforderlich: Ja

<IMPORT_KEY_ATTRIBUTES>

Gibt eine durch Leerzeichen getrennte Liste von Schlüsselattributen an, die für den importierten Schlüssel festgelegt werden sollen, in der Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (z. B.`token=true`). Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

Verwandte Themen

- Crypto-Zeichen
- Crypto-Verifizierung

key list

Der key list Befehl findet alle Schlüssel für den aktuellen Benutzer, der in Ihrem AWS CloudHSM Cluster vorhanden ist. Die Ausgabe umfasst Schlüssel, die der Benutzer besitzt und die für ihn freigegeben sind, sowie alle öffentlichen Schlüssel im CloudHSM-Cluster.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Syntax

```
aws-cloudhsm > help key list
List the keys the current user owns, shares, and all public keys in the HSM cluster

Usage: key list [OPTIONS]

Options:
```

```
     --cluster-id <CLUSTER_ID>
         Unique Id to choose which of the clusters in the config file to run the
operation against. If not provided, will fall back to the value provided when
interactive mode was started, or error
     --filter [<FILTER>...]
         Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select
matching key(s) to list
     --max-items <MAX_ITEMS>
         The total number of items to return in the command's output. If the total
number of items available is more than the value specified, a next-token is provided
in the command's output. To resume pagination, provide the next-token value in the
starting-token argument of a subsequent command [default: 10]
     --starting-token <STARTING_TOKEN>
         A token to specify where to start paginating. This is the next-token from a
previously truncated response
 -v, --verbose
         If included, prints all attributes and key information for each matched key.
By default each matched key only displays its key-reference and label attribute
 -h, --help
         Print help
```

Beispiele

Die folgenden Beispiele zeigen die verschiedenen Möglichkeiten, wie Sie den key list-Befehl ausführen.

Example Beispiel: Find all keys - default

Dieser Befehl listet die Schlüssel des angemeldeten Benutzers auf, der sich im AWS CloudHSM Cluster befindet.

> **Note**
>
> Standardmäßig werden nur 10 Schlüssel des aktuell angemeldeten Benutzers angezeigt, und nur die Schlüssel `key-reference` und `label` werden als Ausgabe angezeigt. Verwenden Sie die entsprechenden Paginierungsoptionen, um mehr oder weniger Schlüssel als Ausgabe anzuzeigen.

```
aws-cloudhsm > key list
{
```

```
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000003d5",
        "attributes": {
          "label": "test_label_1"
        }
      },
      {
        "key-reference": "0x0000000000000626",
        "attributes": {
          "label": "test_label_2"
        }
      },.
      ...8 keys later...
    ],
    "total_key_count": 56,
    "returned_key_count": 10,
    "next_token": "10"
  }
}
```

Example Beispiel: Find all keys - verbose

Die Ausgabe umfasst Schlüssel, die der Benutzer besitzt und die für ihn freigegeben sind, sowie alle öffentlichen Schlüssel in den HSMs.

> **Note**
>
> Hinweis: Standardmäßig werden nur 10 Schlüssel des aktuell angemeldeten Benutzers angezeigt. Verwenden Sie die entsprechenden Paginierungsoptionen, um mehr oder weniger Schlüssel als Ausgabe anzuzeigen.

```
aws-cloudhsm > key list --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000012000c",
```

```
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "session"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-private-key",
          "id": "",
          "check-value": "0x2a737d",
          "class": "private-key",
          "encrypt": false,
          "decrypt": false,
          "token": false,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": false,
          "trusted": false,
          "unwrap": false,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 122,
          "ec-point":
"0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
          "curve": "secp224r1"
        }
      },
      {
        "key-reference": "0x000000000012000d",
        "key-info": {
          "key-owners": [
```

```
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "session"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-public-key",
          "id": "",
          "check-value": "0x2a737d",
          "class": "public-key",
          "encrypt": false,
          "decrypt": false,
          "token": false,
          "always-sensitive": false,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": false,
          "sign": false,
          "trusted": false,
          "unwrap": false,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 57,
          "ec-point":
"0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
          "curve": "secp224r1"
        }
      }
    ],
      ...8 keys later...
    "total_key_count": 1580,
    "returned_key_count": 10
  }
```

```
}
```

Example Beispiel: Paginated return

Das folgende Beispiel zeigt eine paginierte Teilmenge der Schlüssel, die nur zwei Schlüssel enthält. Das Beispiel bietet dann einen nachfolgenden Aufruf, um die nächsten beiden Schlüssel anzuzeigen.

```
aws-cloudhsm > key list --verbose --max-items 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000000030",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "98a6688d1d964ed7b45b9cec5c4b1909",
          "id": "",
          "check-value": "0xb28a46",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
```

```
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      },
      {
        "key-reference": "0x0000000000000042",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "4ad6cdcbc02044e09fa954143efde233",
          "id": "",
          "check-value": "0xc98104",
          "class": "secret-key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": true,
          "wrap": true,
          "wrap-with-trusted": false,
```

```
          "key-length-bytes": 16
        }
      }
    ],
    "total_key_count": 1580,
    "returned_key_count": 2,
    "next_token": "2"
  }
}
```

Um die nächsten 2 Schlüssel anzuzeigen, kann ein nachfolgender Aufruf getätigt werden:

```
aws-cloudhsm > key list --verbose --max-items 2 --starting-token 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000000081",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "6793b8439d044046982e5b895791e47f",
          "id": "",
          "check-value": "0x3f986f",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
```

```
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      },
      {
        "key-reference": "0x0000000000000089",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "56b30fa05c6741faab8f606d3b7fe105",
          "id": "",
          "check-value": "0xe9201a",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
```

```
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1580,
    "returned_key_count": 2,
    "next_token": "4"
  }
}
```

Weitere Beispiele, die zeigen, wie der wichtige Filtrationsmechanismus in der CloudHSM-CLI funktioniert, finden Sie unter Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln.

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass übereinstimmende Schlüssel zur Liste ausgewählt werden sollen.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<MAX_ITEMS>

Die Gesamtzahl der Elemente, die in der Ausgabe des Befehls zurückgegeben werden sollen. Ist die Gesamtzahl der verfügbaren Elemente größer als der angegebene Wert, wird ein next-token in der Ausgabe des Befehls bereitgestellt. Um die Seitennummerierung fortzusetzen, geben Sie den next-token-Wert im starting-token-Argument eines nachfolgenden Befehls an.

Erforderlich: Nein

<STARTING_TOKEN>

Ein Token für den Beginn der Seitennummerierung. Dies ist das next-token aus einer zuvor abgeschnittenen Antwort.

Erforderlich: Nein

<VERBOSE>

Falls enthalten, werden alle Attribute und Schlüsselinformationen für jeden übereinstimmenden Schlüssel gedruckt. Standardmäßig zeigt jeder übereinstimmende Schlüssel nur seine Schlüsselreferenz und sein Labelattribut an.

Erforderlich: Nein

Verwandte Themen

- key delete
- key generate-file
- key unshare
- Schlüsselattribute für CloudHSM-CLI
- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln

Schlüsselreplikat

Der key replicate Befehl repliziert einen Schlüssel von einem AWS CloudHSM Quellcluster auf einen Zielcluster. AWS CloudHSM

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

> **Note**
>
> Crypto-Benutzer müssen den Schlüssel besitzen, um diesen Befehl verwenden zu können.

Voraussetzungen

- Die Quell- und Zielcluster müssen Klone sein. Das bedeutet, dass einer aus einem Backup des anderen erstellt wurde oder dass beide aus einem gemeinsamen Backup erstellt wurden. Weitere Informationen finden Sie unter Cluster aus Sicherungen erstellen.
- Der Besitzer des Schlüssels muss auf dem Zielcluster vorhanden sein. Wenn der Schlüssel mit anderen Benutzern geteilt wird, müssen diese Benutzer außerdem auch auf dem Zielcluster vorhanden sein.
- Um diesen Befehl ausführen zu können, müssen Sie sowohl auf dem Quell- als auch auf dem Zielcluster als CU angemeldet sein.
  - Im Einzelbefehlsmodus verwendet der Befehl die Umgebungsvariablen CLOUDHSM_PIN und CLOUDHSM_ROLE, um sich auf dem Quellcluster zu authentifizieren. Weitere Informationen finden Sie unter Einzelbefehlsmodus. Um Anmeldeinformationen für den Zielcluster bereitzustellen, müssen Sie zwei zusätzliche Umgebungsvariablen festlegen: DESTINATION_CLOUDHSM_PIN und DESTINATION_CLOUDHSM_ROLE:

    ```
    $ export DESTINATION_CLOUDHSM_ROLE=crypto-user
    ```

    ```
    $ export DESTINATION_CLOUDHSM_PIN=username:password
    ```

  - Im interaktiven Modus müssen sich Benutzer explizit sowohl beim Quell- als auch beim Zielcluster anmelden.

Syntax

```
aws-cloudhsm > help key replicate
Replicate a key from a source to a destination cluster

Usage: key replicate --filter [<FILTER>...] --source-cluster-id <SOURCE_CLUSTER_ID> --
destination-cluster-id <DESTINATION_CLUSTER_ID>

Options:
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select
 matching key on the source cluster
      --source-cluster-id <SOURCE_CLUSTER_ID>
          Source cluster ID
```

```
      --destination-cluster-id <DESTINATION_CLUSTER_ID>
          Destination cluster ID
  -h, --help
          Print help
```

Beispiele

Example Beispiel: Schlüssel replizieren

Dieser Befehl repliziert einen Schlüssel von einem Quellcluster auf einen geklonten Zielcluster.

```
crypto-user-1@cluster-1234abcdefg > key replicate \
      --filter attr.label=example-key \
      --source-cluster-id cluster-1234abcdefg \
      --destination-cluster-id cluster-2345bcdefgh
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x0000000000300006",
      "key-info": {
        "key-owners": [
          {
            "username": "crypto-user-1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-key",
        "id": "0x",
        "check-value": "0x5e118e",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
```

```
        "local": true,
        "modifiable": true,
        "never-extractable": true,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    },
    "message": "Successfully replicated key"
  }
}
```

Argumente

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines passenden Schlüssels auf dem Quellcluster.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Ja

<SOURCE_CLUSTER_ID>

Die Quell-Cluster-ID.

Erforderlich: Ja

<DESTINATION_CLUSTER_ID>

Die Zielcluster-ID.

Erforderlich: Ja

Verwandte Themen

- Mit CLI eine Verbindung zu mehreren Clustern herstellen

key set-attribute

Verwenden Sie den key set-attribute Befehl, um die Attribute der Schlüssel in Ihrem AWS CloudHSM Cluster festzulegen. Nur der CU, der den Schlüssel erstellt hat und folglich Eigentümer des Schlüssels ist, kann die Attribute des Schlüssels ändern.

Eine Liste der wichtigsten Attribute, die in der CloudHSM-CLI verwendet werden können, finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Nur Crypto-Benutzer (CUs) können diesen Befehl ausführen.
- Administratoren können das vertrauenswürdige Attribut festlegen.

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein. Um das vertrauenswürdige Attribut festzulegen, müssen Sie als Admin-Benutzer angemeldet sein.

Syntax

```
aws-cloudhsm > help key set-attribute
Set an attribute for a key in the HSM cluster

Usage: cloudhsm-cli key set-attribute [OPTIONS] --filter [<FILTER>...] --
name <KEY_ATTRIBUTE> --value <KEY_ATTRIBUTE_VALUE>

Options:
      --cluster-id <CLUSTER_ID>         Unique Id to choose which of the clusters in
 the config file to run the operation against. If not provided, will fall back to the
 value provided when interactive mode was started, or error
      --filter [<FILTER>...]            Key reference (e.g. key-
reference=0xabc) or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key to modify
```

```
    --name <KEY_ATTRIBUTE>            Name of attribute to be set
    --value <KEY_ATTRIBUTE_VALUE>...  Attribute value to be set
-h, --help                            Print help
```

Beispiel: Einstellung eines identifizierenden Attributs

Das folgende Beispiel demonstriert, wie Sie mit dem key set-attribute-Befehl das Label setzen.

Example

1. Verwenden Sie den Schlüssel mit dem Label `my_key`, wie hier gezeigt:

```
aws-cloudhsm > key set-attribute --filter attr.label=my_key --name encrypt --value
 false
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

2. Bestätigen Sie mit dem key list-Befehl, dass sich das `encrypt`-Attribut geändert hat:

```
aws-cloudhsm > key list --filter attr.label=my_key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000006400ec",
        "key-info": {
          "key-owners": [
            {
              "username": "bob",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "my_key",
```

```
          "id": "",
          "check-value": "0x6bd9f7",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": true,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": true,
          "unwrap": true,
          "verify": true,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<KEY_ATTRIBUTE>

Gibt den Namen des Attributs des Schlüssels an.

Erforderlich: Ja

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten CloudHSM-CLI-Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Nein

<KEY_ATTRIBUTE_VALUE>

Gibt den Wert des Attributs des Schlüssels an.

Erforderlich: Ja

<KEY_REFERENCE>

Eine hexadezimale oder dezimale Darstellung des Schlüssels. (z. B. ein Tastenkürzel).

Erforderlich: Nein

Verwandte Themen

- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln
- Schlüsselattribute für CloudHSM-CLI

key share

Der key share Befehl teilt sich einen Schlüssel mit anderen CUs in Ihrem AWS CloudHSM Cluster.

Nur der CU, der den Schlüssel erstellt hat und somit Eigentümer des Schlüssels ist, kann den Schlüssel teilen. Benutzer, mit denen ein Schlüssel geteilt wird, können den Schlüssel für kryptografische Operationen verwenden, aber sie können den Schlüssel nicht löschen, exportieren, teilen oder das Teilen aufheben. Darüber hinaus können diese Benutzer Schlüsselattribute nicht ändern.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key share
Share a key in the HSM cluster with another user

Usage: key share --filter [<FILTER>...] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key for sharing

      --username <USERNAME>
          A username with which the key will be shared

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

  -h, --help
          Print help (see a summary with '-h')
```

Beispiel: Teilen Sie sich einen Schlüssel mit einem anderen CU

Das folgende Beispiel zeigt, wie Sie den key share-Befehl verwenden, um einen Schlüssel mit CU `alice` zu teilen.

Example

1. Führen Sie den key share-Befehl aus, um den Schlüssel mit `alice` zu teilen.

```
aws-cloudhsm > key share --filter attr.label="rsa_key_to_share" attr.class=private-
key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key shared successfully"
  }
}
```

2. Führen Sie den Befehl key list aus.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
```

```
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
  "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
```

```
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. Überprüfen Sie in der obigen Liste, ob `alice` in der Liste von `shared-users` ist.

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Ja

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Um die Rolle des Benutzers zu erfahren, verwenden Sie den Befehl user list. Ausführliche

Informationen zu den Benutzertypen in einem HSM finden Sie unter Grundlegendes zu HSM-Benutzern.

Erforderlich: Ja

Verwandte Themen

- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln
- Schlüsselattribute für CloudHSM-CLI

key unshare

key unshareMit dem Befehl wird die gemeinsame Nutzung eines Schlüssels mit anderen CUs in Ihrem AWS CloudHSM Cluster aufgehoben.

Nur der CU, der den Schlüssel erstellt hat und somit Eigentümer des Schlüssels ist, kann das Teilen des Schlüssels aufheben. Benutzer, mit denen ein Schlüssel geteilt wird, können den Schlüssel für kryptografische Operationen verwenden, aber sie können den Schlüssel nicht löschen, exportieren, teilen oder das Teilen aufheben. Darüber hinaus können diese Benutzer Schlüsselattribute nicht ändern.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unshare
Unshare a key in the HSM cluster with another user

Usage: key unshare --filter [<FILTER>...] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
```

```
        Unique Id to choose which of the clusters in the config file to run the
operation against. If not provided, will fall back to the value provided when
interactive mode was started, or error

     --filter [<FILTER>...]
        Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
matching key for unsharing

     --username <USERNAME>
        A username with which the key will be unshared

     --role <ROLE>
        Role the user has in the cluster

        Possible values:
        - crypto-user: A CryptoUser has the ability to manage and use keys
        - admin:       An Admin has the ability to manage user accounts

 -h, --help
        Print help (see a summary with '-h')
```

Beispiel: Die gemeinsame Nutzung eines Schlüssels mit einer anderen CU rückgängig machen

Das folgende Beispiel zeigt, wie Sie den key unshare-Befehl verwenden, um einen Schlüssel mit CU `alice` nicht mehr zu teilen.

Example

1. Führen Sie den key list-Befehl aus und filtern Sie nach dem bestimmten Schlüssel, mit dem Sie das Teilen mit `alice` beenden möchten.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
```

```
            "username": "cu3",
            "key-coverage": "full"
          }
        ],
        "shared-users": [
          {
            "username": "cu2",
            "key-coverage": "full"
          },
          {
            "username": "cu1",
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
          {
            "username": "cu5",
            "key-coverage": "full"
          },
          {
            "username": "cu6",
            "key-coverage": "full"
          },
          {
            "username": "cu7",
            "key-coverage": "full"
          },
          {
            "username": "alice",
            "key-coverage": "full"
          }
        ],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa_key_to_share",
        "id": "",
        "check-value": "0xae8ff0",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
```

```
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

2. Bestätigen Sie, dass `alice` in der `shared-users`-Ausgabe enthalten ist, und führen Sie den folgenden key unshare-Befehl aus, um das Teilen des Schlüssels mit `alice` aufzuheben.

```
aws-cloudhsm > key unshare --filter attr.label="rsa_key_to_share"
 attr.class=private-key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key unshared successfully"
  }
}
```

3. Führen Sie den `key list`-Befehl erneut aus, um zu bestätigen, dass das Teilen des Schlüssels mit `alice` aufgehoben wurde.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
          ],
          "cluster-coverage": "full"
        },
```

```
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```


Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein passender Schlüssel zum Löschen ausgewählt werden soll.

Eine Liste der unterstützten Schlüsselattribute finden Sie unter Schlüsselattribute für CloudHSM-CLI.

Erforderlich: Ja

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Um die Rolle des Benutzers zu erfahren, verwenden Sie den Befehl user list. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter Grundlegendes zu HSM-Benutzern.

Erforderlich: Ja

Verwandte Themen

- Verwenden der CloudHSM-CLI zum Filtern von Schlüsseln
- Schlüsselattribute für CloudHSM-CLI

Schlüssel entpacken

Der key unwrap übergeordnete Befehl in der CloudHSM-CLI importiert einen verschlüsselten (verpackten) symmetrischen oder asymmetrischen privaten Schlüssel aus einer Datei in das HSM. Dieser Befehl wurde entwickelt, um verschlüsselte Schlüssel zu importieren, die vom Schlüsselumbruch Befehl umschlossen wurden. Er kann jedoch auch zum Entpacken von Schlüsseln

verwendet werden, die mit anderen Tools umschlossen wurden. In diesen Situationen empfehlen wir jedoch, die Softwarebibliotheken PKCS#11 oder JCE zu verwenden, um den Schlüssel zu entpacken.

- Schlüssel auspacken aes-gcm
- Schlüssel auspacken aes-no-pad
- Schlüssel entpacken aes-pkcs5-Pad
- Schlüssel auspacken aes-zero-pad
- Schlüssel auspacken cloudhsm-aes-gcm
- Schlüssel zum Auspacken von RSA-AES
- key unwrap rsa-oaep
- Schlüssel unwrap rsa-pkcs

Schlüssel auspacken aes-gcm

Der key unwrap aes-gcm Befehl entpackt mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus einen Payload-Schlüssel in den Cluster. `AES-GCM`

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den key unwrap aes-gcm Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf gesetzt sein`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unwrap aes-gcm
```

```
Usage: key unwrap aes-gcm [OPTIONS] --filter [<FILTER>...] --tag-length-
bits <TAG_LENGTH_BITS> --key-type-class <KEY_TYPE_CLASS> --label <LABEL> --iv <IV> <--
data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --iv <IV>
          Initial value used to wrap the key, in hex
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie der key unwrap aes-gcm Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf gesetzt ist`true`.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten umschlossenen Schlüsseldaten

```
aws-cloudhsm > key unwrap aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --iv
 0xf90613bb8e337ec0339aad21 --data xvslgrtg8kHzrvekny97tLSIeokpPwV8
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e4",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
```

```
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
aws-cloudhsm > key unwrap aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --iv
 0xf90613bb8e337ec0339aad21 --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e4",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
```

```
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die verpackten Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<AAD>

AES GCM-Wert für zusätzliche authentifizierte Daten (AAD), in Hexadezimalzahl.

Erforderlich: Nein

<TAG_LENGTH_BITS>

Länge des Aes-GCM-Tags in Bit.

Erforderlich: Ja

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`,`ec-private`,`generic-secret`,,`rsa-private`].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

<IV>

Anfangswert, der zum Umschließen des Schlüssels verwendet wird, in Hexadezimalform.

Erforderlich: Nein

Verwandte Themen

- Schlüsselumbruch

- Schlüssel entpacken

Schlüssel auspacken aes-no-pad

Der key unwrap aes-no-pad Befehl entpackt mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus einen Payload-Schlüssel in den Cluster. `AES-NO-PAD`

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den key unwrap aes-no-pad Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf gesetzt sein`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unwrap aes-no-pad
Usage: key unwrap aes-no-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
```

```
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie der key unwrap aes-no-pad Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf gesetzt ist`true`.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```
aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data eXK3PMAOnKM9y3YX6brbhtMoC060EOH9
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
```

```
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
```

```
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die verpackten Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`,`ec-private`,`generic-secret`,,`rsa-private`].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüssel entpacken aes-pkcs5-Pad

Der key unwrap aes-pkcs5-pad Befehl entpackt einen Payload-Schlüssel mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus. `AES-PKCS5-PAD`

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den key unwrap aes-pkcs5-pad Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf gesetzt sein`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unwrap aes-pkcs5-pad
Usage: key unwrap aes-pkcs5-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
```

```
          Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
to unwrap with
     --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
     --data <DATA>
          Base64 encoded wrapped key data
     --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
     --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
generic-secret, rsa-private]
     --label <LABEL>
          Label for the unwrapped key
     --session
          Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
 -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie der key unwrap aes-pkcs5-pad Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf gesetzt ist`true`.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```
aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data MbuYNresfOKyGNnxKWen88nSfX+uUE/0qmGofSisicY=
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
```

```
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e3",
```

```
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die verpackten Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`,`ec-private`,`generic-secret`,,`rsa-private`].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüssel auspacken aes-zero-pad

Der key unwrap aes-zero-pad Befehl entpackt mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus einen Payload-Schlüssel in den Cluster. `AES-ZERO-PAD`

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den key unwrap aes-no-pad Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf gesetzt sein`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unwrap aes-zero-pad
Usage: key unwrap aes-zero-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>
```

```
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie der key unwrap aes-zero-pad Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf gesetzt ist`true`.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```
aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data L1wVlL/YeBNVAw6Mpk3owFJZXBzDLONt
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e7",
```

```
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
```

```
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die verpackten Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`,`ec-private`,`generic-secret`,,`rsa-private`].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüssel auspacken cloudhsm-aes-gcm

Der key unwrap cloudhsm-aes-gcm Befehl entpackt mithilfe des AES-Wrapping-Schlüssels und des Entpackungsmechanismus einen Payload-Schlüssel in den Cluster. `CLOUDHSM-AES-GCM`

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die Schlüssel, die von generiert wurden. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den key unwrap cloudhsm-aes-gcm Befehl verwenden zu können, müssen Sie den AES-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben und sein `unwrap` Attribut muss auf gesetzt sein`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unwrap cloudhsm-aes-gcm
Usage: key unwrap cloudhsm-aes-gcm [OPTIONS] --filter [<FILTER>...] --tag-length-
bits <TAG_LENGTH_BITS> --key-type-class <KEY_TYPE_CLASS> --label <LABEL> <--data-
path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie der key unwrap cloudhsm-aes-gcm Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf gesetzt ist`true`.

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten umschlossenen Schlüsseldaten

```
aws-cloudhsm > key unwrap cloudhsm-aes-gcm --key-type-class aes --label aes-
unwrapped --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --data
 6Rn8nkjEriDYlnP3P8nPkYQ8hplOEJ899zsrF+aTB0i/fIlZ
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
```

```
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
aws-cloudhsm > key unwrap cloudhsm-aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --data-path payload-
key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
```

```
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die verpackten Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<AAD>

AES GCM-Wert für zusätzliche authentifizierte Daten (AAD), in Hexadezimalzahl.

Erforderlich: Nein

<TAG_LENGTH_BITS>

Länge des Aes-GCM-Tags in Bit.

Erforderlich: Ja

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`,`ec-private`,`generic-secret`,,`rsa-private`].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüssel zum Auspacken von RSA-AES

Der key unwrap rsa-aes Befehl entpackt einen Payload-Schlüssel mithilfe eines privaten RSA-Schlüssels und des Entpackungsmechanismus. `RSA-AES`

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die von generierten Schlüssel. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den verwenden zu könnenkey unwrap rsa-aes, müssen Sie den privaten RSA-Schlüssel oder den öffentlichen RSA-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben, und sein `unwrap` Attribut muss auf gesetzt sein. `true`

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unwrap rsa-aes
Usage: key unwrap rsa-aes [OPTIONS] --filter [<FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --key-type-class <KEY_TYPE_CLASS> --label <LABEL>
 <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
```

```
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
 mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

Beispiel

Diese Beispiele zeigen, wie der key unwrap rsa-aes Befehl unter Verwendung des privaten RSA-Schlüssels mit dem unwrap Attributwert auf verwendet wird. `true`

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```
aws-cloudhsm > key unwrap rsa-aes --key-type-class aes --label aes-unwrapped
 --filter attr.label=rsa-private-key-example --hash-function sha256 --
mgf mgf1-sha256 --data HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
```

```
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
aws-cloudhsm > key unwrap rsa-aes --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-key-example --hash-function sha256 --mgf mgf1-sha256 --data-
path payload-key.pem
{
```

```
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die verpackten Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`,`ec-private`,`generic-secret`,,`rsa-private`].

Erforderlich: Ja

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<MGF>

Definiert die Funktion zur Maskengenerierung.

> **Note**
>
> Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<LABEL>

Etikett für den unverpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

key unwrap rsa-oaep

Der key unwrap rsa-oaep Befehl entpackt einen Payload-Schlüssel mithilfe des privaten RSA-Schlüssels und des Entpackungsmechanismus. `RSA-OAEP`

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die von generierten Schlüssel. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den key unwrap rsa-oaep Befehl verwenden zu können, müssen Sie den privaten RSA-Schlüssel oder den öffentlichen RSA-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben, und sein `unwrap` Attribut muss auf gesetzt sein. `true`

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unwrap rsa-oaep
Usage: key unwrap rsa-oaep [OPTIONS] --filter [<FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --key-type-class <KEY_TYPE_CLASS> --label <LABEL>
 <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <<DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
 mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie der key unwrap rsa-oaep Befehl unter Verwendung des privaten RSA-Schlüssels mit dem unwrap Attributwert auf verwendet wird. `true`

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```
aws-cloudhsm > key unwrap rsa-oaep --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-example-key --hash-function sha256 --mgf mgf1-sha256 --data
 OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/jGkDNdB4qyTAOQwEpggGf6v
+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d+F2KOONXsSxMhmxzzNG/
gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw==
{
```

```
  "data": {
    "key": {
      "key-reference": "0x00000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
aws-cloudhsm > key unwrap rsa-oaep --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-example-key --hash-function sha256 --mgf mgf1-sha256 --data-
path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
```

```
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die verpackten Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`,`ec-private`,`generic-secret`,,`rsa-private`].

Erforderlich: Ja

<HASH_FUNCTION>

Spezifiziert die Hash-Funktion.

Zulässige Werte:

- sha1
- sha224
- sha256
- sha384
- sha512

Erforderlich: Ja

<MGF>

Definiert die Funktion zur Maskengenerierung.

> **Note**
>
> Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

<LABEL>

Etikett für den unverpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüssel unwrap rsa-pkcs

Der key unwrap rsa-pkcs Befehl entpackt einen Payload-Schlüssel mithilfe des privaten RSA-Schlüssels und des Entpackungsmechanismus. `RSA-PKCS`

Unverpackte Schlüssel können auf die gleiche Weise verwendet werden wie die von generierten Schlüssel. AWS CloudHSM Um anzuzeigen, dass sie nicht lokal generiert wurden, ist ihr `local` Attribut auf `false` gesetzt.

Um den unwrap rsa-pkcs Befehl key verwenden zu können, müssen Sie den privaten RSA-Schlüssel oder den öffentlichen RSA-Wrapping-Schlüssel in Ihrem AWS CloudHSM Cluster haben, und sein `unwrap` Attribut muss auf gesetzt sein. `true`

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key unwrap rsa-pkcs
Usage: key unwrap rsa-pkcs [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>
```

```
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

Beispiele

Diese Beispiele zeigen, wie der key unwrap rsa-oaep Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `unwrap` Attributwert auf gesetzt ist. `true`

Example Beispiel: Entpacken Sie einen Payload-Schlüssel aus Base64-kodierten verpackten Schlüsseldaten

```
aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label
 aes-unwrapped --filter attr.label=rsa-private-key-example --data
 am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnGl4GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYRO1lGlfq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcpOOobJlITCoXhtMRepSdO4OggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg==
{
```

```
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example Beispiel: Entpacken Sie einen Payload-Schlüssel, der über einen Datenpfad bereitgestellt wird

```
aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-key-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
```

```
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`, dass ein Schlüssel zum Entpacken ausgewählt werden soll.

Erforderlich: Ja

<DATA_PATH>

Pfad zur Binärdatei, die die verpackten Schlüsseldaten enthält.

Erforderlich: Ja (sofern nicht über Base64-kodierte Daten bereitgestellt)

<DATA>

Base64-kodierte umhüllte Schlüsseldaten.

Erforderlich: Ja (sofern nicht über den Datenpfad angegeben)

<ATTRIBUTES>

Durch Leerzeichen getrennte Liste von Schlüsselattributen in Form von `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` für den umschlossenen Schlüssel.

Erforderlich: Nein

<KEY_TYPE_CLASS>

Schlüsseltyp und Klasse des umschlossenen Schlüssels [mögliche Werte: `aesdes3`,`ec-private`,`generic-secret`,,`rsa-private`].

Erforderlich: Ja

<LABEL>

Bezeichnung für den entpackten Schlüssel.

Erforderlich: Ja

<SESSION>

Erstellt einen Sitzungsschlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Erforderlich: Nein

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüsselumbruch

Der key wrap Befehl in der CloudHSM-CLI exportiert eine verschlüsselte Kopie eines symmetrischen oder asymmetrischen privaten Schlüssels aus dem HSM in eine Datei. Wenn Sie ausführenkey wrap, geben Sie zwei Dinge an: den zu exportierenden Schlüssel und die Ausgabedatei. Der zu exportierende Schlüssel ist ein Schlüssel auf dem HSM, der den zu exportierenden Schlüssel verschlüsselt (umschließt).

Der key wrap Befehl entfernt den Schlüssel nicht aus dem HSM und hindert Sie nicht daran, ihn in kryptografischen Operationen zu verwenden. Sie können den gleichen Schlüssel mehrmals exportieren. Um den verschlüsselten Schlüssel zurück in das HSM zu importieren, verwenden Sie Schlüssel entpacken. Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, für die der Schlüssel freigegeben ist, können den Schlüssel nur in kryptografischen Operationen verwenden.

Der key wrap Befehl besteht aus den folgenden Unterbefehlen:

- Schlüsselhülle aes-gcm
- Schlüsselhülle aes-no-pad
- Schlüsselhülle aes-pkcs5-Pad

- Schlüsselhülle aes-zero-pad
- Schlüsselhülle cloudhsm-aes-gcm
- Schlüsselumschlag rsa-aes
- Schlüsselhülle rsa-oaep
- Schlüsselumschlag rsa-pkcs

Schlüsselhülle aes-gcm

Der key wrap aes-gcm Befehl umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des Wrapping-Mechanismus. `AES-GCM` Das `extractable` Attribut des Payload-Schlüssels muss auf gesetzt sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap aes-gcm Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können einen AES-Schlüssel zum Umschließen generieren, indem Sie den key generate-symmetric aes Befehl und das `wrap` Attribut auf setzen`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key wrap aes-gcm
Usage: key wrap aes-gcm [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --tag-length-bits <TAG_LENGTH_BITS>
```

```
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
  -h, --help
          Print help
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap aes-gcm Befehl mithilfe eines AES-Schlüssels verwendet wird.

Example

```
aws-cloudhsm > key wrap aes-gcm --payload-filter attr.label=payload-key --wrapping-
filter attr.label=aes-example --tag-length-bits 64  --aad 0x10
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "iv": "0xf90613bb8e337ec0339aad21",
    "wrapped_key_data": "xvslgrtg8kHzrvekny97tLSIeokpPwV8"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Ja

<WRAPPING_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Umbruchschlüssels.

Erforderlich: Ja

<AAD>

AES GCM-Wert für zusätzliche authentifizierte Daten (AAD), in Hexadezimalzahl.

Erforderlich: Nein

<TAG_LENGTH_BITS>

Länge des AES-GCM-Tags in Bit.

Erforderlich: Ja

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüsselhülle aes-no-pad

Der key wrap aes-no-pad Befehl umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des Wrapping-Mechanismus. `AES-NO-PAD` Das `extractable` Attribut des Payload-Schlüssels muss auf gesetzt sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap aes-no-pad Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können mithilfe des key generate-symmetric aes Befehls einen AES-Schlüssel für das Umschließen generieren, und das `wrap` Attribut ist auf gesetzt`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key wrap aes-no-pad
Usage: key wrap aes-no-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
```

```
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
  -h, --help
          Print help
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap aes-no-pad Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `wrap` Attributwert auf gesetzt ist`true`.

Example

```
aws-cloudhsm > key wrap aes-no-pad --payload-filter attr.label=payload-key --wrapping-
filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "eXK3PMAOnKM9y3YX6brbhtMoC060EOH9"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Ja

<WRAPPING_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Umbruchschlüssels.

Erforderlich: Ja

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüsselhülle aes-pkcs5-Pad

Der key wrap aes-pkcs5-pad Befehl umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und dem Wrapping-Mechanismus. `AES-PKCS5-PAD` Das `extractable` Attribut des Payload-Schlüssels muss auf gesetzt sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap aes-pkcs5-pad Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können mithilfe des key generate-symmetric aes Befehls einen AES-Schlüssel für das Umschließen generieren, und das `wrap` Attribut ist auf gesetzt`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key wrap aes-pkcs5-pad
Usage: key wrap aes-pkcs5-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
  -h, --help
          Print help
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap aes-pkcs5-pad Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `wrap` Attributwert auf gesetzt ist`true`.

Example

```
aws-cloudhsm > key wrap aes-pkcs5-pad --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "MbuYNresfOKyGNnxKWen88nSfX+uUE/0qmGofSisicY="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Ja

<WRAPPING_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Umbruchschlüssels.

Erforderlich: Ja

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüsselhülle aes-zero-pad

Der key wrap aes-zero-pad Befehl umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und dem Wrapping-Mechanismus. `AES-ZERO-PAD` Das `extractable` Attribut des Payload-Schlüssels muss auf gesetzt sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap aes-zero-pad Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können mit dem key generate-symmetric aes Befehl, dessen `wrap` Attribut auf gesetzt ist, einen AES-Schlüssel zum Umschließen generieren`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key wrap aes-zero-pad
Usage: key wrap aes-zero-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
  -h, --help
          Print help
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap aes-zero-pad Befehl mithilfe eines AES-Schlüssels verwendet wird, dessen `wrap` Attributwert auf gesetzt ist`true`.

Example

```
aws-cloudhsm > key wrap aes-zero-pad --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "L1wVlL/YeBNVAw6Mpk3owFJZXBzDLONt"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Ja

<WRAPPING_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Umbruchschlüssels.

Erforderlich: Ja

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüsselhülle cloudhsm-aes-gcm

Der key wrap cloudhsm-aes-gcm Befehl umschließt einen Payload-Schlüssel mithilfe eines AES-Schlüssels auf dem HSM und des Wrapping-Mechanismus. `CLOUDHSM-AES-GCM` Das `extractable` Attribut des Payload-Schlüssels muss auf gesetzt sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap cloudhsm-aes-gcm Befehl verwenden zu können, benötigen Sie zunächst einen AES-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können einen AES-Schlüssel zum Umschließen generieren, indem Sie den key generate-symmetric aes Befehl verwenden und das `wrap` Attribut auf gesetzt haben`true`.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key wrap cloudhsm-aes-gcm
Usage: key wrap cloudhsm-aes-gcm [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...] --tag-length-bits <TAG_LENGTH_BITS>
```

```
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
  -h, --help
          Print help
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap cloudhsm-aes-gcm Befehl mithilfe eines AES-Schlüssels verwendet wird.

Example

```
aws-cloudhsm > key wrap cloudhsm-aes-gcm --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example --tag-length-bits 64 --aad 0x10
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "6Rn8nkjEriDYlnP3P8nPkYQ8hplOEJ899zsrF+aTB0i/fIlZ"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Ja

<WRAPPING_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Umbruchschlüssels.

Erforderlich: Ja

<AAD>

AES GCM-Wert für zusätzliche authentifizierte Daten (AAD), in Hexadezimalzahl.

Erforderlich: Nein

<TAG_LENGTH_BITS>

Länge des AES-GCM-Tags in Bit.

Erforderlich: Ja

Verwandte Themen

- Schlüsselumbruch

- Schlüssel entpacken

Schlüsselumschlag rsa-aes

Der key wrap rsa-aes Befehl umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des RSA-AES-Wrapping-Mechanismus. Das Attribut des Payload-Schlüssels muss auf gesetzt sein. `extractable` `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap rsa-aes Befehl verwenden zu können, benötigen Sie zunächst einen RSA-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können ein RSA-Schlüsselpaar generieren, indem Sie den key generate-asymmetric-pair Befehl verwenden und das `wrap` Attribut auf setzen. `true`

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key wrap rsa-aes
Usage: key wrap rsa-aes [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --hash-function <HASH_FUNCTION> --mgf <MGF>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
```

```
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
 mgf1-sha256, mgf1-sha384, mgf1-sha512]
  -h, --help
          Print help
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap rsa-ae Befehl mithilfe eines öffentlichen RSA-Schlüssels verwendet wird, dessen `wrap` Attributwert auf gesetzt ist. `true`

Example

```
aws-cloudhsm > key wrap rsa-aes --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Ja

<WRAPPING_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Umbruchschlüssels.

Erforderlich: Ja

<MGF>

Definiert die Funktion zur Maskengenerierung.

> **Note**
>
> Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüsselhülle rsa-oaep

Der key wrap rsa-oaep Befehl umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des Wrapping-Mechanismus. `RSA-OAEP` Das `extractable` Attribut des Payload-Schlüssels muss auf gesetzt sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap rsa-oaep Befehl verwenden zu können, müssen Sie zunächst über einen RSA-Schlüssel in Ihrem AWS CloudHSM Cluster verfügen. Sie können ein RSA-Schlüsselpaar generieren, indem Sie den key generate-asymmetric-pair Befehl verwenden und das `wrap` Attribut auf setzen. `true`

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key wrap rsa-oaep
Usage: key wrap rsa-oaep [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --hash-function <HASH_FUNCTION> --mgf <MGF>

Options:
      --cluster-id <CLUSTER_ID>
```

```
        Unique Id to choose which of the clusters in the config file to run the
operation against. If not provided, will fall back to the value provided when
interactive mode was started, or error
     --payload-filter [<PAYLOAD_FILTER>...]
         Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
payload key
     --wrapping-filter [<WRAPPING_FILTER>...]
         Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
wrapping key
     --path <PATH>
         Path to the binary file where the wrapped key data will be saved
     --hash-function <HASH_FUNCTION>
         Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
     --mgf <MGF>
         Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
mgf1-sha256, mgf1-sha384, mgf1-sha512]
 -h, --help
         Print help
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap rsa-oaep Befehl mithilfe eines öffentlichen RSA-Schlüssels verwendet wird, dessen `wrap` Attributwert auf gesetzt ist. `true`

Example

```
aws-cloudhsm > key wrap rsa-oaep --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/
jGkDNdB4qyTAOQwEpggGf6v+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d
+F2KOONXsSxMhmxzzNG/gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/
wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw=="
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Ja

<WRAPPING_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Umbruchschlüssels.

Erforderlich: Ja

<MGF>

Definiert die Funktion zur Maskengenerierung.

> **Note**
>
> Die Hash-Funktion der Maskengenerierungsfunktion muss mit der Hash-Funktion des Signaturmechanismus übereinstimmen.

Zulässige Werte

- mgf1-sha1
- mgf1-sha224

- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Erforderlich: Ja

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

Schlüsselumschlag rsa-pkcs

Der key wrap rsa-pkcs Befehl umschließt einen Payload-Schlüssel mithilfe eines öffentlichen RSA-Schlüssels auf dem HSM und des Wrapping-Mechanismus. `RSA-PKCS` Das `extractable` Attribut des Payload-Schlüssels muss auf gesetzt sein. `true`

Nur der Besitzer eines Schlüssels, d. h. der Crypto-Benutzer (CU), der den Schlüssel erstellt hat, kann den Schlüssel umschließen. Benutzer, die den Schlüssel gemeinsam nutzen, können den Schlüssel für kryptografische Operationen verwenden.

Um den key wrap rsa-pkcs Befehl verwenden zu können, benötigen Sie zunächst einen RSA-Schlüssel in Ihrem AWS CloudHSM Cluster. Sie können ein RSA-Schlüsselpaar generieren, indem Sie den key generate-asymmetric-pair Befehl verwenden und das `wrap` Attribut auf setzen. `true`

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (CUs)

Voraussetzungen

- Um diesen Befehl auszuführen, müssen Sie als CU angemeldet sein.

Syntax

```
aws-cloudhsm > help key wrap rsa-pkcs
```

```
Usage: key wrap rsa-pkcs [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
  -h, --help
          Print help
```

Beispiel

Dieses Beispiel zeigt, wie der key wrap rsa-pkcs Befehl mit einem öffentlichen RSA-Schlüssel verwendet wird.

Example

```
aws-cloudhsm > key wrap rsa-pkcs --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000007008da",
    "wrapped_key_data": "am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp
+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnGl4GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYRO1lGlfq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcpOOobJlITCoXhtMRepSdO4OggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg=="
  }
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PAYLOAD_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Payload-Schlüssels.

Erforderlich: Ja

<PATH>

Pfad zur Binärdatei, in der die verpackten Schlüsseldaten gespeichert werden.

Erforderlich: Ja

<WRAPPING_FILTER>

Schlüsselreferenz (z. B.`key-reference=0xabc`) oder durch Leerzeichen getrennte Liste von Schlüsselattributen in der Form `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` zur Auswahl eines Umbruchschlüssels.

Erforderlich: Ja

Verwandte Themen

- Schlüsselumbruch
- Schlüssel entpacken

login

Sie können den login-Befehl in CloudHSM-CLI verwenden, um sich bei jedem HSM in einem Cluster an- und abzumelden.

> **Note**
>
> Wenn Sie mehr als fünf falsche Anmeldeversuche tätigen, wird Ihr Konto gesperrt. Um das Konto zu entsperren, muss ein Administrator Ihr Passwort mit dem Befehl user change-password in cloudhsm_cli zurücksetzen.

Um Probleme bei der An- und Abmeldung zu beheben

Wenn Sie mehr als ein HSM in Ihrem Cluster haben, sind Ihnen möglicherweise weitere falsche Anmeldeversuche gestattet, bevor Ihr Konto gesperrt wird. Dies liegt daran, dass der CloudHSM-Client die Last auf verschiedene HSMs verteilt. Aus diesem Grund beginnt der Anmeldeversuch möglicherweise nicht jedes Mal auf demselben HSM. Wenn Sie diese Funktion testen, empfehlen wir Ihnen, dies auf einem Cluster mit nur einem aktiven HSM zu tun.

Wenn Sie Ihren Cluster vor Februar 2018 erstellt haben, wird Ihr Konto nach 20 falschen Anmeldeversuchen gesperrt.

Benutzertyp

Die folgenden Benutzer können diese Befehle ausführen.

- Nicht aktivierter Admin
- Admin.
- Crypto-Benutzer (Crypto User, CU)

Syntax

```
aws-cloudhsm > help login
Login to your cluster

USAGE:
    cloudhsm-cli login [OPTIONS] --username <USERNAME> --role <ROLE> [COMMAND]

Commands:
  mfa-token-sign  Login with token-sign mfa
  help            Print this message or the help of the given subcommand(s)

OPTIONS:
```

```
        --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

        --username <USERNAME>
            Username to access the Cluster

        --role <ROLE>
            Role the user has in the Cluster

           Possible values:
           - crypto-user: A CryptoUser has the ability to manage and use keys
           - admin:       An Admin has the ability to manage user accounts

        --password <PASSWORD>
           Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

  -h, --help
          Print help (see a summary with '-h')
```

Beispiel

Example

Mit diesem Befehl melden Sie sich bei allen HSMs in einem Cluster mit den Anmeldedaten eines Admin-Benutzers namens `admin1`.

```
aws-cloudhsm >login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Gültige Werte sind admin, crypto-user.

Verwenden Sie den user list-Befehl, um die Rolle des Benutzers abzurufen. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter HSM-Benutzer verstehen.

<PASSWORD>

Gibt das Passwort des Benutzers an, der sich bei den HSMs anmeldet.

Verwandte Themen

- Erste Schritte mit CloudHSM-CLI
- Aktivieren des Clusters

einloggen mfa-token-sign

Verwenden Sie den login mfa-token-sign-Befehl in der AWS CloudHSM -CloudHSM-CLI-Anmeldung mit Multi-Faktor-Authentifizierung. Um diesen Befehl verwenden zu können, müssen Sie zuerst MFA für CloudHSM-CLI einrichten.

Benutzertyp

Die folgenden Benutzer können diese Befehle ausführen.

- Admin.
- Crypto-Benutzer (Crypto User, CU)

Syntax

```
aws-cloudhsm > help login mfa-token-sign
Login with token-sign mfa

USAGE:
    login --username <USERNAME> --role <ROLE> mfa-token-sign --token <TOKEN>

OPTIONS:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --token <TOKEN>            Filepath where the unsigned token file will be written
  -h, --help                     Print help
```

Beispiel

Example

```
aws-cloudhsm >login --username test_user --role admin mfa-token-sign --token /home/
valid.token
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
{
  "error_code": 0,
  "data": {
    "username": "test_user",
    "role": "admin"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<TOKEN>

Dateipfad, in den die unsignierte Tokendatei geschrieben wird.

Erforderlich: Ja

Verwandte Themen

- Erste Schritte mit CloudHSM-CLI
- Aktivieren des Clusters
- Verwenden von CloudHSM-CLI zur Verwaltung von MFA

logout

Sie können den logout-Befehl in CloudHSM-CLI verwenden, um sich bei jedem HSM in einem Cluster abzumelden.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer (Crypto User, CU)

Syntax

```
aws-cloudhsm > help logout
Logout of your cluster

USAGE:
    logout

OPTIONS:
        --cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
    -h, --help                    Print help information
    -V, --version                 Print version information
```

Beispiel

Example

Mit diesem Befehl werden Sie von allen HSMs in einem Cluster abgemeldet.

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

Verwandte Themen

- Erste Schritte mit CloudHSM-CLI
- Aktivieren des Clusters

user

user ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen benutzerspezifischen Befehl erzeugen. Derzeit besteht die Benutzerkategorie aus den folgenden Befehlen:

- user change-mfa
- user change-password
- user create
- user delete
- user list

user change-mfa

Derzeit besteht diese Kategorie aus dem folgenden Unterbefehl:

- user change-mfa token-sign

user change-mfa token-sign

Verwenden Sie den user change-mfa-Befehl in der CloudHSM-CLI, um die Einrichtung der Multi-Faktor-Authentifizierung (MFA) eines Benutzerkontos zu aktualisieren. Jedes Benutzerkonto kann

diesen Befehl ausführen. Konten mit der Administratorrolle können diesen Befehl für andere Benutzer ausführen.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer

Syntax

Derzeit steht Benutzern nur eine einzige Multi-Faktor-Strategie zur Verfügung: Token Sign.

```
aws-cloudhsm > help user change-mfa
Change a user's Mfa Strategy

Usage:
    user change-mfa <COMMAND>

Commands:
  token-sign  Register or Deregister a public key using token-sign mfa strategy
  help        Print this message or the help of the given subcommand(s)
```

Die Token-Sign-Strategie verlangt nach einer Token-Datei, in die unsignierte Token geschrieben werden sollen.

```
aws-cloudhsm > help user change-mfa token-sign
Register or Deregister a public key using token-sign mfa strategy

Usage: user change-mfa token-sign [OPTIONS] --username <USERNAME> --role <ROLE> <--
token <TOKEN>|--deregister>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
```

```
          Username of the user that will be modified

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --change-password <CHANGE_PASSWORD>
          Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

      --token <TOKEN>
          Filepath where the unsigned token file will be written. Required for enabling
 MFA for a user

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

      --deregister
          Deregister the MFA public key, if present

      --change-quorum
          Change the Quorum public key along with the MFA key

  -h, --help
          Print help (see a summary with '-h')
```

Beispiel

Dieser Befehl schreibt ein unsigniertes Token pro HSM in Ihrem Cluster in die von `token` angegebene Datei. Signieren Sie die Token in der Datei, wenn Sie dazu aufgefordert werden.

Example : Schreiben Sie ein unsigniertes Token pro HSM in Ihr Cluster

```
aws-cloudhsm > user change-mfa token-sign --username cu1 --change-password password --
role crypto-user --token /path/myfile
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:/path/mypemfile
{
  "error_code": 0,
  "data": {
```

```
    "username": "test_user",
    "role": "admin"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<ROLE>

Gibt die Rolle an, die dem Benutzerkonto zugewiesen wurde. Dieser Parameter muss angegeben werden. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter HSM-Benutzer verstehen.

Zulässige Werte

- Admin: Administratoren können Benutzer verwalten, aber sie können keine Schlüssel verwalten.
- Crypto-Benutzer: Crypto-Benutzer können Schlüssel erstellen und verwalten und Schlüssel in kryptografischen Vorgängen verwenden.

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. Bei CloudHSM-CLI-Befehlen wird bei der Rolle und dem Passwort zwischen Groß- und Kleinschreibung unterschieden, beim Benutzernamen jedoch nicht.

Erforderlich: Ja

<CHANGE_PASSWORD>

Gibt das neue Klartext-Passwort des Benutzers an, dessen MFA registriert/deregistriert wird.

Erforderlich: Ja

<TOKEN>

Dateipfad, in den die unsignierte Tokendatei geschrieben wird.

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorum-Benutzerdienstes größer als 1 ist.

<DEREGISTER>

Deregistriert den öffentlichen MFA-Schlüssel, falls vorhanden.

<CHANGE-QUORUM>

Ändert den öffentlichen Quorum-Schlüssel zusammen mit dem MFA-Schlüssel.

Verwandte Themen

- 2FA für HSM-Benutzer verstehen

user change-password

Verwenden Sie den user change-password Befehl in der CloudHSM-CLI, um das Passwort eines vorhandenen Benutzers in Ihrem AWS CloudHSM Cluster zu ändern. Verwenden Sie den `user change-mfa`-Befehl, um MFA für einen Benutzer zu aktivieren.

Jeder Benutzer kann das eigene Passwort ändern. Darüber hinaus können Benutzer mit der Administratorrolle das Passwort eines anderen Benutzers im Cluster ändern. Sie müssen das aktuelle Passwort nicht eingeben, um die Änderung vorzunehmen.

> **Note**
>
> Sie können jedoch nicht das Passwort eines Benutzers ändern, der beim Cluster oder angemeldet ist.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer (Crypto User, CU)

Syntax

> **Note**
>
> Verwenden Sie den user change-mfa-Befehl, um die Multi-Faktor-Authentifizierung (MFA) für einen Benutzer zu aktivieren.

```
aws-cloudhsm > help user change-password
Change a user's password

    Usage:
        cloudhsm-cli user change-password [OPTIONS] --username <USERNAME> --role <ROLE>
 [--password <PASSWORD>]

    Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
          Username of the user that will be modified

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --password <PASSWORD>
          Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

      --deregister-mfa <DEREGISTER-MFA>
          Deregister the user's mfa public key, if present

      --deregister-quorum <DEREGISTER-QUORUM>
```

```
          Deregister the user's quorum public key, if present
 -h, --help
          Print help (see a summary with '-h')
```

Beispiel

Die folgenden Beispiele zeigen, wie Sie mit user change-password das Passwort für den aktuellen Benutzer oder einen anderen Benutzer in Ihrem Cluster zurücksetzen können.

Example : Ändert Ihr Passwort

Jeder Benutzer im Cluster kann mit user change-password sein eigenes Passwort ändern.

Die folgende Ausgabe zeigt, dass Bob derzeit als CU angemeldet ist.

```
aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
    "role": "crypto-user"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorum-Benutzerdienstes größer als 1 ist.

<DEREGISTER-MFA>

Deregistriert den öffentlichen MFA-Schlüssel, falls vorhanden.

<DEREGISTER-QUORUM>

Zum Deregistrieren des öffentlichen Quorum-Schlüssels, falls vorhanden.

<PASSWORD>

Gibt das neue Klartext-Passwort des Benutzers an.

Erforderlich: Ja

<ROLE>

Gibt die Rolle an, die dem Benutzerkonto zugewiesen wurde. Dieser Parameter muss angegeben werden. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter HSM-Benutzer verstehen.

Zulässige Werte

- Admin: Administratoren können Benutzer verwalten, aber sie können keine Schlüssel verwalten.
- Crypto-Benutzer: Crypto-Benutzer können Schlüssel erstellen und verwalten und Schlüssel in kryptografischen Vorgängen verwenden.

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. Bei CloudHSM-CLI-Befehlen wird bei der Rolle und dem Passwort zwischen Groß- und Kleinschreibung unterschieden, beim Benutzernamen jedoch nicht.

Erforderlich: Ja

Verwandte Themen

- user list
- user create
- user delete

user change-quorum

user change-quorum ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit der übergeordneten Kategorie einen Befehl zur Änderung des Quorums für Benutzer erstellen.

user change-quorum wird verwendet, um die Benutzerquorumauthentifizierung mithilfe einer bestimmten Quorumstrategie zu registrieren. Ab SDK 5.8.0 steht Benutzern nur eine einzige Quorumstrategie zur Verfügung, wie unten dargestellt.

Derzeit besteht diese Kategorie aus der folgenden Kategorie und dem folgenden Unterbefehl:

- token-sign
  - register

user change-quorum token-sign

user change-quorum token-sign ist eine übergeordnete Kategorie für Befehle, die in Kombination mit dieser übergeordneten Kategorie einen spezifischen Befehl für Quorumoperationen mit Token-Signaturen erzeugen.

Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- register

user change-quorum token-sign register

Verwenden Sie den user change-quorum token-sign register-Befehl in der CloudHSM-CLI, um die Token-Sign-Quorumstrategie für einen Admin-Benutzer zu registrieren.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

Syntax

```
aws-cloudhsm > help user change-quorum token-sign register
Register a user for quorum authentication with a public key

Usage: user change-quorum token-sign register --public-key <PUBLIC_KEY> --signed-
token <SIGNED_TOKEN>

Options:
```

```
      --cluster-id <CLUSTER_ID>      Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --public-key <PUBLIC_KEY>      Filepath to public key PEM file
      --signed-token <SIGNED_TOKEN>  Filepath with token signed by user private key
  -h, --help Print help (see a summary with '-h')
```

Beispiel

Example

Um diesen Befehl auszuführen, müssen Sie als der Benutzer angemeldet sein, für den Sie register quorum token-sign wollen.

```
aws-cloudhsm > login --username admin1 --role admin
  Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

Der user change-quorum token-sign register-Befehl registriert Ihren öffentlichen Schlüssel beim HSM. Dadurch qualifizieren Sie sich als Quorum-Genehmiger für Quorum-Operationen, bei denen ein Benutzer Quorumsignaturen einholen muss, um den erforderlichen Quorumschwellenwert zu erreichen.

```
aws-cloudhsm > user change-quorum token-sign register \
    --public-key /home/mypemfile
    --signed-token /home/mysignedtoken
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

Sie können jetzt den user list-Befehl ausführen und bestätigen, dass das Quorum-token-sign für diesen Benutzer registriert wurde.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin1",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<PUBLIC-KEY>

Dateipfad zur PEM-Datei mit öffentlichem Schlüssel.

Erforderlich: Ja

<SIGNED-TOKEN>

Dateipfad mit Token, das mit dem privaten Schlüssel des Benutzers signiert wurde.

Erforderlich: Ja

Verwandte Themen

- Verwenden der CloudHSM-CLI zur Verwaltung der Quorum-Authentifizierung
- Quorum-Authentifizierung für Admins verwenden: erstmalige Einrichtung
- Ändern Sie den Quorum-Mindestwert für Administratoren
- Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen

user create

Der user create Befehl in der CloudHSM-CLI erstellt einen Benutzer in Ihrem AWS CloudHSM Cluster. Nur Benutzerkonten mit der Administratorrolle können diesen Befehl ausführen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Admin.

Voraussetzungen

Um diesen Befehl auszuführen, müssen Sie als Admin-Benutzer angemeldet sein

Syntax

```
aws-cloudhsm > help user create
Create a new user

Usage: cloudhsm-cli user create [OPTIONS] --username <USERNAME> --role <ROLE> [--
password <PASSWORD>]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
      --username <USERNAME>
          Username to access the HSM cluster

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --password <PASSWORD>
          Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

  -h, --help
          Print help (see a summary with '-h')
```

Beispiel

Diese Beispiele verdeutlichen, wie mit user create neue Benutzer in Ihren HSMs erstellt werden.

Example : Erstellen eines Crypto-Benutzers

In diesem Beispiel wird in Ihrem AWS CloudHSM Cluster ein Konto mit der Crypto-Benutzerrolle erstellt.

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Gültige Werte sind admin, crypto-user.

Verwenden Sie den user list-Befehl, um die Rolle des Benutzers abzurufen. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter HSM-Benutzer verstehen.

<PASSWORD>

Gibt das Passwort des Benutzers an, der sich bei den HSMs anmeldet.

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorum-Benutzerdienstes größer als 1 ist.

Verwandte Themen

- user list
- user delete
- user change-password

user delete

Der user delete Befehl in der CloudHSM-CLI löscht einen Benutzer aus Ihrem Cluster. AWS CloudHSM Nur Benutzerkonten mit der Administratorrolle dürfen diesen Befehl ausführen. Sie können keinen Benutzer löschen, der derzeit bei einem HSM angemeldet ist.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Admin.

Voraussetzungen

- Sie können keine Benutzerkonten löschen, die Schlüssel besitzen.
- Ihr Benutzerkonto muss über die Administratorrolle verfügen, um diesen Befehl ausführen zu können.

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
aws-cloudhsm > help user delete
Delete a user

Usage: user delete [OPTIONS] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
          Username to access the HSM cluster

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
```

```
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
```

Beispiel

```
aws-cloudhsm > user delete --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<USERNAME>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_). Beim Benutzernamen wird in diesem Befehl nicht zwischen Groß- und Kleinschreibung unterschieden, der Benutzername wird immer in Kleinbuchstaben angezeigt.

Erforderlich: Ja

<ROLE>

Gibt die diesem Benutzer zugewiesene Rolle an. Dieser Parameter muss angegeben werden. Gültige Werte sind admin, crypto-user.

Verwenden Sie den user list-Befehl, um die Rolle des Benutzers abzurufen. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter HSM-Benutzer verstehen.

Erforderlich: Ja

<APPROVAL>

Gibt den Dateipfad zu einer signierten Quorum-Token-Datei an, um den Vorgang zu genehmigen. Nur erforderlich, wenn der Quorumwert des Quorum-Benutzerdienstes größer als 1 ist.

Erforderlich: Ja

Verwandte Themen

- user list
- user create
- user change-password

user list

Der user list-Befehl in der CloudHSM-CLI listet die Benutzerkonten auf, die in Ihrem CloudHSM-Cluster vorhanden sind. Sie müssen nicht bei CloudHSM-CLI angemeldet sein, um diesen Befehl auszuführen.

> **Note**
>
> Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien, die der AWS CloudHSM Client und die Befehlszeilentools verwenden. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

```
aws-cloudhsm > help user list
List the users in your cluster

USAGE:
    user list
```

```
Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

Beispiel

Dieser Befehl listet die Benutzer auf, die in Ihrem CloudHSM-Cluster vorhanden sind.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "test_user",
        "role": "admin",
        "locked": "false",
        "mfa": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
```

```
}
```

Die Ausgabe umfasst die folgenden Benutzerattribute:

- Benutzername: Zeigt den benutzerdefinierten Anzeigenamen für den Benutzer an. Der Benutzername wird immer in Kleinbuchstaben angezeigt.
- Benutzertyp: Bestimmt die Operationen, die der Benutzer im HSM ausführen kann.
- Gesperrt: Zeigt an, ob dieses Benutzerkonto gesperrt wurde.
- MFA: Gibt die unterstützten Multi-Faktor-Authentifizierungsmechanismen für dieses Benutzerkonto an.
- Clusterabdeckung: Zeigt die clusterweite Verfügbarkeit dieses Benutzerkontos an.

Verwandte Themen

- listUsers in key_mgmt_util
- user create
- user delete
- user change-password

quorum

quorum ist eine übergeordnete Kategorie für eine Gruppe von Befehlen, die in Kombination mit quorum einen spezifischen Befehl für die Quorumauthentifizierung oder M-of-N-Vorgänge erzeugen. Derzeit besteht diese Kategorie aus der token-sign-Unterkategorie, die aus eigenen Befehlen besteht. Weitere Informationen finden Sie unter dem nachfolgenden Link.

- token-sign

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Jeder Diensttyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Servicetyp	Serviceoperationen
user	Admin.	• user create • user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Token-Zeichen set-quorum-value

Verwandte Themen

- Quorum-Authentifizierung für Admins verwenden: erstmalige Einrichtung
- Verwendung von CloudHSM CLI zur Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle)

quorum token-sign

quorum token-sign ist eine Kategorie für eine Gruppe von Befehlen, die in Kombination mit quorum token-sign einen spezifischen Befehl für die Quorumauthentifizierung oder M-of-N-Operationen erzeugen.

Derzeit besteht diese Kategorie aus den folgenden Befehlen:

- delete
- generate
- auflisten
- list-quorum-values
- list-timeouts
- set-quorum-value
- set-timeout

quorum token-sign delete

Verwenden Sie den quorum token-sign delete-Befehl in der CloudHSM-CLI, um ein oder mehrere Token für einen vom Quorum autorisierten Dienst zu löschen.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

Syntax

```
aws-cloudhsm > help quorum token-sign delete
Delete one or more Quorum Tokens

Usage: quorum token-sign delete --scope <SCOPE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --scope <SCOPE>
          Scope of which token(s) will be deleted

          Possible values:
          - user: Deletes all token(s) of currently logged in user
          - all:  Deletes all token(s) on the HSM
  -h, --help
          Print help (see a summary with '-h')
```

Beispiel

Das folgende Beispiel zeigt, wie der quorum token-sign delete-Befehl in der CloudHSM-CLI verwendet werden kann, um ein oder mehrere Token für einen vom Quorum autorisierten Dienst zu löschen.

Example : Löschen Sie ein oder mehrere Token für einen vom Quorum autorisierten Dienst

```
aws-cloudhsm > quorum token-sign delete --scope all
```

```
{
  "error_code": 0,
  "data": "Deletion of quorum token(s) successful"
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<SCOPE>

Der Bereich, in dem Token im AWS CloudHSM Cluster gelöscht werden.

Zulässige Werte

- Benutzer: Wird verwendet, um nur Token zu löschen, die dem angemeldeten Benutzer gehören.
- Alle: Wird verwendet, um alle Token im AWS CloudHSM Cluster zu löschen.

Verwandte Themen

- user list
- user create
- user delete

quorum token-sign generate

Verwenden Sie den quorum token-sign generate-Befehl in der CloudHSM-CLI, um ein Token für einen vom Quorum autorisierten Dienst zu generieren.

In einem HSM-Cluster für Dienstbenutzer und Quorum gibt es eine Obergrenze für den Erhalt eines aktiven Tokens pro Benutzer und Dienst.

> **Note**
>
> Nur Administratoren können ein Service-Token generieren.

Admin-Dienste: Die Quorum-Authentifizierung wird für Dienste mit Administratorrechten verwendet, z. B. für das Erstellen von Benutzern, das Löschen von Benutzern, das Ändern von Benutzerkennwörtern, das Festlegen von Quorumwerten und das Deaktivieren von Quorum- und MFA-Funktionen.

Jeder Diensttyp wird weiter in einen qualifizierenden Dienstnamen unterteilt, der eine bestimmte Gruppe von Quorum-unterstützten Dienstvorgängen enthält, die ausgeführt werden können.

Service-Name	Servicetyp	Serviceoperationen
user	Admin.	• user create • user delete • user change-password • user change-mfa
quorum	Admin.	• Quorum-Token-Zeichen set-quorum-value

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer (Crypto User, CU)

Syntax

```
aws-cloudhsm > help quorum token-sign generate
Generate a token

Usage: quorum token-sign generate --service <SERVICE> --token <TOKEN>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
      --service <SERVICE>
          Service the token will be used for

          Possible values:
          - user:
            User management service is used for executing quorum authenticated user
 management operations
          - quorum:
            Quorum management service is used for setting quorum values for any quorum
 service
          - registration:
            Registration service is used for registering a public key for quorum
 authentication

      --token <TOKEN>
          Filepath where the unsigned token file will be written
  -h, --help                     Print help
```

Beispiel

Dieser Befehl schreibt ein unsigniertes Token pro HSM in Ihrem Cluster in die von `token` angegebene Datei.

Example : Schreiben Sie ein unsigniertes Token pro HSM in Ihr Cluster

```
aws-cloudhsm > quorum token-sign generate --service user --token /home/tfile
{
  "error_code": 0,
  "data": {
    "filepath": "/home/tfile"
  }
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<SERVICE>

Gibt den vom Quorum autorisierten Dienst an, für den ein Token generiert werden soll Dieser Parameter muss angegeben werden.

Zulässige Werte

- Benutzer: Der Benutzerverwaltungsdienst, der für die Ausführung von Vorgängen zur Verwaltung autorisierter Quorum-Benutzer verwendet wird.
- Quorum: Der Quorumverwaltungsdienst, der zum Festlegen von Quorum-autorisierten Quorumwerten für jeden vom Quorum autorisierten Dienst verwendet wird.
- Registrierung: Generiert ein unsigniertes Token zur Registrierung eines öffentlichen Schlüssels für die Quorumautorisierung.

Erforderlich: Ja

<TOKEN>

Dateipfad, in den die unsignierte Tokendatei geschrieben wird.

Erforderlich: Ja

Verwandte Themen

- Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen

quorum token-sign list

Verwenden Sie den quorum token-sign list Befehl in der CloudHSM-CLI, um alle Tokensign-Quorum-Token aufzulisten, die in Ihrem Cluster vorhanden sind. AWS CloudHSM

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.
- Crypto-Benutzer (Crypto User, CU)

Syntax

```
aws-cloudhsm > help quorum token-sign list
```

```
List the token-sign tokens in your cluster

Usage: quorum token-sign list

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

Beispiel

Dieser Befehl listet alle Tokensign-Token auf, die in Ihrem Cluster vorhanden sind. AWS CloudHSM

Example

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
      {
        "username": "admin",
        "service": "quorum",
        "approvals-required": 2
        "number-of-approvals": 0
        "token-timeout-seconds": 397
        "cluster-coverage": "full"
      },
      {
        "username": "admin",
        "service": "user",
        "approvals-required": 2
        "number-of-approvals": 2
        "token-timeout-seconds": 588
        "cluster-coverage": "full"
      }
    ]
  }
}
```

Verwandte Themen

- quorum token-sign generate

Quorum-Token-Zeichen list-quorum-values

Verwenden Sie den quorum token-sign list-quorum-values Befehl in der CloudHSM-CLI, um die in Ihrem Cluster festgelegten Quorumwerte aufzulisten. AWS CloudHSM

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

```
aws-cloudhsm > help quorum token-sign list-quorum-values
List current quorum values

Usage: quorum token-sign list-quorum-values

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

Beispiel

Dieser Befehl listet die in Ihrem AWS CloudHSM Cluster für jeden Service festgelegten Quorumwerte auf.

Example

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 1
```

```
  }
}
```

Verwandte Themen

- Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen

quorum token-sign list-timeouts

Verwenden Sie den quorum token-sign list-timeouts-Befehl in der CloudHSM-CLI, um den Token-Timeout-Zeitraum in Sekunden für alle Tokentypen abzurufen.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

```
aws-cloudhsm > help quorum token-sign list-timeouts
List timeout durations in seconds for token validity

Usage: quorum token-sign list-timeouts

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

Beispiel

Example

```
aws-cloudhsm > quorum token-sign list-timeouts
{
  "error_code": 0,
  "data": {
    "generated": 600,
```

```
    "approved": 600
  }
}
```

Die Ausgabe enthält Folgendes.

- generiert: Timeout-Zeitraum in Sekunden für die Genehmigung eines generierten Tokens.
- genehmigt: Timeout-Zeitraum in Sekunden für die Verwendung eines genehmigten Tokens zur Ausführung eines vom Quorum autorisierten Vorgangs.

Verwandte Themen

- quorum token-sign set-timeout

Quorum-Token-Zeichen set-quorum-value

Verwenden Sie den quorum token-sign set-quorum-value-Befehl in der CloudHSM-CLI, um einen neuen Quorumwert für einen autorisierten Quorum-Dienst festzulegen.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

Syntax

```
aws-cloudhsm > help quorum token-sign set-quorum-value
Set a quorum value

Usage: quorum token-sign set-quorum-value [OPTIONS] --service <SERVICE> --value <VALUE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --service <SERVICE>
          Service the token will be used for
```

```
          Possible values:
          - user:
            User management service is used for executing quorum authenticated user
 management operations
          - quorum:
            Quorum management service is used for setting quorum values for any quorum
 service

      --value <VALUE>
          Value to set for service

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

  -h, --help
          Print help (see a summary with '-h')
```

Beispiel

Example

Im folgenden Beispiel schreibt dieser Befehl ein unsigniertes Token pro HSM in Ihr Cluster in die durch das Token angegebene Datei. Signieren Sie die Token in der Datei, wenn Sie dazu aufgefordert werden.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2
{
  "error_code": 0,
  "data": "Set Quorum Value successful"
}
```

Anschließend können Sie den list-quorum-values-Befehl ausführen, um zu bestätigen, dass der Quorumwert für den Quorumverwaltungsdienst festgelegt wurde:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 2
  }
```

```
}
```

Argumente

<CLUSTER_ID>

Die ID des Clusters, auf dem dieser Vorgang ausgeführt werden soll.

Erforderlich: Wenn mehrere Cluster konfiguriert wurden.

<APPROVAL>

Der Dateipfad der signierten Tokendatei, die auf dem HSM genehmigt werden soll.

<SERVICE>

Gibt den vom Quorum autorisierten Dienst an, für den ein Token generiert werden soll Dieser Parameter muss angegeben werden. Weitere Informationen zu Diensttypen und -namen finden Sie unter Dienstnamen und -typen, die die Quorumauthentifizierung unterstützen.

Zulässige Werte

- Benutzer: Der Benutzerverwaltungsdienst. Dienst, der für die Ausführung autorisierter Quorum-Benutzerverwaltungsvorgänge verwendet wird.
- Quorum: Der Quorum-Verwaltungsdienst. Dienst, der zum Festlegen von Quorum-autorisierten Quorumwerten für jeden vom Quorum autorisierten Dienst verwendet wird.
- Registrierung: Generiert ein unsigniertes Token zur Registrierung eines öffentlichen Schlüssels für die Quorumautorisierung.

Erforderlich: Ja

<VALUE>

Gibt den Quorumwert an, der festgelegt werden soll. Der maximale Quorumwert ist acht (8).

Erforderlich: Ja

Verwandte Themen

- Quorum-Tokenzeichen list-quorum-values
- Namen und Typen von Diensten, die die Quorum-Authentifizierung unterstützen

quorum token-sign set-timeout

Verwenden Sie den quorum token-sign set-timeout-Befehl in der CloudHSM-CLI, um das Token-Timeout in Sekunden für jeden Tokentyp festzulegen.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Admin.

Syntax

```
aws-cloudhsm > help quorum token-sign set-timeout
Set timeout duration in seconds for token validity

Usage: quorum token-sign set-timeout <--generated <GENERATED> |--approved <APPROVED>>

Options:
      --cluster-id <CLUSTER_ID>   Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --generated <GENERATED>     Timeout period in seconds for a generated (non-
approved) token to be approved
      --approved <APPROVED>      Timeout period in seconds for an approved token to be
 used to execute a quorum operation
  -h, --help    Print help (see a summary with '-h')
```

Beispiel

Die folgenden Beispiele zeigen, wie Sie über den quorum token-sign set-timeout-Befehl zum Festlegen des Token-Timeout-Zeitraums verwenden.

```
aws-cloudhsm > quorum token-sign set-timeout --generated 900
{
  "error_code": 0,
  "data": "Set token timeout successful"
}
```

Verwandte Themen

- quorum token-sign list-timeouts

CloudHSM-Verwaltungsdienstprogramm (CMU)

Das cloudhsm_mgmt_util-Befehlszeilentool hilft Crypto Officers bei der Verwaltung von Benutzern in HSMs. Es umfasst Tools zum Erstellen, Löschen und Auflisten von Benutzern sowie zum Ändern von Benutzerpasswörtern.

KMU und CMU sind Teil der Client-SDK-3-Suite.

cloudhsm_mgmt_util umfasst auch Befehle, mit denen Crypto-Benutzer (CUs) Schlüssel freigeben sowie Schlüsselattribute abrufen und festlegen können. Diese Befehle ergänzen die Schlüsselverwaltungsbefehle im Primärschlüssel-Verwaltungstool, key_mgmt_util.

Informationen zum schnellen Einstieg finden Sie unter Verwalten von geklonten Clustern. Ausführliche Informationen über die cloudhsm_mgmt_util-Befehle und Beispiele für die Verwendung der Befehle finden Sie unter cloudhsm_mgmt_util-Befehlsreferenz.

Themen

Unterstützte Plattformen für AWS CloudHSM-Verwaltungsdienstprogramme

Linux-Support

- Amazon Linux
- Amazon Linux 2
- CentOS 6.10+
- CentOS 7.3+
- CentOS 8
- Red Hat Enterprise Linux (RHEL) 6.10+
- Red Hat Enterprise Linux (RHEL) 7.9+

- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 16.04 LTS
- Ubuntu 18.04 LTS

Windows-Unterstützung

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Erste Schritte mit CloudHSM Management Utility (CMU)

Mit dem CloudHSM Management Utility (CMU) können Sie Benutzer des Hardware Security Module (HSM) verwalten. Erfahren Sie in diesem Thema, um mit grundlegenden HSM-Benutzerverwaltungsaufgaben wie dem Erstellen von Benutzern, dem Auflisten von Benutzern und dem Verbinden der CMU mit dem Cluster zu beginnen.

1. Um CMU verwenden zu können, müssen Sie zunächst das Konfigurationstool verwenden, um die lokale CMU-Konfiguration mit dem `--cmu`-Parameter und einer IP-Adresse von einem der HSMs in Ihrem Cluster zu aktualisieren. Tun Sie dies jedes Mal, wenn Sie CMU verwenden, um sicherzustellen, dass Sie HSM-Benutzer auf allen HSM im Cluster verwalten.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Verwenden Sie den folgenden Befehl, um die CLI im interaktiven Modus zu starten.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

Windows

```
C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

Die Ausgabe sollte in Abhängigkeit von der Anzahl vorliegender HSMs der folgenden ähneln.

```
Connecting to the server(s), it may take time
depending on the server(s) load, please wait...

Connecting to server '10.0.2.9': hostname '10.0.2.9', port 2225...
Connected to server '10.0.2.9': hostname '10.0.2.9', port 2225.

Connecting to server '10.0.3.11': hostname '10.0.3.11', port 2225...
Connected to server '10.0.3.11': hostname '10.0.3.11', port 2225.

Connecting to server '10.0.1.12': hostname '10.0.1.12', port 2225...
Connected to server '10.0.1.12': hostname '10.0.1.12', port 2225.
```

Wenn cloudhsm_mgmt_util ausgeführt wird, ändert sich die Eingabeaufforderung in `aws-cloudhsm>`.

3. Verwenden Sie den loginHSM-Befehl, um sich beim Cluster anzumelden. Jeder Benutzertyp kann sich mit diesem Befehl beim Cluster anmelden.

   Mit dem Befehl im folgenden Beispiel wird admin angemeldet, der Standard-Verschlüsselungsverantwortliche (CO, Crypto Officer). Sie legen das Passwort für diesen Benutzer nach der Aktivierung des Clusters fest. Sie können den `-hpswd`-Parameter verwenden, um Ihr Passwort zu verbergen.

   ```
   aws-cloudhsm>loginHSM CO admin -hpswd
   ```

   Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Sie geben das Passwort ein, das System versteckt das Passwort und die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie eine Verbindung zu allen HSMs auf dem Cluster hergestellt haben.

   ```
   Enter password:
   ```

```
loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)
loginHSM success on server 2(10.0.1.12)
```

4. Verwenden Sie listUsers, um alle Benutzer im Cluster aufzulisten.

```
aws-cloudhsm>listUsers
```

CMU listet alle Benutzer des Clusters auf.

```
Users on server 0(10.0.2.9):
Number of users found:2

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                NO
               0               NO
         2              AU              app_user                             NO
               0               NO
Users on server 1(10.0.3.11):
Number of users found:2

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                NO
               0               NO
         2              AU              app_user                             NO
               0               NO
Users on server 2(10.0.1.12):
Number of users found:2

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                NO
               0               NO
         2              AU              app_user                             NO
               0               NO
```

5. Verwenden Sie createUser, um einen CU-Benutzer namens **example_user** mit dem Passwort **password1** zu erstellen.

Sie verwenden CU-Benutzer in Ihren Anwendungen, um kryptografische und Schlüsselverwaltungsvorgänge durchzuführen. Sie können CU-Benutzer erstellen, da Sie sich in Schritt 3 als CO-Benutzer angemeldet haben. Nur CO-Benutzer können mit dem CMU Benutzerverwaltungsaufgaben durchführen, wie das Anlegen und Löschen von Benutzern und das Ändern der Passwörter anderer Benutzer.

```
aws-cloudhsm>createUser CU example_user password1
```

Die CMU fordert Sie auf, den Vorgang zum Erstellen eines Benutzers auszuführen.

```
*************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
***************************************************************

Do you want to continue(y/n)?
```

6. Geben Sie **y** ein, um den CU-Benutzer **example_user** zu erstellen.
7. Verwenden Sie listUsers, um alle Benutzer im Cluster aufzulisten.

```
aws-cloudhsm>listUsers
```

CMU listet alle Benutzer im Cluster auf, einschließlich des neuen CU-Benutzers, den Sie gerade erstellt haben.

```
Users on server 0(10.0.2.9):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                NO
               0               NO
         2              AU              app_user                             NO
               0               NO
```

```
         3              CU              example_user                              NO
               0               NO
Users on server 1(10.0.3.11):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                     NO
               0               NO
         2              AU              app_user                                  NO
               0               NO
         3              CU              example_user                              NO
               0               NO
Users on server 2(10.0.1.12):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                     NO
               0               NO
         2              AU              app_user                                  NO
               0               NO
         3              CU              example_user                              NO
               0               NO
```

8. Verwenden Sie den Befehl logoutHSM, um sich von den HSMs abzumelden.

```
aws-cloudhsm>logoutHSM


logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
logoutHSM success on server 2(10.0.1.12)
```

9. Verwenden Sie den quit-Befehl, um cloudhsm_mgmt_util zu beenden.

```
aws-cloudhsm>quit


disconnecting from servers, please wait...
```

Den AWS CloudHSM Client installieren und konfigurieren (Linux)

Um mit dem HSM in Ihrem AWS CloudHSM Cluster zu interagieren, benötigen Sie die AWS CloudHSM Client-Software für Linux. Sie sollten die Software auf der Linux-EC2-Client-Instance installieren, die Sie zuvor erstellt haben. Sie können auch einen Client für Windows installieren. Weitere Informationen finden Sie unter Installieren und Konfigurieren des AWS CloudHSM-Clients (Windows).

Aufgaben

- Installieren Sie den AWS CloudHSM Client und die Befehlszeilentools
- Bearbeiten der Clientkonfiguration

Installieren Sie den AWS CloudHSM Client und die Befehlszeilentools

Connect zu Ihrer Client-Instance her und führen Sie die folgenden Befehle aus, um den AWS CloudHSM Client und die Befehlszeilentools herunterzuladen und zu installieren.

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb
```

Bearbeiten der Clientkonfiguration

Bevor Sie den AWS CloudHSM Client verwenden können, um eine Verbindung zu Ihrem Cluster herzustellen, müssen Sie die Client-Konfiguration bearbeiten.

Bearbeiten der Clientkonfiguration

1. Wenn Sie Client-SDK 3 auf cloudhsm_mgmt_util installieren, führen Sie die folgenden Schritte aus, um sicherzustellen, dass alle Knoten im Cluster synchronisiert sind.

   a. Führen Sie configure -a ***<IP of one of the HSMs>***.

   b. Starten Sie den Client-Service neu.

   c. Führen Sie config -m.
2. Kopieren Sie Ihr ausstellendes Zertifikat – das Zertifikat, mit dem Sie das Zertifikat des Clusters signiert haben – an den folgenden Speicherort in der Client-Instance: `/opt/cloudhsm/etc/customerCA.crt`. Sie müssen Root-Berechtigungen für die Client-Instance haben, um Ihr Zertifikat an diesen Speicherort zu kopieren.
3. Verwenden Sie den folgenden Konfigurationsbefehl, um die Konfigurationsdateien für den AWS CloudHSM Client und die Befehlszeilentools zu aktualisieren. Geben Sie dabei die IP-Adresse des HSM in Ihrem Cluster an. Um die IP-Adresse des HSM abzurufen, zeigen Sie Ihren Cluster in der AWS CloudHSM Konsole an oder führen Sie den describe-clusters CLI-Befehl aus. Die IP-Adresse des HSM ist in der Ausgabe des Befehls der Wert des Feldes `EniIp`. Wenn Sie über mehr als ein HSM verfügen, wählen Sie einfach eine die IP-Adresse eines beliebigen HSM aus.

   ```
   sudo /opt/cloudhsm/bin/configure -a <IP address>

   Updating server config in /opt/cloudhsm/etc/cloudhsm_client.cfg
   Updating server config in /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```
4. Wechseln Sie zu Aktivieren des Clusters.

Installieren und Konfigurieren des AWS CloudHSM-Clients (Windows)

Um mit einem HSM in Ihrem AWS CloudHSM-Cluster unter Windows zu arbeiten, benötigen Sie die AWS CloudHSM-Clientsoftware für Windows. Sie sollten die Software auf der Windows Server-Instance installieren, die Sie zuvor erstellt haben.

So installieren (oder aktualisieren) Sie die den neuesten Windows-Client und die Befehlszeilen-Tools

1. Stellen Sie eine Verbindung zur Windows Server-Instance her.
2. Laden Sie das AWSCloudHSMClient-latest.msi-Installationsprogramm herunter.
3. Wenn Sie Client-SDK 3 auf cloudhsm_mgmt_util installieren, führen Sie die folgenden Schritte aus, um sicherzustellen, dass alle Knoten im Cluster synchronisiert sind.
   a. Führen Sie configure -a ***<IP of one of the HSMs>***.
   b. Starten Sie den Client-Service neu.
   c. Führen Sie config -m.
4. Gehen Sie zu Ihrem Download-Ort und führen Sie das Installationsprogramm (AWSCloudHSMClient-latest.msi) mit administrativen Rechten aus.
5. Folgen Sie den Anweisungen des Installationsprogramms und wählen Sie dann Schließen, wenn das Installationsprogramm abgeschlossen ist.
6. Kopieren Sie das selbstsignierte Ausstellerzertifikat – das Zertifikat, mit dem Sie das Clusterzertifikat signiert haben – in den Ordner `C:\ProgramData\Amazon\CloudHSM`.
7. Führen Sie den folgenden Befehl aus, um die Konfigurationsdateien zu aktualisieren. Beim Aktualisieren muss der Client während der Neukonfiguration angehalten und dann wieder gestartet werden.

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -a <HSM IP address>
   ```

8. Wechseln Sie zu Aktivieren des Clusters.

Hinweise:

- Bei der Aktualisierung des Clients werden vorhandene Konfigurationsdateien aus früheren Installationen nicht überschrieben.
- Das AWS CloudHSM-Client-Installationsprogramm für Windows registriert automatisch die Cryptography API: Next Generation (CNG) und den Key Storage Provider (KSP). Um den Client zu

deinstallieren, führen Sie das Installationsprogramm erneut aus und folgen Sie den Anweisungen zur Deinstallation.

- Wenn Sie Linux verwenden, können Sie den Linux-Client installieren. Weitere Informationen finden Sie unter Den AWS CloudHSM Client installieren und konfigurieren (Linux).

cloudhsm_mgmt_util-Befehlsreferenz

Das cloudhsm_mgmt_util-Befehlszeilentool hilft Crypto Officers bei der Verwaltung von Benutzern in HSMs. Es umfasst auch Befehle, mit denen Crypto-Benutzer (CUs, Crypto Users) Schlüssel freigeben sowie Schlüsselattribute abrufen und festlegen können. Diese Befehle ergänzen die Primärschlüsselverwaltungsbefehle im key_mgmt_util-Befehlszeilen-Tool.

Informationen zum schnellen Einstieg finden Sie unter Verwalten von geklonten Clustern.

Bevor Sie einen cloudhsm_mgmt_util-Befehl ausführen, müssen Sie cloudhsm_mgmt_util starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzerkontotyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Um alle cloudhsm_mgmt_util-Befehle aufzuführen, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm> help
```

Um die Syntax für einen cloudhsm_mgmt_util-Befehl anzufordern, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm> help <command-name>
```

> **Note**
>
> Verwenden Sie die Syntax gemäß der Dokumentation. Die integrierte Softwarehilfe bietet zwar zusätzliche Optionen, diese sollten jedoch nicht als unterstützt betrachtet und nicht im Produktionscode verwendet werden.

Um einen Befehl auszuführen, geben Sie den Namen des Befehls ein, oder einen Teil des Namens, der ausreicht, um ihn von den Namen anderer cloudhsm_mgmt_util-Befehle zu unterscheiden.

Wenn Sie beispielsweise eine Liste von Benutzern der HSMs abrufen möchten, geben Sie listUsers oder listU ein.

```
aws-cloudhsm> listUsers
```

Um Ihre cloudhsm_mgmt_util-Sitzung zu beenden, führen Sie den folgenden Befehl aus:

```
aws-cloudhsm> quit
```

Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Die folgenden Themen beschreiben Befehle in cloudhsm_mgmt_util.

> **Note**
>
> Einige Befehle in key_mgmt_util und cloudhsm_mgmt_util haben dieselben Namen. Die Befehle haben jedoch in der Regel eine andere Syntax, eine andere Ausgabe und eine leicht unterschiedliche Funktionalität.

Befehl	Beschreibung	Benutzertyp
changePswd	Ändert die Passwörter von Benutzern der HSMs. Jeder Benutzer kann das eigene Passwort ändern. COs können das Passwort jedes Benutzers ändern.	CO
createUser	Erstellt Benutzer aller Typen in den HSMs.	CO
deleteUser	Löscht Benutzer aller Typen aus den HSMs.	CO
findAllKeys	Ruft die Schlüssel ab, die ein Benutzer besitzt oder freigibt. Außerdem wird ein Hash der Schlüsseleigentümerschaft und Freigabedaten für	CO, AU

Befehl	Beschreibung	Benutzertyp
	alle Schlüssel in jedem HSM abgerufen.	
getAttribute	Ruft einen Attributwert für einen AWS CloudHSM-Schlüssel ab und schreibt ihn in eine Datei oder in die stdout (Standardausgabe).	CU
getCert	Ruft das Zertifikat eines bestimmten HSM ab und speichert es in einem gewünschten Zertifikatformat.	Alle.
getHSMInfo	Ruft Informationen über die Hardware ab, auf der ein HSM ausgeführt wird.	Alle. Eine Anmeldung ist nicht erforderlich.
getKeyInfo	Ruft Eigentümer, gemeinsame Benutzer und den Quorum-Authentifizierungsstatus eines Schlüssels ab.	Alle. Eine Anmeldung ist nicht erforderlich.
info	Ruft Informationen zu einem HSM ab, einschließlich IP-Adresse, Hostname, Port und aktuellem Benutzer.	Alle. Eine Anmeldung ist nicht erforderlich.
listUsers	Ruft die Benutzer der einzelnen HSMs ab, deren Benutzertyp und ID sowie andere Attribute.	Alle. Eine Anmeldung ist nicht erforderlich.
loginHSM und logoutHSM	An- und Abmeldung bei einem HSM.	Alle.

Befehl	Beschreibung	Benutzertyp
quit	Beendet cloudhsm_mgmt_util.	Alle. Eine Anmeldung ist nicht erforderlich.
server	Dient zum Aufrufen und Beenden des Servermodus auf einem HSM.	Alle.
registerQuorumPubKey	Ordnet einen HSM-Benutzer einem asymmetrischen RSA-2048-Schlüsselpaar zu.	CO
setAttribute	Ändert die Werte der Attribute zum Bezeichnen, Verschlüsseln, Entschlüsseln, Verpacken und Entpacken eines vorhandenen Schlüssels.	CU
shareKey	Gibt einen vorhandenen Schlüssel für andere Benutzer frei.	CU
syncKey	Synchronisiert einen Schlüssel für geklonte AWS CloudHSM-Cluster.	CU, CO
syncUser	Synchronisiert einen Benutzer über geklonte AWS CloudHSM-Cluster hinweg.	CO

changePswd

Der Befehl changePswd in cloudhsm_mgmt_util ändert das Passwort eines existierenden Benutzers auf den HSMs im Cluster.

Jeder Benutzer kann das eigene Passwort ändern. Darüber hinaus können Crypto Officers (COs und PCOs) das Passwort eines anderen COs oder Crypto-Benutzers (CU) ändern. Sie müssen das aktuelle Passwort nicht eingeben, um die Änderung vorzunehmen.

> **Note**
>
> Sie können jedoch nicht das Passwort eines Benutzers ändern, der bei AWS CloudHSM-Client oder key_mgmt_util angemeldet ist.

Um changePswd zu beheben

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (Crypto Officers, CO)
- Crypto-Benutzer (Crypto User, CU)

Syntax

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um die Zwei-Faktor-Authentifizierung (2FA) für einen CO-Benutzer zu aktivieren, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen finden Sie unter the section called “Argumente”.

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

Beispiele

Die folgenden Beispiele zeigen, wie Sie mit changePassword das Passwort für den aktuellen Benutzer oder einen anderen Benutzer in Ihren HSMs zurücksetzen können.

Example : Ändert Ihr Passwort

Jeder Benutzer in den HSMs kann mit changePswd sein eigenes Passwort ändern. Bevor Sie das Passwort ändern, verwenden Sie info, um Informationen über jeden der HSMs im Cluster zu erhalten - einschließlich des Benutzernamens und des Benutzertyps des angemeldeten Benutzers.

Die folgende Ausgabe zeigt, dass Bob derzeit als CU angemeldet ist.

```
        aws-cloudhsm> info server 0

Id      Name                    Hostname         Port   State           Partition
  LoginState
0       10.1.9.193              10.1.9.193      2225    Connected       hsm-jqici4covtv
  Logged in as 'bob(CU)'

aws-cloudhsm> info server 1

Id      Name                    Hostname         Port   State           Partition
  LoginState
1       10.1.10.7               10.1.10.7       2225    Connected       hsm-ogi3sywxbqx
  Logged in as 'bob(CU)'
```

Um das Passwort zu ändern, führt Bob changePswd aus, gefolgt von dem Benutzertyp, dem Benutzernamen und einem neuen Passwort.

```
aws-cloudhsm> changePswd CU bob newPassword

**************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
******************************************************************

Do you want to continue(y/n)?y
Changing password for bob(CU) on 2 nodes
```

Example : Ändert das Passwort eines anderen Benutzers

Sie müssen ein CO oder PCO sein, um das Passwort eines anderen CO oder CU auf den HSMs zu ändern. Bevor Sie das Passwort für einen anderen Benutzer ändern, prüfen Sie mit dem Befehl info, dass Ihr Benutzertyp entweder CO oder PCO ist.

Die folgende Ausgabe bestätigt, dass Alice eine CO und derzeit angemeldet ist.

```
aws-cloudhsm>info server 0

Id      Name             Hostname         Port   State           Partition
 LoginState
0       10.1.9.193       10.1.9.193        2225   Connected      hsm-jqici4covtv
 Logged in as 'alice(CO)'


aws-cloudhsm>info server 1

Id      Name             Hostname         Port   State           Partition
 LoginState
0       10.1.10.7        10.1.10.7        2225   Connected       hsm-ogi3sywxbqx
 Logged in as 'alice(CO)'
```

Alice will das Passwort eines anderen Benutzers (John) zurücksetzen. Bevor sie das Passwort ändert, überprüft sie mit dem Befehl listUsers den Benutzertyp von John.

Die folgende Ausgabe listet John als CO-Benutzer auf.

```
aws-cloudhsm> listUsers
Users on server 0(10.1.9.193):
Number of users found:5

    User Id             User Type       User Name            MofnPubKey
 LoginFailureCnt         2FA
        1               PCO             admin                   YES              0
            NO
        2               AU              jane                     NO              0
            NO
        3               CU              bob                      NO              0
            NO
```

```
        4               CU              alice                   NO              0
            NO
        5               CO              john                    NO              0
            NO
Users on server 1(10.1.10.7):
Number of users found:5

    User Id             User Type       User Name           MofnPubKey
 LoginFailureCnt         2FA
        1               PCO             admin                   YES             0
            NO
        2               AU              jane                    NO              0
            NO
        3               CU              bob                     NO              0
            NO
        4               CO              alice                   NO              0
            NO
        5               CO              john                    NO              0
            NO
```

Um das Passwort zu ändern, führt Alice changePswd aus, gefolgt von Johns Benutzertyp, Benutzername und einem neuen Passwort.

```
aws-cloudhsm>changePswd CO john newPassword

*************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
Changing password for john(CO) on 2 nodes
```

Argumente

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um 2FA für einen CO-Benutzer zu aktivieren, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen zur Arbeit mit 2FA finden Sie unter Verwendung von CMU zur Verwaltung von 2FA.

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

Gibt den aktuellen Typ des Benutzers an, dessen Passwort Sie ändern. Sie können changePswd nicht verwenden, um den Benutzertyp zu ändern.

Gültige Werte sind `CO`, `CU`, `PCO` und `PRECO`.

Zum Abrufen des Benutzertyps verwenden Sie den Befehl listUsers. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter Grundlegendes zu HSM-Benutzern.

Erforderlich: Ja

<user-name>

Gibt den Anzeigenamen des Benutzers an. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden. Sie können changePswd nicht verwenden, um den Benutzernamen zu ändern.

Erforderlich: Ja

<password | -hpswd >

Gibt ein neues Passwort für den Benutzer an. Geben Sie eine Zeichenfolge von 7 bis 32 Zeichen ein. Bei diesem Wert ist die Groß- und Kleinschreibung zu beachten. Das Passwort wird in Klartext angezeigt, wenn Sie es eingeben. Um Ihr Passwort zu verbergen, verwenden Sie den `-hpswd`-Parameter anstelle des Passworts und folgen Sie den Aufforderungen.

Erforderlich: Ja

[-2fa </path/to/authdata>]

Gibt an, dass 2FA für diesen CO-Benutzer aktiviert wird. Um die für die Einrichtung von 2FA erforderlichen Daten abzurufen, fügen Sie einen Pfad zu einem Speicherort im Dateisystem mit einem Dateinamen nach dem `-2fa`-Parameter ein. Weitere Informationen zur Arbeit mit 2FA finden Sie unter Verwendung von CMU zur Verwaltung von 2FA.

Erforderlich: Nein

Verwandte Themen

- info

- listUsers
- createUser
- deleteUser

createUser

Der createUser-Befehl in cloudhsm_mgmt_util erstellt einen Benutzer auf den HSMs. Nur Crypto Officers (COs und PRECOs) können diesen Befehl ausführen. Wenn der Befehl erfolgreich ist, wird der Benutzer in allen HSMs im Cluster erstellt.

Um Fehler bei createUser zu beheben

Wenn Ihre HSM-Konfiguration ungenau ist, wird der Benutzer möglicherweise nicht in allen HSMs erstellt. Um den Benutzer den HSMs hinzuzufügen, in denen er fehlt, verwenden Sie den Befehl syncUser oder createUser nur in den HSMs, in denen dieser Benutzer fehlt. Um Konfigurationsfehler zu meiden, führen Sie das configure-Tool mit der `-m`-Option aus.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto Officers (CO, PRECO)

Syntax

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um einen CO-Benutzer mit Zwei-Faktor-Authentifizierung (2FA) zu erstellen, verwenden Sie den `-2fa`-Parameter und geben einen Dateipfad an. Weitere Informationen finden Sie unter the section called “Argumente”.

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

Beispiele

Diese Beispiele verdeutlichen, wie mit createUser neue Benutzer in Ihren HSMs erstellt werden.

Example : Erstellen eines Crypto Officers

Dieses Beispiel erstellt einen Verschlüsselungsverantwortlichen (CO, Crypto Officer) in den HSMs in einem Cluster. Der erste Befehl verwendet loginHSM, um sich beim HSM als Verschlüsselungsverantwortlicher anzumelden.

```
aws-cloudhsm> loginHSM CO admin 735782961

loginHSM success on server 0(10.0.0.1)
loginHSM success on server 1(10.0.0.2)
loginHSM success on server 1(10.0.0.3)
```

Der zweite Befehl verwendet den createUser-Befehl zum Erstellen von `alice`, einem neuen Verschlüsselungsverantwortlichen im HSM.

Die Meldung weist darauf hin, dass der Befehl Benutzer in allen HSMs im Cluster erstellt. Wenn der Befehl jedoch in irgendwelchen HSMs fehlschlägt, ist der Benutzer in diesen HSMs nicht vorhanden. Geben Sie y ein, um fortzufahren.

Die Ausgabe zeigt, dass der neue Benutzer in allen drei HSMs im Cluster erstellt wurde.

```
aws-cloudhsm> createUser CO alice 391019314

************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)?Invalid option, please type 'y' or 'n'

Do you want to continue(y/n)?y
Creating User alice(CO) on 3 nodes
```

Wenn der Befehl abgeschlossen ist, verfügt `alice` über dieselben Berechtigungen im HSM wie der CO-Benutzer `admin`, einschließlich der Berechtigung zum Ändern der Passwörter aller Benutzer in den HSMs.

Der letzte Befehl verwendet den listUsers-Befehl, um zu überprüfen, ob `alice` in allen drei HSMs im Cluster vorhanden ist. Die Ausgabe zeigt auch, dass `alice` die Benutzer-ID `3` zugewiesen wird.. Sie verwenden die Benutzer-ID, um `alice` in anderen Befehlen zu identifizieren, etwa in findAllKeys.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO

Users on server 1(10.0.0.3):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO
```

Example : Erstellen eines Crypto-Benutzers

In diesem Beispiel wird ein Crypto-Benutzer (CU, Crypto User), bob, im HSM erstellt. Crypto-Benutzer können Schlüssel erstellen und verwalten, aber sie können keine Benutzer verwalten.

Nachdem Sie y als Antwort auf die Meldung eingegeben haben, zeigt die Ausgabe, dass bob in allen drei HSMs im Cluster erstellt wurde. Der neue CU kann sich bei dem HSM anmelden, um Schlüssel zu erstellen und zu verwalten.

Der Befehl verwendete den Passwortwert `defaultPassword`. Später kann bob oder ein beliebiger CO mithilfe des changePswd-Befehls sein Passwort ändern.

```
aws-cloudhsm> createUser CU bob defaultPassword

************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?Invalid option, please type 'y' or 'n'

Do you want to continue(y/n)?y
Creating User bob(CU) on 3 nodes
```

Argumente

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um einen CO-Benutzer mit aktivierter 2FA zu erstellen, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen über 2FA finden Sie in Verwendung von CMU zur Verwaltung von 2FA.

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

Gibt den Benutzertyp an. Dieser Parameter muss angegeben werden.

Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter Grundlegendes zu HSM-Benutzern.

Zulässige Werte:

- CO: Verschlüsselungsverantwortliche können Benutzer, aber keine Schlüssel verwalten.
- CU: Crypto-Benutzer können Schlüssel erstellen und verwalten und Schlüssel in kryptografischen Vorgängen verwenden.

Der PRECO wird in einen CO konvertiert, wenn Sie während der HSM-Aktivierung ein Passwort zuweisen.

Erforderlich: Ja

<user-name>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. In cloudhsm_mgmt_util-Befehlen muss bei Benutzertyp und Passwort die Groß- und Kleinschreibung beachtet werden, beim Benutzernamen nicht.

Erforderlich: Ja

<password | -hpswd >

Gibt ein Passwort für den Benutzer an. Geben Sie eine Zeichenfolge von 7 bis 32 Zeichen ein. Bei diesem Wert ist die Groß- und Kleinschreibung zu beachten. Das Passwort wird in Klartext angezeigt, wenn Sie es eingeben. Um Ihr Passwort zu verbergen, verwenden Sie den `-hpswd`-Parameter anstelle des Passworts und folgen Sie den Aufforderungen.

Um ein Benutzerpasswort zu ändern, verwenden Sie changePswd. Jeder HSM-Benutzer kann das eigene Passwort ändern, aber CO-Benutzer können das Passwort eines jeden Benutzers (jedes Typs) in den HSMs ändern.

Erforderlich: Ja

[-2fa </path/to/authdata>]

Gibt die Erstellung eines CO-Benutzers mit aktivierter 2FA an. Um die für die Einrichtung der 2FA-Authentifizierung erforderlichen Daten abzurufen, fügen Sie einen Pfad zu einem Speicherort im Dateisystem mit einem Dateinamen nach dem `-2fa`-Parameter ein. Weitere Informationen zum Einrichten und Arbeiten mit 2FA finden Sie unter Verwendung von CMU zur Verwaltung von 2FA.

Erforderlich: Nein

Verwandte Themen

- listUsers
- deleteUser
- syncUser
- changePswd

deleteUser

Der deleteUser-Befehl in cloudhsm_mgmt_util löscht einen Benutzer aus den Hardware-Sicherheitsmodulen (HSM). Nur Crypto Officers (CO) können diesen Befehl ausführen. Sie können keinen Benutzer löschen, der derzeit bei einem HSM angemeldet ist. Weitere Informationen zum Löschen von Benutzern finden Sie unter Löschen von HSM-Benutzern.

> **Tip**
>
> Sie können keine Crypto-Benutzer (CU) löschen, die Schlüssel besitzen.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- CO

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
deleteUser <user-type> <user-name>
```

Beispiel

Dieses Beispiel löscht einen Verschlüsselungsverantwortlichen (CO) aus den HSMs in einem Cluster. Der erste Befehl verwendet listUsers, um alle Benutzer in den HSMs aufzulisten.

Die Ausgabe zeigt, dass Benutzer `3`, `alice`, ein CO in den HSMs ist.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO

Users on server 1(10.0.0.3):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO
```

Der zweite Befehl verwendet den Befehl deleteUser, um `alice` in den HSMs zu löschen.

Die Ausgabe zeigt, dass der Befehl in allen drei HSMs im Cluster erfolgreich ausgeführt wurde.

```
aws-cloudhsm> deleteUser CO alice
Deleting user alice(CO) on 3 nodes
deleteUser success on server 0(10.0.0.1)
```

```
deleteUser success on server 0(10.0.0.2)
deleteUser success on server 0(10.0.0.3)
```

Der letzte Befehl verwendet listUsers, um zu überprüfen, ob `alice` in allen drei HSMs im Cluster gelöscht wurde.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:2

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:2

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
Users on server 1(10.0.0.3):
Number of users found:2

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
deleteUser <user-type> <user-name>
```

<user-type>

Gibt den Benutzertyp an. Dieser Parameter muss angegeben werden.

> **Tip**
>
> Sie können keine Crypto-Benutzer (CU) löschen, die Schlüssel besitzen.

Gültige Werte sind CO und CU.

Zum Abrufen des Benutzertyps verwenden Sie den Befehl listUsers. Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter Grundlegendes zu HSM-Benutzern.

Erforderlich: Ja

<user-name>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. In cloudhsm_mgmt_util-Befehlen muss bei Benutzertyp und Passwort die Groß- und Kleinschreibung beachtet werden, beim Benutzernamen nicht.

Erforderlich: Ja

Verwandte Themen

- listUsers
- createUser
- syncUser
- changePswd

findAllKeys

Mit dem Befehl findAllKeys in cloudhsm_mgmt_util, werden die Schlüssel abgerufen, die ein angegebener Crypto-Benutzer (CU) besitzt oder die für ihn freigegeben sind. Außerdem wird ein Hashwert der Benutzerdaten für die einzelnen HSMs abgerufen. Sie können mit dem Hashwert auf

einen Blick feststellen, ob die Daten für Benutzer, Schlüsseleigentümer und -freigaben in allen HSMs des Clusters identisch sind. In der Ausgabe werden die Schlüssel, die dem Benutzer gehören, von (o) mit Anmerkungen versehen, und gemeinsam genutzte Schlüssel werden von (s) kommentiert.

findAllKeys gibt nur dann öffentliche Schlüssel zurück, wenn der angegebene CU den Schlüssel besitzt, auch wenn alle CUs im HSM jeden beliebigen öffentlichen Schlüssel verwenden können. Dieses Verhalten unterscheidet sich vom Befehl findKey in key_mgmt_util, der öffentliche Schlüssel für alle Crypto-Benutzer zurückgibt.

Nur Verschlüsselungsverantwortliche (COs und PCOs) und Appliance-Benutzer (AUs) können diesen Befehl verwenden. Crypto-Benutzer (CUs, Crypto Users) können die folgenden Befehle ausführen:

- listUsers, um alle Benutzer zu suchen
- findKey in key_mgmt_util, um die Schlüssel, die sie verwenden können, zu suchen
- getKeyInfo in key_mgmt_util, um den Besitzer und gemeinsame Benutzer eines bestimmten Schlüssels zu finden, den sie besitzen oder der für sie freigegeben ist

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (CO, PCO)
- Appliance-Benutzer (Appliance User, AU)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

Beispiele

Diese Beispiele verdeutlichen, wie Sie mit `findAllKeys` alle Schlüssel für einen Benutzer und einen Hashwert der Schlüsselbenutzerdaten in den einzelnen HSMs suchen können.

Example : Suchen der Schlüssel für einen CU

In diesem Beispiel wird findAllKeys verwendet, um die Schlüssel in den HSMs zu suchen, die Benutzer 4 besitzt und die für ihn freigegeben sind. Der Befehl verwendet den Wert `0` für das zweite Argument, um den Hashwert zu unterdrücken. Da der optionale Dateiname weggelassen wird, führt der Befehl Schreibvorgänge in der Standardausgabe (stdout) aus.

Die Ausgabe zeigt, dass Benutzer 4 6 Schlüssel verwenden kann: 8, 9, 17, 262162, 19 und 31. Die Ausgabe verwendet ein `(s)`, um Schlüssel anzugeben, die explizit vom Benutzer freigegeben werden. Die Schlüssel, die der Benutzer besitzt, werden durch ein `(o)` gekennzeichnet und umfassen symmetrische und private Schlüssel, die der Benutzer nicht freigibt, sowie öffentliche Schlüssel, die für alle Crypto-Benutzer verfügbar sind.

```
aws-cloudhsm> findAllKeys 4 0
Keys on server 0(10.0.0.1):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 0(10.0.0.1)

Keys on server 1(10.0.0.2):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 1(10.0.0.2)

Keys on server 1(10.0.0.3):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 1(10.0.0.3)
```

Example : Sicherstellen, dass Benutzerdaten synchronisiert sind

Dieses Beispiel verwendet findAllKeys, um sicherzustellen, dass alle HSMs im Cluster dieselben Werte für Benutzer, Schlüsseleigentümer und gemeinsame Schlüsselverwendung enthalten. Zu

diesem Zweck wird ein Hashwert der Schlüsselbenutzerdaten in jedem HSM abgerufen und die Hashwerte werden verglichen.

Zum Abrufen des Schlüsselhashwerts verwendet der Befehl den Wert 1 für das zweite Argument. Der optionale Dateiname wird weggelassen, sodass der Befehl den Schlüsselhashwert in die Standardausgabe (stdout) schreibt.

Das Beispiel gibt den Benutzer 6 an, aber der Hashwert ist für jeden Benutzer, der einen der Schlüssel in den HSMs besitzt oder gemeinsam verwendet, identisch. Wenn der angegebene Benutzer keine Schlüssel besitzt oder gemeinsam verwendet, z. B. ein Verschlüsselungsverantwortlicher (CO), gibt der Befehl keinen Hashwert zurück.

Die Ausgabe zeigt, dass der Schlüsselhashwert in beiden HSMs im Cluster identisch ist. Wenn eines der HSMs verschiedene Benutzer, unterschiedliche Schlüsselbesitzer oder verschiedene gemeinsame Benutzer hätte, wären die Schlüsselhashwerte nicht identisch.

```
aws-cloudhsm> findAllKeys 6 1
Keys on server 0(10.0.0.1):
Number of keys found 3
number of keys matched from start index 0::3
8(s),9(s),11,17(s)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 0(10.0.0.1)
Keys on server 1(10.0.0.2):
Number of keys found 3
number of keys matched from start index 0::3
8(s),9(s),11(o),17(s)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 1(10.0.0.2)
```

Mit diesem Befehl wird gezeigt, dass der Hashwert die Benutzerdaten für alle Schlüssel im HSM darstellt. Der Befehl verwendet findAllKeys für Benutzer 3. Im Gegensatz zu Benutzer 6, der nur 3 Schlüssel besitzt oder gemeinsam verwendet, besitzt oder verwendet Benutzer 3 17 Schlüssel. Dennoch ist der Schlüsselhashwert identisch.

```
aws-cloudhsm> findAllKeys 3 1
Keys on server 0(10.0.0.1):
```

```
Number of keys found 17
number of keys matched from start index 0::17
6(o),7(o),8(s),11(o),12(o),14(o),262159(o),262160(o),17(s),262162(s),19(s),20(o),21(o),262177(o
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 0(10.0.0.1)
Keys on server 1(10.0.0.2):
Number of keys found 17
number of keys matched from start index 0::17
6(o),7(o),8(s),11(o),12(o),14(o),262159(o),262160(o),17(s),262162(s),19(s),20(o),21(o),262177(o
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 1(10.0.0.2)
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

<user id>

Ruft alle Schlüssel ab, die der Benutzer besitzt oder die für ihn freigegeben sind. Geben Sie die Benutzer-ID eines Benutzers in den HSMs ein. Verwenden Sie zur Suche der Benutzer-IDs aller Benutzer den Befehl listUsers.

Alle Benutzer-IDs sind gültig, jedoch gibt `findAllKeys` nur Schlüssel für Crypto-Benutzer (CUs) zurück.

Erforderlich: Ja

<key hash>

Umfasst (`1`) oder schließt (`0`) einen Hashwert des Benutzereigentums oder der Freigabedaten für alle Schlüssel in jedem HSM aus.

Wenn das Argument `user id` einen Benutzer darstellt, der Schlüssel besitzt oder gemeinsam verwendet, ist der Schlüsselhashwert ausgefüllt. Der Schlüsselhashwert ist für alle Benutzer, die Schlüssel besitzen oder im HSM gemeinsam verwenden, identisch, auch wenn sie verschiedene Schlüssel besitzen und gemeinsam verwenden. Wenn `user id` jedoch einen

Benutzer darstellt, der keine Schlüssel besitzt oder gemeinsam verwendet, z. B. einen Verschlüsselungsverantwortlichen (CO), wird der Hashwert nicht angegeben.

Erforderlich: Ja

<output file>

Schreibt die Ausgabe in die angegebene Datei.

Erforderlich: Nein

Standard: Stdout

Verwandte Themen

- changePswd
- deleteUser
- listUsers
- syncUser
- findKey in key_mgmt_util
- getKeyInfo in key_mgmt_util

getAttribute

Der Befehl getAttribute in cloudhsm_mgmt_util ruft einen Attributwert für einen Schlüssel aus allen HSMs im Cluster ab und schreibt diesen in die Standardausgabe (stdout) oder in eine Datei. Nur Crypto-Benutzer (CUs) können diesen Befehl ausführen.

Schlüsselattribute sind Eigenschaften eines Schlüssels. Dazu zählen Merkmale wie der Schlüsseltyp, Klasse, Beschriftung und ID sowie Werte, die Aktionen darstellen, die Sie für den Schlüssel ausführen können, wie verschlüsseln, entschlüsseln, packen, signieren und überprüfen.

Sie können getAttribute nur für Schlüssel ausführen, die Ihnen gehören oder die für Sie freigegeben wurden. Sie können diesen Befehl oder den Befehl getAttribute in key_mgmt_util ausführen, der einen oder alle Attributwerte eines Schlüssels in eine Datei schreibt.

Zum Abrufen einer Liste der Attribute und Konstanten, die sie darstellen, verwenden Sie den Befehl listAttributes. Um die Attributwerte von vorhandenen Schlüsseln zu ändern, verwenden Sie setAttribute in key_mgmt_util und setAttribute in cloudhsm_mgmt_util. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Crypto-Benutzer (Crypto User, CU)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
getAttribute <key handle> <attribute id> [<filename>]
```

Beispiel

In diesem Beispiel wird der Wert des extrahierbaren Attributs für einen Schlüssel in den HSMs abgerufen. Sie können einen Befehl wie diesen verwenden, um zu ermitteln, ob Sie einen Schlüssel aus den HSMs exportieren können.

Der erste Befehl verwendet listAttributes, um die Konstante zu suchen, die das extrahierbare Attribut darstellt. Die Ausgabe zeigt, dass die Konstante für `OBJ_ATTR_EXTRACTABLE` `354` ist. Sie können diese Informationen mit Beschreibungen der Attribute und ihrer Werte auch unter Schlüsselattributreferenz finden.

```
aws-cloudhsm> listAttributes

Following are the possible attribute values for getAttribute:

      OBJ_ATTR_CLASS                  = 0
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
```

```
    OBJ_ATTR_TRUSTED                = 134
    OBJ_ATTR_KEY_TYPE               = 256
    OBJ_ATTR_ID                     = 258
    OBJ_ATTR_SENSITIVE              = 259
    OBJ_ATTR_ENCRYPT                = 260
    OBJ_ATTR_DECRYPT                = 261
    OBJ_ATTR_WRAP                   = 262
    OBJ_ATTR_UNWRAP                 = 263
    OBJ_ATTR_SIGN                   = 264
    OBJ_ATTR_VERIFY                 = 266
    OBJ_ATTR_DERIVE                 = 268
    OBJ_ATTR_LOCAL                  = 355
    OBJ_ATTR_MODULUS                = 288
    OBJ_ATTR_MODULUS_BITS           = 289
    OBJ_ATTR_PUBLIC_EXPONENT        = 290
    OBJ_ATTR_VALUE_LEN              = 353
    OBJ_ATTR_EXTRACTABLE            = 354
    OBJ_ATTR_NEVER_EXTRACTABLE      = 356
    OBJ_ATTR_ALWAYS_SENSITIVE       = 357
    OBJ_ATTR_DESTROYABLE            = 370
    OBJ_ATTR_KCV                    = 371
    OBJ_ATTR_WRAP_WITH_TRUSTED      = 528
    OBJ_ATTR_WRAP_TEMPLATE          = 1073742353
    OBJ_ATTR_UNWRAP_TEMPLATE        = 1073742354
    OBJ_ATTR_ALL                    = 512
```

Der zweite Befehl verwendet getAttribute, um den Wert des extrahierbaren Attributs für den Schlüssel mit dem Schlüssel-Handle `262170` in den HSMs abzurufen. Zum Angeben des extrahierbaren Attributs verwendet der Befehl `354`, die Konstante, die das extrahierbare Attribut darstellt. Da der Befehl keinen Dateinamen angibt, schreibt getAttribute die Ausgabe in die Standardausgabe (stdout).

Die Ausgabe zeigt, dass der Wert des extrahierbaren Attributs im gesamten HSM 1 ist. Dieser Wert gibt an, dass der Eigentümer des Schlüssels ihn exportieren kann. Wenn der Wert 0 (0x0) ist, kann er aus den HSMs nicht exportiert werden. Sie legen den Wert des extrahierbaren Attributs beim Erstellen eines Schlüssels fest und können ihn nicht mehr ändern.

```
aws-cloudhsm> getAttribute 262170 354

Attribute Value on server 0(10.0.1.10):
OBJ_ATTR_EXTRACTABLE
0x00000001
```

```
Attribute Value on server 1(10.0.1.12):
OBJ_ATTR_EXTRACTABLE
0x00000001

Attribute Value on server 2(10.0.1.7):
OBJ_ATTR_EXTRACTABLE
0x00000001
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
getAttribute <key handle> <attribute id> [<filename>]
```

<key-handle>

Gibt das Schlüssel-Handle des Zielschlüssels an. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Um das Schlüsselhandle eines Schlüssels zu erhalten, verwenden Sie findKey in key_mgmt_util.

Sie müssen Eigentümer des angegebenen Schlüssels sein oder er muss für Sie freigegeben sein. Um die Benutzer eines Schlüssels zu ermitteln, verwenden Sie getKeyInfo in key_mgmt_util.

Erforderlich: Ja

<attribute id>

Bezeichnet das Attribut. Geben Sie eine Konstante, die ein Attribut darstellt, oder den Wert `512` ein, der alle Attribute repräsentiert. Zum Abrufen des Schlüsseltyps geben Sie z. B. `256` ein. Dies ist die Konstante für das Attribut `OBJ_ATTR_KEY_TYPE`.

Verwenden Sie listAttributes, um die Attribute und deren Konstanten aufzulisten. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Erforderlich: Ja

<filename>

Schreibt die Ausgabe in die angegebene Datei. Geben Sie einen Dateipfad ein.

Wenn die angegebene Datei vorhanden ist, überschreibt getAttribute die Datei ohne Warnung.

Erforderlich: Nein

Standard: Stdout

Verwandte Themen

- getAttribute in key_mgmt_util
- listAttributes
- setAttribute in cloudhsm_mgmt_util
- setAttribute in key_mgmt_util
- Schlüsselattributreferenz

getCert

Mit dem Befehl getCert in cloudhsm_mgmt_util können Sie die Zertifikate von einem bestimmten HSM in einem Cluster abrufen. Wenn Sie den Befehl ausführen, geben Sie den Typ des abzurufenden Zertifikats an. Dazu verwenden Sie eine der entsprechenden Ganzzahlen nach der Beschreibung im Abschnitt Argumente unten. Weitere Informationen über die Rolle der einzelnen Zertifikate finden Sie unter Überprüfen der HSM-Identität.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer.

Voraussetzungen

Bevor Sie beginnen, müssen Sie auf dem Ziel-HSM den Servermodus aufrufen. Weitere Informationen finden Sie unter Server.

Syntax

So verwenden Sie den Befehl getCert einmal im Servermodus:

```
server> getCert <file-name> <certificate-type>
```

Beispiel

Rufen Sie zunächst den Servermodus auf. Dieser Befehl ruft den Servermodus auf einem HSM mit der Servernummer `0` auf.

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

Verwenden Sie dann den getCert-Befehl. In diesem Beispiel verwenden wir `/tmp/P0.crt` als Name der Datei, in der das Zertifikat gespeichert werden soll, und `4` (Kunden-Stammzertifikat) als gewünschten Zertifikattyp:

```
server0> getCert /tmp/P0.crt 4
getCert Success
```

Argumente

```
getCert <file-name> <certificate-type>
```

<file-name>

Gibt den Namen der Datei an, in der das Zertifikat gespeichert wird.

Erforderlich: Ja

<certificate-type>

Eine Ganzzahl, die den Typ des abzurufenden Zertifikats angibt. Die Ganzzahlen und ihre entsprechenden Zertifikattypen sind:

- 1 – Hersteller-Stammzertifikat
- 2 – Hersteller-Hardwarezertifikat
- 4 – Kunden-Stammzertifikat
- 8 – Clusterzertifikat (signiert vom Kunden-Stammzertifikat)

- 16 – Clusterzertifikat (verkettet mit dem Hersteller-Stammzertifikat)

Erforderlich: Ja

Verwandte Themen

- server

getHSMInfo

Der getHSMInfo-Befehl in cloudhsm_mgmt_util ruft Informationen über die Hardware ab, auf der jedes HSM ausgeführt wird. Hierzu gehören das Modell, Seriennummern, FIPS-Status, Speicher, Temperatur und die Versionsnummer der Hardware und der Firmware. Die Informationen umfassen auch die Server-ID, die cloudhsm_mgmt_util verwendet, um auf das HSM zu verweisen.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

Dieser Befehl hat keine Parameter.

```
getHSMInfo
```

Beispiel

In diesem Beispiel wird getHSMInfo verwendet, um Informationen über die HSMs im Cluster abzurufen.

```
aws-cloudhsm> getHSMInfo
Getting HSM Info on 3 nodes
                *** Server 0 HSM Info ***

        Label                :cavium
        Model                :NITROX-III CNN35XX-NFBE

        Serial Number        :3.0A0101-ICM000001
        HSM Flags            :0
        FIPS state           :2 [FIPS mode with single factor authentication]

        Manufacturer ID      :
        Device ID            :10
        Class Code           :100000
        System vendor ID     :177D
        SubSystem ID         :10


        TotalPublicMemory    :560596
        FreePublicMemory     :294568
        TotalPrivateMemory   :0
        FreePrivateMemory    :0

        Hardware Major       :3
        Hardware Minor       :0

        Firmware Major       :2
        Firmware Minor       :03

        Temperature          :56 C

        Build Number         :13

        Firmware ID          :xxxxxxxxxxxxxx

...
```

Verwandte Themen

- info

getKeyInfo

Der getKeyInfo-Befehl in key_mgmt_util gibt die HSM-Benutzer-IDs der Benutzer zurück, die den Schlüssel verwenden können, einschließlich Eigentümer und Crypto-Benutzer (CUs), mit denen der Schlüssel geteilt wird. Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, gibt getKeyInfo auch die Anzahl der Benutzer zurück, die kryptografische Operationen genehmigen müssen, von denen der Schlüssel verwendet wird. Sie können getKeyInfo nur auf Schlüsseln ausführen, die Ihnen gehören oder die für Sie freigegeben wurden.

Wenn Sie getKeyInfo auf öffentliche Schlüsseln ausführen, gibt getKeyInfo nur den Schlüsseleigentümer zurück, auch wenn alle HSM-Benutzer den öffentlichen Schlüssel verwenden können. Verwenden Sie zur Suche nach den HSM-Benutzer-IDs der Benutzer in Ihren HSMs listUsers. Um nach den Schlüsseln für einen bestimmten Benutzer zu suchen, verwenden Sie findKey `-u` in key_mgmt_util. Crypto Officers können findAllKeys in cloudhsm_mgmt_util verwenden.

Ihren gehören die Schlüssel, die Sie erstellen. Sie können einen Schlüssel für andere Benutzer freigeben, wenn Sie ihn erstellen. Nutzen Sie dann zum Freigeben oder zum Aufheben der Freigabe shareKey in cloudhsm_mgmt_util.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (Crypto User, CU)

Syntax

```
getKeyInfo -k <key-handle> [<output file>]
```

Beispiele

Diese Beispiele veranschaulichen die Verwendung von getKeyInfo zum Abrufen von Informationen über die Benutzer eines Schlüssels.

Example : Abfragen der Benutzer eines asymmetrischen Schlüssels

Über diesen Befehl erhalten Sie die Benutzer, die den (asymmetrischen) AES-Schlüssel mit dem Schlüssel-Handle 262162 verwenden können. Die Ausgabe zeigt, dass Benutzer 3 den Schlüssel besitzt und mit Benutzer 4 und 6 teilt.

Nur die Benutzer 3, 4 und 6 können getKeyInfo auf dem Schlüssel 262162 ausführen.

```
aws-cloudhsm>getKeyInfo 262162
Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 2 user(s):

                4
                6
Key Info on server 1(10.0.0.2):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 2 user(s):

                4
                6
```

Example : Abfragen der Benutzer eines symmetrischen Schlüsselpaares

Diese Befehle verwenden getKeyInfo, um die Benutzer abzurufen, die die Schlüssel in einem (symmetrischen) ECC-Schlüsselpaar verwenden können. Das Schlüssel-Handle des öffentlichen Schlüssels ist 262179. Das Schlüssel-Handle des privaten Schlüssels ist 262177.

Wenn Sie getKeyInfo auf dem privaten Schlüssel (262177) ausführen, werden der Schlüsselbesitzer (3) und die Crypto-Benutzer (CUs) 4 zurückgegeben, mit denen der Schlüssel geteilt wird.

```
aws-cloudhsm>getKeyInfo -k 262177
Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
Key Info on server 1(10.0.0.2):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
```

Wenn Sie getKeyInfo auf dem öffentlichen Schlüssel (262179) ausführen, wird nur der Schlüsseleigentümer, Benutzer 3, zurückgegeben.

```
aws-cloudhsm>getKeyInfo -k 262179
Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3

Key Info on server 1(10.0.3.6):

        Token/Flash Key,

        Owned by user 3
```

Um zu bestätigen, dass Benutzer 4 den öffentlichen Schlüssel (und alle öffentlichen Schlüssel im HSM) verwenden kann, verwenden Sie den -u-Parameter von findKey in key_mgmt_util.

Die Ausgabe zeigt, dass Benutzer 4 sowohl den öffentlichen (262179) als auch den privaten Schlüssel (262177) im Schlüsselpaar verwenden kann. Benutzer 4 kann zudem alle anderen öffentlichen und privaten Schlüssel nutzen, die sie erstellt haben oder die für sie freigegeben wurden.

```
Command:  findKey -u 4

Total number of keys present 8

 number of keys matched from start index 0::7
11, 12, 262159, 262161, 262162, 19, 20, 21, 262177, 262179

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example : Abrufen des Quorum-Authentifizierungswerts (m_value) für einen Schlüssel

Dieses Beispiel zeigt, wie der `m_value` für einen Schlüssel abgerufen wird. m_value gibt die Anzahl der Benutzer im Quorum an, die alle kryptografischen Operationen genehmigen müssen, bei denen der Schlüssel verwendet wird, sowie alle Operationen zum Freigeben oder zum Aufheben der Freigabe eines Schlüssels.

Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, muss das Quorum der Benutzer alle kryptografischen Operationen genehmigen, bei denen der Schlüssel verwendet wird. Zum Aktivieren der Quorum-Authentifizierung und Festlegen der Quorum-Größe nutzen Sie beim Erstellen des Schlüssels den Parameter `-m_value`.

Dieser Befehl verwendet genSymKey zum Erstellen eines 256-Bit-AES-Schlüssels, der mit Benutzer 4 geteilt wird. Er verwendet den Parameter `m_value` zur Aktivierung der Quorum-Authentifizierung und zur Festlegung der Quorum-Größe auf zwei Benutzer. Die Anzahl der Benutzer muss groß genug sein, um die erforderlichen Genehmigungen bereitstellen zu können.

Die Ausgabe zeigt, dass die Befehl Schlüssel 10 erstellt hat.

```
Command:  genSymKey -t 31 -s 32 -l aes256m2 -u 4 -m_value 2

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 10

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Dieser Befehl nutzt getKeyInfo in cloudhsm_mgmt_util zum Abrufen von Informationen über die Benutzer des Schlüssels 10. Die Ausgabe zeigt, dass der Schlüssel im Besitz des Benutzers 3 ist und gemeinsam mit Benutzer 4 verwendet wird. Sie zeigt auch, dass ein Quorum von zwei Benutzern jede kryptografische Operation genehmigen muss, bei der der Schlüssel verwendet wird.

```
aws-cloudhsm>getKeyInfo 10

Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
         2 Users need to approve to use/manage this key
Key Info on server 1(10.0.0.2):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
         2 Users need to approve to use/manage this key
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
getKeyInfo -k <key-handle> <output file>
```

<key-handle>

Gibt das Schlüssel-Handle eines Schlüssels im HSM an. Geben Sie das Schlüssel-Handle eines Schlüssels ein, den Sie besitzen oder teilen. Dieser Parameter muss angegeben werden.

Erforderlich: Ja

<output file>

Schreibt die Ausgabe nicht in stdout, sondern in die angegebene Datei. Wenn die Datei vorhanden ist, wird diese ohne Warnung überschrieben.

Erforderlich: Nein

Standard: stdout

Verwandte Themen

- getKeyInfo in key_mgmt_util
- findKey in key_mgmt_util
- findAllKeys in cloudhsm_mgmt_util
- listUsers
- shareKey

info

Der info-Befehl in cloudhsm_mgmt_util ruft Informationen über jedes der HSMs im Cluster ab, einschließlich des Hostnamens, des Ports, der IP-Adresse und des Namens und Typs des Benutzers, der bei cloudhsm_mgmt_util auf dem HSM angemeldet ist.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
info server <server ID>
```

Beispiel

In diesem Beispiel wird info verwendet, um Informationen über ein HSM im Cluster abzurufen. Der Befehl verwendet 0, um auf das erste HSM im Cluster zu verweisen. Die Ausgabe zeigt die IP-Adresse, den Port, den Typ und den Namen des aktuellen Benutzers an.

```
aws-cloudhsm> info server 0
Id      Name                    Hostname          Port    State           Partition
         LoginState
0       10.0.0.1                10.0.0.1         2225    Connected       hsm-udw0tkfg1ab
         Logged in as 'testuser(CU)'
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
info server <server ID>
```

<server id>

Gibt die Server-ID des HSMs an. Den HSMs werden Ordinalzahlen zugewiesen, die die Reihenfolge angeben, in der sie dem Cluster hinzugefügt werden (beginnend mit 0). Um die Server-ID eines HSMs zu ermitteln, verwenden Sie getHSMInfo.

Erforderlich: Ja

Verwandte Themen

- getHSMInfo
- loginHSM und logoutHSM

listAttributes

Der listAttributes-Befehl in cloudhsm_mgmt_util listet die Attribute eines AWS CloudHSM-Schlüssels und die Konstanten, die sie repräsentieren, auf. Sie nutzen die Konstanten zum Identifizieren der Attribute in den Befehlen getAttribute und setAttribute.

Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

```
listAttributes [-h]
```

Beispiel

Mit diesem Befehl werden die Schlüsselattribute aufgelistet, die Sie mit key_mgmt_util abrufen und ändern können, sowie die Konstanten, die sie darstellen. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz. Um alle Attribute darzustellen, verwenden Sie 512.

```
Command: listAttributes

    Description
===========
The following are all of the possible attribute values for getAttribute.

      OBJ_ATTR_CLASS                  = 0
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
      OBJ_ATTR_TRUSTED                = 134
      OBJ_ATTR_KEY_TYPE               = 256
      OBJ_ATTR_ID                     = 258
      OBJ_ATTR_SENSITIVE              = 259
```

```
    OBJ_ATTR_ENCRYPT                = 260
    OBJ_ATTR_DECRYPT                = 261
    OBJ_ATTR_WRAP                   = 262
    OBJ_ATTR_UNWRAP                 = 263
    OBJ_ATTR_SIGN                   = 264
    OBJ_ATTR_VERIFY                 = 266
    OBJ_ATTR_DERIVE                 = 268
    OBJ_ATTR_LOCAL                  = 355
    OBJ_ATTR_MODULUS                = 288
    OBJ_ATTR_MODULUS_BITS           = 289
    OBJ_ATTR_PUBLIC_EXPONENT        = 290
    OBJ_ATTR_VALUE_LEN              = 353
    OBJ_ATTR_EXTRACTABLE            = 354
    OBJ_ATTR_NEVER_EXTRACTABLE      = 356
    OBJ_ATTR_ALWAYS_SENSITIVE       = 357
    OBJ_ATTR_DESTROYABLE            = 370
    OBJ_ATTR_KCV                     = 371
    OBJ_ATTR_WRAP_WITH_TRUSTED       = 528
    OBJ_ATTR_WRAP_TEMPLATE           = 1073742353
    OBJ_ATTR_UNWRAP_TEMPLATE        = 1073742354
    OBJ_ATTR_ALL                    = 512
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

Verwandte Themen

- getAttribute
- setAttribute
- Schlüsselattributreferenz

listUsers

Mit dem listUsers-Befehl in cloudhsm_mgmt_util werden die Benutzer in den einzelnen HSMs zusammen mit dem Benutzertyp und anderen Attributen abgerufen. Alle Arten von Benutzern können

diesen Befehl ausführen. Sie müssen nicht bei cloudhsm_mgmt_util angemeldet sein, um diesen Befehl auszuführen.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

Dieser Befehl hat keine Parameter.

```
listUsers
```

Beispiel

Mit diesem Befehl werden die Benutzer der einzelnen HSMs im Cluster aufgeführt und deren Attribute angezeigt. Sie können das Attribut `User ID` zum Identifizieren von Benutzern in anderen Befehlen, z. B. deleteUser, changePswd und findAllKeys, verwenden.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:6

    User Id             User Type       User Name            MofnPubKey
 LoginFailureCnt         2FA
        1              PCO             admin                    YES              0
            NO
        2              AU              app_user                  NO              0
            NO
        3              CU              crypto_user1              NO              0
            NO
```

```
        4               CU              crypto_user2                NO              0
            NO
        5               CO              officer1                    YES             0
            NO
        6               CO              officer2                    NO              0
            NO
Users on server 1(10.0.0.2):
Number of users found:5

    User Id             User Type       User Name           MofnPubKey
 LoginFailureCnt         2FA
        1               PCO             admin                   YES             0
            NO
        2               AU              app_user                NO              0
            NO
        3               CU              crypto_user1            NO              0
            NO
        4               CU              crypto_user2            NO              0
            NO
        5               CO              officer1                YES             0
            NO
```

Die Ausgabe umfasst die folgenden Benutzerattribute:

- Benutzer-ID: Identifiziert den Benutzer in den Befehlen key_mgmt_util und cloudhsm_mgmt_util.
- User type: Bestimmt die Operationen, die der Benutzer im HSM ausführen kann.
- User Name: Zeigt den benutzerdefinierten Anzeigenamen für den Benutzer an.
- MofnPubKey: Gibt an, ob der Benutzer ein Schlüsselpaar zum Signieren von Token für die Quorum-Authentifizierung registriert hat.
- LoginFailureCnt: Gibt an, wie oft der Benutzer sich nicht erfolgreich angemeldet hat.
- 2FA: Gibt an, dass der Benutzer Multifaktor-Authentifizierung aktiviert hat.

Verwandte Themen

- listUsers in key_mgmt_util
- createUser
- deleteUser
- changePswd

loginHSM und logoutHSM

Sie können die Befehle loginHSM und logoutHSM in cloudhsm_mgmt_util verwenden, um sich bei jedem HSM in einem Cluster an- und abzumelden. Jeder beliebige Benutzer kann unabhängig vom Typ diese Befehle verwenden.

> **Note**
>
> Wenn Sie mehr als fünf falsche Anmeldeversuche tätigen, wird Ihr Konto gesperrt. Ein Crypto Officer (CO) muss Ihr Passwort mit dem Befehl changePswd in cloudhsm_mgmt_util zurücksetzen, um das Konto zu entsperren.

So beheben Sie Probleme mit dem LoginHSM und dem LogoutHSM

Bevor Sie diese cloudhsm_mgmt_util-Befehle ausführen, müssen Sie cloudhsm_mgmt_util starten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien, die vom AWS CloudHSM-Client und den Befehlszeilen-Tools verwendet werden. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Wenn Sie mehr als ein HSM in Ihrem Cluster haben, sind Ihnen möglicherweise weitere falsche Anmeldeversuche gestattet, bevor Ihr Konto gesperrt wird. Dies liegt daran, dass der CloudHSM-Client die Last auf verschiedene HSMs verteilt. Aus diesem Grund beginnt der Anmeldeversuch möglicherweise nicht jedes Mal auf demselben HSM. Wenn Sie diese Funktion testen, empfehlen wir Ihnen, dies auf einem Cluster mit nur einem aktiven HSM zu tun.

Wenn Sie Ihren Cluster vor Februar 2018 erstellt haben, wird Ihr Konto nach 20 falschen Anmeldeversuchen gesperrt.

Benutzertyp

Die folgenden Benutzer können diese Befehle ausführen.

- Precrypto Officer (PRECO)
- Crypto Officer (CO)
- Crypto-Benutzer (Crypto User, CU)

Syntax

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um sich mit der Zwei-Faktor-Authentifizierung (2FA) anzumelden, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen finden Sie unter the section called “Argumente”.

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

```
logoutHSM
```

Beispiele

Diese Beispiele verdeutlichen, wie loginHSM und logoutHSM zum An- und Abmelden von allen HSMs in einem Cluster verwendet werden.

Example : Anmelden bei den HSMs in einem Cluster

Dieser Befehl meldet Sie bei allen HSMs in einem Cluster mit den Anmeldeinformationen eines CO-Benutzers mit dem Namen `admin` und dem Passwort `co12345` an. Die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie eine Verbindung mit den HSMs (in diesem Fall `server 0` und `server 1`) hergestellt haben.

```
aws-cloudhsm>loginHSM CO admin co12345

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)
```

Example : Melden Sie sich mit einem versteckten Passwort an

Dieser Befehl entspricht dem obigen Beispiel, außer dass Sie dieses Mal angeben, dass das System das Passwort verbergen soll.

```
aws-cloudhsm>loginHSM CO admin -hpswd
```

Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Sie geben das Passwort ein, das System versteckt das Passwort und die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie eine Verbindung zu den HSMs hergestellt haben.

```
Enter password:

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)

aws-cloudhsm>
```

Example : Abmelden von einem HSM

Dieser Befehl meldet Sie von den HSMs ab, bei denen Sie zurzeit angemeldet sind (in diesem Fall `server 0` und `server 1`). Die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie von den HSMs getrennt wurden.

```
aws-cloudhsm>logoutHSM

logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
```

Argumente

Geben Sie die Argumente in der Reihenfolge ein, die im Syntaxdiagramm angegeben ist. Verwenden Sie den `-hpswd`-Parameter, um Ihr Passwort zu maskieren. Um sich mit der Zwei-Faktor-Authentifizierung (2FA) anzumelden, verwenden Sie den `-2fa`-Parameter und geben Sie einen Dateipfad an. Weitere Informationen zur Arbeit mit 2FA finden Sie unter Verwendung von CMU zur Verwaltung von 2FA.

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user type>

Gibt den Typ des Benutzers an, der sich bei den HSMs anmeldet. Weitere Informationen finden Sie unter Benutzertyp oben.

Erforderlich: Ja

<user name>

Gibt den Benutzernamen des Benutzers an, der sich bei den HSMs anmeldet.

Erforderlich: Ja

<password | -hpswd >

Gibt das Passwort des Benutzers an, der sich bei den HSMs anmeldet. Um Ihr Passwort zu verbergen, verwenden Sie den `-hpswd`-Parameter anstelle des Passworts und folgen Sie der Aufforderung.

Erforderlich: Ja

[-2fa </path/to/authdata>]

Gibt an, dass das System einen zweiten Faktor verwenden soll, um diesen 2FA-fähigen CO-Benutzer zu authentifizieren. Um die erforderlichen Daten für die Anmeldung mit 2FA zu erhalten, fügen Sie einen Pfad zu einem Speicherort im Dateisystem mit einem Dateinamen nach dem `-2fa`-Parameter ein. Weitere Informationen zur Arbeit mit 2FA finden Sie unter Verwendung von CMU zur Verwaltung von 2FA.

Erforderlich: Nein

Verwandte Themen

- Erste Schritte mit cloudhsm_mgmt_util
- Aktivieren des Clusters

registerQuorumPubSchlüssel

Der registerQuorumPubKey-Befehl in cloudhsm_mgmt_util ordnet Benutzern des Hardware Security Module (HSM) asymmetrischen RSA-2048-Schlüsselpaaren zu. Sobald Sie HSM-Benutzer mit Schlüsseln verknüpft haben, können diese Benutzer den privaten Schlüssel verwenden, um Quorumanfragen zu genehmigen, und der Cluster kann den registrierten öffentlichen Schlüssel verwenden, um zu überprüfen, ob die Signatur vom Benutzer stammt. Weitere Informationen zur Quorum-Authentifizierung finden Sie unter Quorum-Authentifizierung verwalten (M-von-N-Zugriffskontrolle).

> **Tip**
>
> In der AWS CloudHSM-Dokumentation wird die Quorum-Authentifizierung manchmal als M von N (MoFN) bezeichnet, was bedeutet, dass mindestens M Genehmiger von insgesamt N Genehmigern vorhanden sind.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (Crypto Officers, CO)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

Beispiele

Dieses Beispiel zeigt, wie Crypto Officers (CO) als Genehmiger für Quorum-Authentifizierungsanfragen mit registerQuorumPubKey registriert werden können. Um diesen Befehl auszuführen, benötigen Sie ein asymmetrisches RSA-2048-Schlüsselpaar, ein signiertes Token und ein unsigniertes Token. Weitere Informationen zu diesen Anforderungen finden Sie unter the section called “Argumente”.

Example : Registrieren eines HSM-Benutzers für die Quorumauthentifizierung

In diesem Beispiel wird ein CO namens `quorum_officer` als Genehmiger für die Quorumauthentifizierung registriert.

```
aws-cloudhsm> registerQuorumPubKey CO <quorum_officer> </path/to/quorum_officer.token>
 </path/to/quorum_officer.token.sig> </path/to/quorum_officer.pub>

**************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
******************************************************************

Do you want to continue(y/n)?y
registerQuorumPubKey success on server 0(10.0.0.1)
```

Der letzte Befehl verwendet den Befehl ListUsers, um zu überprüfen, ob `quorum_officer` als MoFN-Benutzer registriert ist.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                      MofnPubKey
  LoginFailureCnt         2FA
        1               PCO             admin                           NO
          0               NO
        2               AU              app_user                        NO
          0               NO
        3               CO              quorum_officer                 YES
          0               NO
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

<user-type>

Gibt den Benutzertyp an. Dieser Parameter muss angegeben werden.

Ausführliche Informationen zu den Benutzertypen in einem HSM finden Sie unter Grundlegendes zu HSM-Benutzern.

Zulässige Werte:

- CO: Verschlüsselungsverantwortliche können Benutzer, aber keine Schlüssel verwalten.

Erforderlich: Ja

<user-name>

Gibt einen Anzeigenamen für den Benutzer an. Die maximale Länge beträgt 31 Zeichen. Das einzige zulässige Sonderzeichen ist ein Unterstrich (_).

Sie können den Namen eines Benutzers nach der Erstellung nicht mehr ändern. In cloudhsm_mgmt_util-Befehlen muss bei Benutzertyp und Passwort die Groß- und Kleinschreibung beachtet werden, beim Benutzernamen nicht.

Erforderlich: Ja

<registration-token>

Gibt den Pfad zu einer Datei an, die ein unsigniertes Registrierungstoken enthält. Kann beliebige Daten mit einer maximalen Dateigröße von 245 Byte enthalten. Weitere Informationen zum Erstellen eines unsignierten Registrierungstokens finden Sie unter Registrierungstoken erstellen und signieren.

Erforderlich: Ja

<signed-registration-token>

Gibt den Pfad zu einer Datei an, die den vom SHA256_PKCS-Mechanismus signierten Hash des Registrierungstokens enthält. Weitere Informationen finden Sie unter Registrierungstoken erstellen und signieren.

Erforderlich: Ja

<public-key>

Gibt den Pfad zu einer Datei an, die den öffentlichen Schlüssel eines asymmetrischen RSA-2048-Schlüsselpaars enthält. Verwenden Sie den privaten Schlüssel, um das Registrierungstoken zu signieren. Weitere Informationen finden Sie unter Erstellen eines RSA-Schlüsselpaars.

Erforderlich: Ja

> **Note**
>
> Der Cluster verwendet denselben Schlüssel für die Quorum-Authentifizierung und für die Zwei-Faktor-Authentifizierung (2FA). Das bedeutet, dass Sie einen Quorumschlüssel für einen Benutzer, für den 2FA mit registerQuorumPubKey aktiviert ist, nicht rotieren können. Um den Schlüssel zu rotieren, müssen Sie changePswd verwenden. Weitere Informationen zur Verwendung von Quorum-Authentifizierung und 2FA finden Sie unter Quorum-Authentifizierung und 2FA.

Verwandte Themen

- Erstellen eines RSA-Schlüsselpaares
- Erstellen und signieren Sie ein Registrierungstoken
- Registrieren Sie den öffentlichen Schlüssel beim HSM
- Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle)
- Quorum-Authentifizierung und 2FA
- listUsers

server

Wenn Sie einen Befehl in cloudhsm_mgmt_util eingeben, wirkt sich dieser Befehl normalerweise auf alle HSMs im angegebenen Cluster aus (globaler Modus). Unter bestimmten Umständen müssen Sie auf einem einzelnen HSM Befehle ausgeben. Sollte beispielsweise die automatische Synchronisierung fehlschlagen, müssen Sie möglicherweise Schlüssel und Benutzer auf einem HSM synchronisieren, um die Konsistenz im Cluster zu wahren. Mit dem Befehl server in der cloudhsm_mgmt_util können Sie den Servermodus aufrufen und direkt mit einer bestimmten HSM-Instance interagieren.

Nach erfolgreicher Initiierung wird die `aws-cloudhsm>`-Eingabeaufforderung durch die `server>`-Eingabeaufforderung ersetzt.

Um den Servermodus zu beenden, verwenden Sie den Befehl `exit`. Nach erfolgreicher Beendigung kehren Sie zur cloudhsm_mgmt_util-Eingabeaufforderung zurück.

Bevor Sie einen cloudhsm_mgmt_util-Befehl ausführen, müssen Sie cloudhsm_mgmt_util starten.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer.

Voraussetzungen

Um den Servermodus aufrufen zu können, muss Ihnen die Servernummer des Ziel-HSM bekannt sein. Die Servernummern werden in der Trace-Ausgabe aufgelistet, die von der cloudhsm_mgmt_util

bei der Initiierung generiert wird. Servernummern werden in der gleichen Reihenfolge zugewiesen, in der die HSMs in der Konfigurationsdatei angezeigt werden. In diesem Beispiel gehen wir davon aus, dass `server 0` der Server ist, der dem gewünschten HSM entspricht.

Syntax

So starten Sie den Servermodus:

```
server <server-number>
```

So beenden Sie den Servermodus:

```
server> exit
```

Beispiel

Dieser Befehl ruft den Servermodus auf einem HSM mit der Servernummer `0` auf.

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

Um den Servermodus zu beenden, verwenden Sie den Befehl exit.

```
server0> exit
```

Argumente

```
server <server-number>
```

<server-number>

Gibt die Servernummer des Ziel-HSM an.

Erforderlich: Ja

Für den Befehl `exit` gibt es keine Argumente.

Verwandte Themen

- syncKey
- createUser
- deleteUser

setAttribute

Der Befehl setAttribute in cloudhsm_mgmt_util ändert den Wert der Attribute label, encrypt, decrypt, wrap und unwrap eines Schlüssels in den HSMs. Sie können auch den Befehl setAttribute in key_mgmt_util verwenden, um einen Sitzungsschlüssel in einen dauerhaften Schlüssel umzuwandeln. Sie können nur die Attribute der Schlüssel ändern, deren Eigentümer Sie sind.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Crypto-Benutzer (Crypto User, CU)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
setAttribute <key handle> <attribute id>
```

Beispiel

Das folgende Beispiel zeigt, wie die Entschlüsselungsfunktionen eines symmetrischen Schlüssels deaktiviert werden. Sie können einen Befehl wie diesen verwenden, um einen Verpackungsschlüssel

zu konfigurieren, der andere Schlüssel verpacken und entpacken, aber nicht Daten verschlüsseln oder entschlüsseln kann.

Der erste Schritt ist das Erstellen des Verpackungsschlüssels. Dieser Befehl verwendet genSymKey in key_mgmt_util, um einen symmetrischen 256-Bit-AES-Schlüssel zu generieren. Die Ausgabe zeigt, dass der neue Schlüssel das Schlüssel-Handle 14 aufweist.

```
$  genSymKey -t 31 -s 32 -l aes256

Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Als nächstes werden wir den aktuellen Wert des Entschlüsselungsattributs bestätigen. Um die Attribut-ID des Entschlüsselungsattributs abzurufen, verwenden Sie listAttributes. Die Ausgabe zeigt, dass die Konstante, die das OBJ_ATTR_DECRYPT-Attribut darstellt, 261 lautet. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

```
aws-cloudhsm> listAttributes

Following are the possible attribute values for getAttribute:

      OBJ_ATTR_CLASS                  = 0
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
      OBJ_ATTR_TRUSTED                = 134
      OBJ_ATTR_KEY_TYPE               = 256
      OBJ_ATTR_ID                     = 258
      OBJ_ATTR_SENSITIVE              = 259
      OBJ_ATTR_ENCRYPT                = 260
      OBJ_ATTR_DECRYPT                = 261
      OBJ_ATTR_WRAP                   = 262
      OBJ_ATTR_UNWRAP                 = 263
      OBJ_ATTR_SIGN                   = 264
      OBJ_ATTR_VERIFY                 = 266
      OBJ_ATTR_DERIVE                 = 268
      OBJ_ATTR_LOCAL                  = 355
      OBJ_ATTR_MODULUS                = 288
```

```
      OBJ_ATTR_MODULUS_BITS           = 289
      OBJ_ATTR_PUBLIC_EXPONENT        = 290
      OBJ_ATTR_VALUE_LEN              = 353
      OBJ_ATTR_EXTRACTABLE            = 354
      OBJ_ATTR_NEVER_EXTRACTABLE      = 356
      OBJ_ATTR_ALWAYS_SENSITIVE       = 357
      OBJ_ATTR_DESTROYABLE            = 370
      OBJ_ATTR_KCV                    = 371
      OBJ_ATTR_WRAP_WITH_TRUSTED      = 528
      OBJ_ATTR_WRAP_TEMPLATE          = 1073742353
      OBJ_ATTR_UNWRAP_TEMPLATE        = 1073742354
      OBJ_ATTR_ALL                    = 512
```

Um den aktuellen Wert des Entschlüsselungsattributs für Schlüssel 14 aufzurufen, verwendet der nächste Befehl getAttribute in cloudhsm_mgmt_util.

Die Ausgabe zeigt, dass der Wert des Entschlüsselungsattributs auf beiden HSMs im Cluster true (1) beträgt.

```
aws-cloudhsm> getAttribute 14 261

Attribute Value on server 0(10.0.0.1):
OBJ_ATTR_DECRYPT
0x00000001

Attribute Value on server 1(10.0.0.2):
OBJ_ATTR_DECRYPT
0x00000001
```

Dieser Befehl verwendet setAttribute, um den Wert des Entschlüsselungsattributs (Attribut 261) des Schlüssels 14 auf `0` zu ändern. Dies deaktiviert die Entschlüsselungsfunktion des Schlüssels.

Die Ausgabe zeigt, dass der Befehl in beiden HSMs im Cluster erfolgreich ausgeführt wurde.

```
aws-cloudhsm> setAttribute 14 261 0
************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************
```

```
Do you want to continue(y/n)? y
setAttribute success on server 0(10.0.0.1)
setAttribute success on server 1(10.0.0.2)
```

Der letzte Befehl wiederholt den Befehl getAttribute. Es wird wieder das Entschlüsselungsattribut (Attribut 261) des Schlüssels 14 abgerufen.

Dieses Mal zeigt die Ausgabe, dass der Wert des Entschlüsselungsattributs auf beiden HSMs im Cluster false (0) beträgt.

```
aws-cloudhsm>getAttribute 14 261
Attribute Value on server 0(10.0.3.6):
OBJ_ATTR_DECRYPT
0x00000000

Attribute Value on server 1(10.0.1.7):
OBJ_ATTR_DECRYPT
0x00000000
```

Argumente

```
setAttribute <key handle> <attribute id>
```

<key-handle>

Gibt das Schlüssel-Handle eines Schlüssels an, dessen Eigentümer Sie sind. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Um das Schlüsselhandle eines Schlüssels zu erhalten, verwenden Sie findKey in key_mgmt_util. Um die Benutzer eines Schlüssels zu finden, verwenden Sie getKeyInfo.

Erforderlich: Ja

<attribute id>

Gibt die Konstante an, die das Attribut darstellt, das Sie ändern möchten. Sie können nur jeweils ein Attribut in den einzelnen Befehlen angeben. Verwenden Sie listAttributes, um die Attribute und deren Ganzzahlwerte abzurufen. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Zulässige Werte:

- 3 – OBJ_ATTR_LABEL.
- 134 – OBJ_ATTR_TRUSTED.
- 260 – OBJ_ATTR_ENCRYPT.
- 261 – OBJ_ATTR_DECRYPT.
- 262 – OBJ_ATTR_WRAP.
- 263 – OBJ_ATTR_UNWRAP.
- 264 – OBJ_ATTR_SIGN.
- 266 – OBJ_ATTR_VERIFY.
- 268 – OBJ_ATTR_DERIVE.
- 370 – OBJ_ATTR_DESTROYABLE.
- 528 – OBJ_ATTR_WRAP_WITH_TRUSTED.
- 1073742353 – OBJ_ATTR_WRAP_TEMPLATE.
- 1073742354 – OBJ_ATTR_UNWRAP_TEMPLATE.

Erforderlich: Ja

Verwandte Themen

- setAttribute in key_mgmt_util
- getAttribute
- listAttributes
- Schlüsselattributreferenz

quit

Der quit-Befehl in der Datei cloudhsm_mgmt_util beendet die Datei cloudhsm_mgmt_util. Jeder beliebige Benutzer kann unabhängig vom Typ diesen Befehl verwenden.

Bevor Sie einen cloudhsm_mgmt_util-Befehl ausführen, müssen Sie cloudhsm_mgmt_util starten.

Benutzertyp

Die folgenden Benutzer können diesen Befehl ausführen.

- Alle Benutzer. Sie müssen nicht angemeldet sein, um diesen Befehl auszuführen.

Syntax

```
quit
```

Beispiel

Dieser Befehl beendet cloudhsm_mgmt_util. Nach dem erfolgreichen Abschluss wechseln Sie zurück zu Ihrer regulären Befehlszeile. Dieser Befehl hat keine Ausgabeparameter.

```
aws-cloudhsm> quit

disconnecting from servers, please wait...
```

Verwandte Themen

- Erste Schritte mit cloudhsm_mgmt_util

shareKey

Mit dem Befehl shareKey in cloudhsm_mgmt_util werden Schlüssel, die Sie besitzen, für andere Crypto-Benutzer freigegeben bzw. deren Freigabe wird aufgehoben. Nur der Schlüsselbesitzer kann einen Schlüssel freigeben und die Freigabe aufheben. Sie können einen Schlüssel auch freigeben, wenn Sie ihn erstellen.

Benutzer, für die der Schlüssel freigegeben ist, können ihn in kryptografischen Vorgänge verwenden, ihn jedoch nicht löschen, exportieren, freigeben bzw. seine Freigabe aufheben oder seine Attribute ändern. Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, muss das Quorum alle Operationen genehmigen, die den Schlüssel freigeben bzw. seine Freigabe aufheben.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Crypto-Benutzer (Crypto User, CU)

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

Benutzertyp: Crypto-Benutzer (CU)

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

Beispiel

Die folgenden Beispiele zeigen, wie Sie mit shareKey-Schlüssel, die Sie besitzen, für andere Crypto-Benutzer freigeben bzw. ihre Freigabe aufheben.

Example : Freigeben eines Schlüssels

Dieses Beispiel verwendet shareKey, um einen privaten ECC-Schlüssel, den der aktuelle Benutzer besitzt, für andere Crypto-Benutzer in den HSMs freizugeben. Öffentliche Schlüssel stehen allen Benutzern des HSM zur Verfügung, sodass Sie diese weder freigeben noch deren Freigabe aufheben können.

Der erste Befehl verwendet getKeyInfo, um die Benutzerinformationen für den Schlüssel 262177, einem privaten ECC-Schlüssel in den HSMs, abzurufen.

Die Ausgabe zeigt, dass der Schlüssel 262177 im Besitz des Benutzers 3, aber nicht freigegeben ist.

```
aws-cloudhsm>getKeyInfo 262177

Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3

Key Info on server 1(10.0.3.6):

        Token/Flash Key,

        Owned by user 3
```

In diesem Beispiel wird shareKey verwendet, um den Schlüssel 262177 mit dem Benutzer 4 zu teilen, einem anderen Crypto-Benutzer auf den HSMs. Das letzte Argument verwendet den Wert 1, um eine Freigabeoperation anzugeben.

Die Ausgabe zeigt, dass die Operation in beiden HSMs im Cluster erfolgreich ausgeführt wurde.

```
aws-cloudhsm>shareKey 262177 4 1
**************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
******************************************************************

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.3.10)
shareKey success on server 1(10.0.3.6)
```

Um zu überprüfen, ob die Operation erfolgreich war, wird im Beispiel der erste Befehl getKeyInfo wiederholt.

Die Ausgabe zeigt, dass der Schlüssel 262177 jetzt für den Benutzer 4 freigegeben ist.

```
aws-cloudhsm>getKeyInfo 262177

Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
Key Info on server 1(10.0.3.6):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):
```

```
                    4
```

Example : Aufheben der Freigabe eines Schlüssels

In diesem Beispiel wird gezeigt, wie die Freigabe für einen symmetrischen Schlüssel aufgehoben wird, d. h., ein Crypto-Benutzer wird aus der Liste der für den Schlüssel freigegebenen Benutzer entfernt.

Dieser Befehl verwendet shareKey, um den Benutzer 4 aus der Liste der für den Schlüssel 6 freigegebenen Benutzer zu entfernen. Das letzte Argument verwendet den Wert 0, um eine Operation zum Aufheben der Freigabe anzugeben.

Die Ausgabe zeigt, dass der Befehl in beiden HSMs erfolgreich ausgeführt wurde. Dies hat zur Folge, dass der Benutzer 4 den Schlüssel 6 nicht mehr in kryptografischen Operationen verwenden kann.

```
aws-cloudhsm>shareKey 6 4 0
************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*************************************************************

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.3.10)
shareKey success on server 1(10.0.3.6)
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

<key-handle>

Gibt das Schlüssel-Handle eines Schlüssels an, dessen Eigentümer Sie sind. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Um das Schlüsselhandle eines Schlüssels zu erhalten, verwenden Sie findKey in key_mgmt_util. Zur Bestätigung, dass sich der Schlüssel in Ihrem Besitz befindet, führen Sie getKeyInfo aus.

Erforderlich: Ja

<user id>

Gibt die Benutzer-ID des Crypto-Benutzers (CU) an, für den Sie den Schlüssel freigeben bzw. dessen Freigabe aufheben. Verwenden Sie zur Suche nach der Benutzer-ID eines Benutzers den Befehl listUsers.

Erforderlich: Ja

<share 1 or unshare 0>

Geben Sie zum Freigeben des Schlüssels für den angegebenen Benutzer 1 ein. Um die Freigabe des Schlüssels aufzuheben, d. h. zum Entfernen des angegebenen Benutzers aus der Liste der gemeinsamen Benutzer für den Schlüssel, geben Sie `0` ein.

Erforderlich: Ja

Verwandte Themen

- getKeyInfo

syncKey

Sie können den syncKey-Befehl in cloudhsm_mgmt_util verwenden, um Schlüssel zwischen HSM-Instances innerhalb eines Clusters, oder über mehrere geklonte Cluster hinweg manuell zu synchronisieren. Im Allgemeinen müssen Sie diesen Befehl nicht verwenden, da HSM-Instances in einem Cluster Schlüssel automatisch synchronisieren. Allerdings muss die Synchronisierung von Schlüsseln für geklonte Cluster manuell durchgeführt werden. Geklonte Cluster werden normalerweise in verschiedenen AWS Regionen erstellt, um die globale Skalierung und Disaster Recovery-Prozesse zu vereinfachen.

Sie können syncKey nicht verwenden, um Schlüssel für beliebige Cluster zu synchronisieren: einer der Cluster muss aus einer Sicherung des anderen erstellt worden sein. Außerdem müssen beide Cluster über konsistente CO- und CU-Anmeldeinformationen verfügen, damit die Operation erfolgreich durchgeführt werden kann. Weitere Informationen finden Sie unter HSM-Benutzer.

Für die Verwendung syncKey müssen Sie zunächst eine AWS CloudHSM Konfigurationsdatei erstellen, die ein HSM aus dem Quellcluster und eines aus dem Zielcluster angibt. Auf diese Weise kann cloudhsm_mgmt_util die Verbindung mit beiden HSM-Instances herstellen. Verwenden Sie diese Konfigurationsdatei, um cloudhsm_mgmt_util zu starten. Dann melden Sie sich mit den

Anmeldeinformationen eines CO oder CU an, der die Schlüssel besitzt, die Sie synchronisieren möchten.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (Crypto Officers, CO)
- Crypto-Benutzer (Crypto User, CU)

> **Note**
>
> COs können syncKey für jeden Schlüssel verwenden, während CUs diesen Befehl nur für Schlüssel verwenden können, die sie besitzen. Weitere Informationen finden Sie unter the section called “Grundlegendes zu HSM-Benutzern”.

Voraussetzungen

Bevor Sie beginnen, müssen Sie den `key handle` des Schlüssels im Quell-HSM kennen, der mit dem Ziel-HSM synchronisiert werden soll. Um den `key handle` zu suchen, verwenden Sie den listUsers-Befehl für das Auflisten aller Kennungen für benannte Benutzer. Verwenden Sie dann den findAllKeysBefehl, um alle Schlüssel zu finden, die zu einem bestimmten Benutzer gehören.

Außerdem müssen Sie die den Quell- und Ziel-HSMs zugewiesenen `server IDs` kennen, die in der Ablaufverfolgungsausgabe angezeigt werden, die von cloudhsm_mgmt_util bei der Initiierung zurückgegeben wurde. Diese werden in der gleichen Reihenfolge zugewiesen, in der die HSMs in der Konfigurationsdatei angezeigt werden.

Folgen Sie den Anweisungen in CMU über geklonte Cluster hinweg verwenden und initialisieren Sie cloudhsm_mgmt_util mit der neuen Konfigurationsdatei. Anschließend wechseln Sie in den Servermodus im Quell-HSM, indem Sie den server-Befehl ausführen.

Syntax

> **Note**
>
> Um syncKey auszuführen, wechseln Sie zuerst in den Servermodus im HSM, der den zu synchronisierenden Schlüssel enthält.

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

Benutzertyp: Crypto-Benutzer (CU)

```
syncKey <key handle> <destination hsm>
```

Beispiel

Führen Sie den Server-Befehl aus, um sich beim Quell-HSM anzumelden und in den Servermodus zu wechseln. In diesem Beispiel gehen wir davon aus, dass `server 0` der Quell-HSM ist.

```
aws-cloudhsm> server 0
```

Führen Sie jetzt den syncKey-Befehl aus. In diesem Beispiel gehen wir davon aus, dass der Schlüssel `261251` mit `server 1` synchronisiert werden soll.

```
aws-cloudhsm> syncKey 261251 1
syncKey success
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
syncKey <key handle> <destination hsm>
```

<key handle>

Gibt den Key-Handle des zu synchronisierenden Schlüssels an. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Um das Schlüssel-Handle eines Schlüssels abzurufen, verwenden Sie, findAllKeyswährend Sie an einem HSM-Server angemeldet sind.

Erforderlich: Ja

<destination hsm>

Gibt die Anzahl der Server an, zu denen Sie einen Schlüssel synchronisieren.

Erforderlich: Ja

Verwandte Themen

- listUsers
- findAllKeys
- Cluster in CLI beschreiben
- server

syncUser

Sie können den syncUser Befehl in cloudhsm_mgmt_util verwenden, um Crypto-Benutzer (CUs) oder Crypto Officers (COs) zwischen HSM-Instanzen innerhalb eines Clusters oder zwischen geklonten Clustern manuell zu synchronisieren. AWS CloudHSM synchronisiert Benutzer nicht automatisch. Im Allgemeinen verwalten Sie Benutzer im globalen Modus, sodass alle HSMs in einem Cluster gemeinsam aktualisiert werden. Möglicherweise müssen Sie syncUser verwenden, wenn ein HSM versehentlich desynchronisiert wurde (z. B. aufgrund von Passwortänderungen) oder wenn Sie Benutzeranmeldeinformationen über geklonte Cluster hinweg rotieren möchten. Geklonte Cluster werden normalerweise in verschiedenen AWS Regionen erstellt, um die globale Skalierung und Disaster Recovery-Prozesse zu vereinfachen.

Vor dem Ausführen eines CMU-Befehls müssen Sie die CMU starten und sich beim HSM anmelden. Stellen Sie sicher, dass Sie sich mit einem Benutzertyp anmelden, der die Befehle ausführen kann, die Sie verwenden möchten.

Wenn Sie HSMs hinzufügen oder löschen, aktualisieren Sie die Konfigurationsdateien für CMU. Andernfalls wirken sich die vorgenommenen Änderungen möglicherweise nicht auf alle HSMs im Cluster aus.

Benutzertyp

Die folgenden Benutzertypen können diesen Befehl ausführen.

- Verschlüsselungsverantwortliche (Crypto Officers, CO)

Voraussetzungen

Bevor Sie beginnen, müssen Sie das `user ID` des Benutzers auf dem Quell-HSM kennen, das mit dem Ziel-HSM synchronisiert werden soll. Um das `user ID` zu finden, verwenden Sie den Befehl listUsers. Mit diesem Befehl können Sie alle Benutzer auf den HSMs in einem Cluster auflisten.

Außerdem müssen Sie die den Quell- und Ziel-HSMs zugewiesenen `server ID` kennen, die in der Ablaufverfolgungsausgabe angezeigt werden, die von cloudhsm_mgmt_util bei der Initiierung zurückgegeben wurde. Diese werden in der gleichen Reihenfolge zugewiesen, in der die HSMs in der Konfigurationsdatei angezeigt werden.

Wenn Sie HSMs zwischen geklonten Clustern synchronisieren, folgen Sie den Anweisungen in CMU zwischen geklonten Clustern verwenden und initialisieren Sie cloudhsm_mgmt_util mit der neuen Konfigurationsdatei.

Wenn Sie zur Ausführung von syncUser bereit sind, wechseln Sie auf dem Quell-HSM mit dem Befehl server in den Servermodus.

Syntax

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
syncUser <user ID> <server ID>
```

Beispiel

Führen Sie den Befehl server aus, um sich am Quell-HSM anzumelden und in den Servermodus zu wechseln. In diesem Beispiel gehen wir davon aus, dass `server 0` der Quell-HSM ist.

```
aws-cloudhsm> server 0
```

Führen Sie nun den Befehl syncUser aus. In diesem Beispiel gehen wir davon aus, dass der Benutzer 6 der zu synchronisierende Benutzer und `server 1` der Ziel-HSM ist.

```
server 0> syncUser 6 1
ExtractMaskedObject: 0x0 !
InsertMaskedObject: 0x0 !
syncUser success
```

Argumente

Da dieser Befehl keine benannten Parameter hat, müssen Sie die Argumente in der im Syntaxdiagramm angegebenen Reihenfolge eingeben.

```
syncUser <user ID> <server ID>
```

<user ID>

Gibt die ID des zu synchronisierenden Benutzers an. Sie können in jedem Befehl nur einen Benutzer angeben. Um die ID eines Benutzers zu erhalten, verwenden Sie listUsers.

Erforderlich: Ja

<server ID>

Gibt die Servernummer des HSM an, mit dem Sie einen Benutzer synchronisieren.

Erforderlich: Ja

Verwandte Themen

- listUsers
- Cluster in CLI beschreiben
- server

Schlüsselverwaltungsdienstprogramm (KMU)

Das Key Management Utility (KMU) ist ein Befehlszeilentool, das Crypto-Benutzern (CU) hilft, Schlüssel auf den Hardware-Sicherheitsmodulen (HSM) zu verwalten. Ein KMU umfasst mehrere Befehle, die Schlüssel erzeugen, löschen, importieren und exportieren, Attribute abrufen und setzen, Schlüssel finden und kryptografische Operationen durchführen.

KMU und CMU sind Teil der Client-SDK-3-Suite.

Informationen zum schnellen Einstieg finden Sie unter Erste Schritte mit key_mgmt_util. Ausführliche Informationen zu den Befehlen finden Sie unter key_mgmt_util-Befehlsreferenz. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Um key_mgmt_util unter Linux zu verwenden, stellen Sie eine Verbindung zur Client-Instance her. Beachten Sie dann Den AWS CloudHSM Client installieren und konfigurieren (Linux). Wenn Sie Windows verwenden, siehe Installieren und Konfigurieren des AWS CloudHSM-Clients (Windows).

Themen

- Erste Schritte mit key_mgmt_util
- Den AWS CloudHSM Client installieren und konfigurieren (Linux)

- Installieren und Konfigurieren des AWS CloudHSM-Clients (Windows)
- key_mgmt_util-Befehlsreferenz

Erste Schritte mit key_mgmt_util

AWS CloudHSM beinhaltet zwei Befehlszeilen-Tools in der AWS CloudHSM-Clientsoftware. Das cloudhsm_mgmt_util-Tool enthält Befehle zum Verwalten von HSM-Benutzern. Das key_mgmt_util-Tool enthält Befehle zum Verwalten von Schlüsseln. Weitere Informationen über die ersten Schritte mit dem key_mgmt_util-Befehlszeilen-Tool finden Sie in den folgenden Themen.

Themen

- Einrichten von key_mgmt_util
- Grundlegende Nutzung von key_mgmt_util

Wenn Sie eine Fehlermeldung oder ein unerwartetes Ergebnis für einen Befehl erhalten, finden Sie in den Fehlerbehebung für AWS CloudHSM-Themen Hilfe. Einzelheiten zu den Befehlen key_mgmt_util finden Sie unter key_mgmt_util-Befehlsreferenz.

Einrichten von key_mgmt_util

Führen Sie die folgende Einrichtung durch, bevor Sie key_mgmt_util verwenden.

Starten des AWS CloudHSM-Clients

Bevor Sie key_mgmt_util verwenden, starten Sie den AWS CloudHSM-Client. Der Client ist ein Daemon, der eine verschlüsselte End-to-End-Kommunikation zwischen Ihrer Client-Instanz und den HSMs in Ihrem Cluster herstellt. Das key_mgmt_util-Tool verwendet die Client-Verbindung für die Kommunikation mit den HSMs in Ihrem Cluster. Ohne diese Verbindung funktioniert key_mgmt_util nicht.

Den AWS CloudHSM-Client starten

Verwenden Sie den folgenden Befehl, um den AWS CloudHSM-Client zu starten.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

key_mgmt_util starten

Nach dem Starten des AWS CloudHSM-Clients verwenden Sie den folgenden Befehl, um key_mgmt_util zu starten.

Amazon Linux

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Amazon Linux 2

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

CentOS 7

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

CentOS 8

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

RHEL 7

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

RHEL 8

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Ubuntu 16.04 LTS

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Ubuntu 18.04 LTS

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Windows

```
c:\Program Files\Amazon\CloudHSM> .\key_mgmt_util.exe
```

Wenn key_mgmt_util ausgeführt wird, ändert sich die Eingabeaufforderung in `Command:`.

Wenn der Befehl fehlschlägt, also z. B. die Meldung `Daemon socket connection error` zurückgegeben wird, versuchen Sie eine Aktualisierung Ihrer Konfigurationsdatei.

Grundlegende Nutzung von key_mgmt_util

Weitere Informationen zur grundlegenden Nutzung des key_mgmt_util Tools finden Sie in den folgenden Themen.

Themen

- Anmelden an den HSMs
- Abmelden von HSMs
- Beenden von key_mgmt_util

Anmelden an den HSMs

Verwenden Sie den Befehl loginHSM, um sich bei den HSMs anzumelden. Mit dem folgenden Befehl melden Sie sich als Crypto-Benutzer (CU, Crypto User) mit dem Namen `example_user` an. Die Ausgabe zeigt eine erfolgreiche Anmeldung für alle drei HSMs in dem Cluster.

```
Command:  loginHSM -u CU -s example_user -p <PASSWORD>
Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Nachfolgend finden Sie die allgemeine Syntax für den Befehl loginHSM.

```
Command:  loginHSM -u <USER TYPE> -s <USERNAME> -p <PASSWORD>
```

Abmelden von HSMs

Verwenden Sie den Befehl logoutHSM, um sich von den HSMs abzumelden.

```
Command:  logoutHSM
```

```
Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Beenden von key_mgmt_util

Verwenden Sie den Befehl exit, um key_mgmt_util zu beenden.

```
Command:  exit
```

Den AWS CloudHSM Client installieren und konfigurieren (Linux)

Um mit dem HSM in Ihrem AWS CloudHSM Cluster zu interagieren, benötigen Sie die AWS CloudHSM Client-Software für Linux. Sie sollten die Software auf der Linux-EC2-Client-Instance installieren, die Sie zuvor erstellt haben. Sie können auch einen Client für Windows installieren. Weitere Informationen finden Sie unter Installieren und Konfigurieren des AWS CloudHSM-Clients (Windows).

Aufgaben

- Installieren Sie den AWS CloudHSM Client und die Befehlszeilentools
- Bearbeiten der Clientkonfiguration

Installieren Sie den AWS CloudHSM Client und die Befehlszeilentools

Connect zu Ihrer Client-Instance her und führen Sie die folgenden Befehle aus, um den AWS CloudHSM Client und die Befehlszeilentools herunterzuladen und zu installieren.

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb
```

Bearbeiten der Clientkonfiguration

Bevor Sie den AWS CloudHSM Client verwenden können, um eine Verbindung zu Ihrem Cluster herzustellen, müssen Sie die Client-Konfiguration bearbeiten.

Bearbeiten der Clientkonfiguration

1. Kopieren Sie Ihr ausstellendes Zertifikat – das Zertifikat, mit dem Sie das Zertifikat des Clusters signiert haben – an den folgenden Speicherort in der Client-Instance: `/opt/cloudhsm/etc/customerCA.crt`. Sie müssen Root-Berechtigungen für die Client-Instance haben, um Ihr Zertifikat an diesen Speicherort zu kopieren.
2. Verwenden Sie den folgenden Konfigurationsbefehl, um die Konfigurationsdateien für den AWS CloudHSM Client und die Befehlszeilentools zu aktualisieren. Geben Sie dabei die IP-Adresse des HSM in Ihrem Cluster an. Um die IP-Adresse des HSM abzurufen, zeigen Sie Ihren Cluster in der AWS CloudHSM Konsole an oder führen Sie den describe-clusters CLI-Befehl aus. Die IP-Adresse des HSM ist in der Ausgabe des Befehls der Wert des Feldes `EniIp`. Wenn Sie über mehr als ein HSM verfügen, wählen Sie einfach eine die IP-Adresse eines beliebigen HSM aus.

```
sudo /opt/cloudhsm/bin/configure -a <IP address>

Updating server config in /opt/cloudhsm/etc/cloudhsm_client.cfg
Updating server config in /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

3. Wechseln Sie zu Aktivieren des Clusters.

Installieren und Konfigurieren des AWS CloudHSM-Clients (Windows)

Um mit einem HSM in Ihrem AWS CloudHSM-Cluster unter Windows zu arbeiten, benötigen Sie die AWS CloudHSM-Clientsoftware für Windows. Sie sollten die Software auf der Windows Server-Instance installieren, die Sie zuvor erstellt haben.

So installieren (oder aktualisieren) Sie die den neuesten Windows-Client und die Befehlszeilen-Tools

1. Stellen Sie eine Verbindung zur Windows Server-Instance her.
2. Laden Sie die neueste Version (AWSCloudHSMClient-latest.msi) von der Download-Seite herunter.
3. Gehen Sie zu Ihrem Download-Ort und führen Sie das Installationsprogramm (AWSCloudHSMClient-latest.msi) mit administrativen Rechten aus.
4. Folgen Sie den Anweisungen des Installationsprogramms und wählen Sie dann Schließen, wenn das Installationsprogramm abgeschlossen ist.
5. Kopieren Sie das selbstsignierte Ausstellerzertifikat – das Zertifikat, mit dem Sie das Clusterzertifikat signiert haben – in den Ordner `C:\ProgramData\Amazon\CloudHSM`.
6. Führen Sie den folgenden Befehl aus, um die Konfigurationsdateien zu aktualisieren. Beim Aktualisieren muss der Client während der Neukonfiguration angehalten und dann wieder gestartet werden.

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure.exe -a <HSM IP address>
```

7. Wechseln Sie zu Aktivieren des Clusters.

Hinweise:

- Bei der Aktualisierung des Clients werden vorhandene Konfigurationsdateien aus früheren Installationen nicht überschrieben.

- Das AWS CloudHSM-Client-Installationsprogramm für Windows registriert automatisch die Cryptography API: Next Generation (CNG) und den Key Storage Provider (KSP). Um den Client zu deinstallieren, führen Sie das Installationsprogramm erneut aus und folgen Sie den Anweisungen zur Deinstallation.
- Wenn Sie Linux verwenden, können Sie den Linux-Client installieren. Weitere Informationen finden Sie unter Den AWS CloudHSM Client installieren und konfigurieren (Linux).

key_mgmt_util-Befehlsreferenz

Das Befehlszeilen-Tool key_mgmt_util hilft Ihnen bei der Schlüsselverwaltung in den HSMs Ihrer Cluster, einschließlich dem Erstellen, Löschen und Finden von Schlüsseln sowie deren Attributen. Das Tool umfasst mehrere Befehle, die in diesem Thema ausführlich beschrieben werden.

Informationen zum schnellen Einstieg finden Sie unter Erste Schritte mit key_mgmt_util. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz. Weitere Informationen über das cloudhsm_mgmt_util-Befehlszeilentool, das Befehle für die Verwaltung von HSM und Benutzern in Ihrem Cluster umfasst, finden Sie unter CloudHSM-Verwaltungsdienstprogramm (CMU) .

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Um alle key_mgmt_util-Befehle aufzulisten, geben Sie Folgendes ein:

```
Command: help
```

Um Hilfe für einen bestimmten key_mgmt_util-Befehl zu erhalten, geben Sie Folgendes ein:

```
Command: <command-name> -h
```

Um Ihre key_mgmt_util-Sitzung zu beenden, geben Sie Folgendes ein:

```
Command: exit
```

Die folgenden Themen beschreiben Befehle in key_mgmt_util.

> **Note**
>
> Einige Befehle in key_mgmt_util und cloudhsm_mgmt_util haben dieselben Namen. Die Befehle haben jedoch in der Regel eine andere Syntax, eine andere Ausgabe und eine leicht unterschiedliche Funktionalität.

Befehl	Beschreibung
aesWrapUnwrap	Verschlüsselt und entschlüsselt den Inhalt eines Schlüssels in einer Datei.
deleteKey	Löscht einen Schlüssel aus den HSMs.
Error2String	Gibt den Fehler zurück, der dem hexadezimalen key_mgmt_util-Fehlercode entspricht.
exit	Beendet das key_mgmt_util.
exportPrivateKey	Exportiert die Kopie eines privaten Schlüssels aus einem HSM in eine Datei auf einem Datenträger.
exportPubKey	Exportiert eine Kopie eines öffentlichen Schlüssels aus einem HSM in eine Datei.
exSymKey	Exportiert eine Klartextkopie eines symmetrischen Schlüssels aus den HSMs in eine Datei.
extractMaskedObject	Extrahiert einen Schlüssel aus einem HSM als Datei in Form eines maskierten Objekts.
findKey	Sucht nach Schlüsseln anhand des Schlüsselattributwerts.
findSingleKey	Verifiziert, dass ein Schlüssel auf allen HSMs in einem Cluster vorhanden ist.

Befehl	Beschreibung
genDSAKeyPair	Generiert ein Digital Signing Algorithm(DSA)-Schlüsselpaar in Ihren HSMs.
genECCKeyPair	Generiert ein Elliptic Curve Cryptography-Schlüsselpaar (ECC) in Ihren HSMs.
genRSAKeyPair	Generiert ein asymmetrisches RSA-Schlüsselpaar in Ihren HSMs.
genSymKey	Generiert einen symmetrischen Schlüssel in Ihren HSMs
getAttribute	Liest die Attributwerte eines AWS CloudHSM-Schlüssels und schreibt sie in eine Datei.
getCaviumPrivKey	Erstellt eine Fake-Version eines privaten Schlüssels im PEM-Format und exportiert ihn in eine Datei.
getCert	Ruft die Partitionszertifikate eines HSMs ab und speichert sie in einer Datei.
getKeyInfo	Liest die HSM-Benutzer-IDs eines Benutzers, der den Schlüssel verwenden kann. Wird der Schlüssel durch ein Quorum gesteuert, wird die Anzahl der Benutzer in dem Quorum gelesen.
help	Zeigt Hilfeinformationen zu den in key_mgmt_util verfügbaren Befehlen an.
importPrivateKey	Importiert einen privaten Schlüssel in ein HSM.
importPubKey	Importiert einen öffentlichen Schlüssel in ein HSM.

Befehl	Beschreibung
imSymKey	Importiert eine Klartextkopie eines symmetrischen Schlüssels von einer Datei in das HSM.
insertMaskedObject	Fügt ein maskiertes Objekt aus einer Datei auf einem Datenträger in ein HSM ein, das sich in einem dem ursprünglichen Objekt-Cluster zugehörigen Cluster befindet. Zusammengehörige Cluster sind alle aus einer Sicherung des ursprünglichen Clusters erstellten Cluster.
???	Bestimmt, ob eine bestimmte Datei einen echten privaten Schlüssel oder einen gefälschten PEM-Schlüssel enthält.
listAttributes	Zeigt die Attribute eines AWS CloudHSM-Schlüssels sowie die Konstanten auf, die ihn darstellen.
listUsers	Ruft die Benutzer in den HSMs, ihre Benutzertypen und IDs sowie andere Attribute ab.
loginHSM und logoutHSM	Melden Sie sich bei den HSMs in einem Cluster an und ab.
setAttribute	Konvertiert einen Sitzungsschlüssel in einen persistenten Schlüssel.
sign	Erstellen Sie mit einem ausgewählten privaten Schlüssel eine Signatur für eine Datei.
unWrapKey	Importiert einen verpackten (verschlüsselten) Schlüssel von einer Datei in die HSMs.
verify	Prüft, ob ein bestimmter Schlüssel zum Signieren einer bestimmten Datei verwendet wurde.

Befehl	Beschreibung
wrapKey	Exportiert eine verschlüsselte Kopie eines Schlüssels aus dem HSM in eine Datei.

aesWrapUnwrap

Der Befehl aesWrapUnwrap verschlüsselt oder entschlüsselt den Inhalt einer Datei auf einem Datenträger. Mit diesem Befehl können Sie Verschlüsselungsschlüssel ein- und auspacken. Sie können ihn für jede Datei verwenden, die weniger als 4 KB (4096 Byte) Daten enthält.

aesWrapUnwrap verwendet AES Key Wrap. Es verwendet einen AES-Schlüssel im HSM als Ver- und Entpackschlüssel. Dann schreibt er das Ergebnis in eine andere Datei auf dem Datenträger.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
aesWrapUnwrap -h

aesWrapUnwrap -m <wrap-unwrap mode>
              -f <file-to-wrap-unwrap>
              -w <wrapping-key-handle>
              [-i <wrapping-IV>]
              [-out <output-file>]
```

Beispiele

Diese Beispiele zeigen, wie man mit aesWrapUnwrap einen Verschlüsselungsschlüssel in einer Datei verschlüsselt und entschlüsselt.

Example : Verschlüsselungsschlüssel verpacken

Dieser Befehl verwendet aesWrapUnwrap, um einen symmetrischen Dreifach-DES-Schlüssel, der vom HSM im Klartext exportiert wurde, in die `3DES.key`-Datei zu packen. Mit einem ähnlichen Befehl können Sie jeden in einer Datei gespeicherten Schlüssel verpacken.

Der Befehl verwendet den `-m`-Parameter mit dem Wert 1, um den Wrap-Modus festzulegen. Er verwendet den `-w`-Parameter, um einen AES-Schlüssel im HSM (Key-Handle 6) als Wrapping-

Schlüssel anzugeben. Der resultierende verpackte Schlüssel wird in die `3DES.key.wrapped`-Datei geschrieben.

Die Ausgabe zeigt, dass der Befehl erfolgreich war und dass die Operation die Standard-IV verwendet hat, die bevorzugt wird.

```
 Command:  aesWrapUnwrap -f 3DES.key -w 6 -m 1 -out 3DES.key.wrapped

        Warning: IV (-i) is missing.
                 0xA6A6A6A6A6A6A6A6 is considered as default IV
result data:
49 49 E2 D0 11 C1 97 22
17 43 BD E3 4E F4 12 75
8D C1 34 CF 26 10 3A 8D
6D 0A 7B D5 D3 E8 4D C2
79 09 08 61 94 68 51 B7

result written to file 3DES.key.wrapped

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

Example : Verschlüsselungsschlüssel entpacken

Dieses Beispiel zeigt, wie man aesWrapUnwrap verwendet, um einen verpackten (verschlüsselten) Schlüssel in einer Datei zu entpacken (entschlüsseln). Sie können eine solche Operation durchführen, bevor Sie einen Schlüssel in das HSM importieren. Wenn Sie beispielsweise versuchen, mit dem Befehl imSymKey einen verschlüsselten Schlüssel zu importieren, wird ein Fehler zurückgegeben, da der verschlüsselte Schlüssel nicht das Format hat, das für einen solchen Klartext-Schlüssel erforderlich ist.

Der Befehl entpackt den Schlüssel in der Datei `3DES.key.wrapped` und schreibt den Klartext in die Datei `3DES.key.unwrapped`. Der Befehl verwendet den `-m`-Parameter mit dem Wert `0`, um den Unwrap-Modus festzulegen. Er verwendet den `-w`-Parameter, um einen AES-Schlüssel im HSM (Key-Handle 6) als Wrapping-Schlüssel anzugeben. Der resultierende verpackte Schlüssel wird in die `3DES.key.unwrapped`-Datei geschrieben.

```
 Command:  aesWrapUnwrap -m 0 -f 3DES.key.wrapped -w 6 -out 3DES.key.unwrapped

        Warning: IV (-i) is missing.
                 0xA6A6A6A6A6A6A6A6 is considered as default IV
result data:
```

```
14 90 D7 AD D6 E4 F5 FA
A1 95 6F 24 89 79 F3 EE
37 21 E6 54 1F 3B 8D 62

result written to file 3DES.key.unwrapped

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-m

Gibt den Modus an. Um den Dateiinhalt zu verpacken (verschlüsseln), geben Sie 1 ein. Um den Dateiinhalt zu entpacken (entschlüsseln), geben Sie 0 ein.

Erforderlich: Ja

-f

Gibt die zu verpackende Datei an. Geben Sie eine Datei ein, die weniger als 4 KB (4096 Byte) Daten enthält. Diese Operation wurde entwickelt, um Verschlüsselungsschlüssel ein- und auszupacken.

Erforderlich: Ja

-w

Gibt den Wrapping-Schlüssel an. Geben Sie das Schlüssel-Handle eines AES-Schlüssels auf dem HSM ein. Dieser Parameter muss angegeben werden. Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Um einen Wrapping Key zu erstellen, verwenden Sie genSymKey, um einen AES-Schlüssel (Typ 31) zu erzeugen.

Erforderlich: Ja

-i

Gibt einen alternativen Initialwert (IV) für den Algorithmus an. Verwenden Sie den Standardwert, es sei denn, es gibt eine spezielle Bedingung, die eine Alternative erfordert.

Standard: `0xA6A6A6A6A6A6A6A6`. Der Standardwert ist in der Spezifikation des AES Key Wrap-Algorithmus definiert.

Erforderlich: Nein

-out

Gibt einen alternativen Namen für die Ausgabedatei an, die den verpackten oder entpackten Schlüssel enthält. Die Voreinstellung ist `wrapped_key` (für Wrap-Operationen) und `unwrapped_key` (für Unwrap-Operationen) im lokalen Verzeichnis.

Wenn die Datei vorhanden ist, wird diese ohne Warnung von aesWrapUnwrap überschrieben. Wenn der Befehl fehlschlägt, erstellt aesWrapUnwrap eine Ausgabedatei ohne Inhalt.

Voreinstellung: Für Verpacken: `wrapped_key`. Zum Entpacken: `unwrapped_key`.

Erforderlich: Nein

Verwandte Themen

- exSymKey
- imSymKey
- unWrapKey
- wrapKey

deleteKey

Mit dem Befehl deleteKey in key_mgmt_util wird ein Schlüssel aus dem HSM gelöscht. Sie können nur jeweils einen Schlüssel löschen. Das Löschen eines Schlüssels in einem Schlüsselpaar hat keine Auswirkungen auf den anderen Schlüssel in diesem Paar.

Nur der Schlüsselbesitzer kann einen Schlüssel löschen. Benutzer, für die der Schlüssel freigegeben ist, können ihn in kryptografischen Vorgängen verwenden, aber nicht löschen.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
deleteKey -h
```

```
deleteKey -k
```

Beispiele

Diese Beispiele verdeutlichen, wie mit deleteKey Schlüssel von Ihren HSMs gelöscht werden können.

Example : Löschen eines Schlüssels

Dieser Befehl löscht den Schlüssel mit dem Schlüssel-Handle 6. Wenn der Befehl erfolgreich ist, gibt deleteKey eine Erfolgsmeldung von jedem HSM im Cluster zurück.

```
Command: deleteKey -k 6

        Cfm3DeleteKey returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : Löschen eines Schlüssels (Fehler)

Wenn der Befehl fehlschlägt, da kein Schlüssel über das angegebenen Schlüssel-Handle verfügt, gibt deleteKey eine Fehlermeldung aufgrund eines ungültigen Objekt-Handles zurück.

```
Command: deleteKey -k 252126

        Cfm3FindKey returned: 0xa8 : HSM Error: Invalid object handle is passed to this
 operation

        Cluster Error Status
        Node id 1 and err state 0x000000a8 : HSM Error: Invalid object handle is passed
 to this operation
        Node id 2 and err state 0x000000a8 : HSM Error: Invalid object handle is passed
 to this operation
```

Wenn der Befehl fehlschlägt, da der aktuelle Benutzer nicht der Eigentümer des Schlüssels ist, gibt der Befehl eine „Zugriff verweigert'-Fehlermeldung zurück.

```
Command:  deleteKey -k 262152
```

```
Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied.
```

Parameter

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle des zu löschenden Schlüssels an. Um die Schlüssel-Handles der Schlüssel im HSM zu finden, verwenden Sie findKey.

Erforderlich: Ja

Verwandte Themen

- findKey

Error2String

Der Error2String-Hilfsbefehl in key_mgmt_util gibt den Fehler zurück, der einem hexadezimalen Fehlercode von key_mgmt_util entspricht. Verwenden Sie diesen Befehl bei der Fehlerbehebung von Befehlen und Skripten.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
Error2String -h

Error2String -r <response-code>
```

Beispiele

Diese Beispiele zeigen, wie Sie mit Error2String den Fehlerstring für einen key_mgmt_util-Fehlercode abrufen können.

Example ; Anfordern einer Fehlerbeschreibung

Dieser Befehl ruft die Fehlerbeschreibung für den `0xdb`-Fehlercode auf. Die Beschreibung erklärt, dass ein Anmeldeversuch bei key_mgmt_util fehlgeschlagen ist, da der Benutzer nicht über den richtigen Benutzertypen verfügt. Nur Crypto-Benutzer (CU) können sich bei key_mgmt_util anmelden.

```
        Command:  Error2String -r 0xdb

        Error Code db maps to HSM Error: Invalid User Type.
```

Example : Finden des Fehlercodes

Dieses Beispiel zeigt, wo der Fehlercode in einem key_mgmt_util-Fehler gefunden werden kann. Der Fehlercode `0xc6` wird nach der Zeichenfolge angezeigt: `Cfm3`*`command-name`* `returned:` .

In diesem Beispiel gibt getKeyInfo an, dass der aktuelle Benutzer (Benutzer 4) den Schlüssel in Verschlüsselungsvorgängen verwenden kann. Wenn der Benutzer jedoch deleteKey verwendet, um den Schlüssel zu löschen, gibt der Befehl den Fehlercode `0xc6` zurück.

```
        Command:  deleteKey -k 262162

        Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied

        Cluster Error Status

        Command:  getKeyInfo -k 262162

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 1 user(s):

                4
```

Wen Ihnen der Fehler `0xc6` zurückgegeben wird, können Sie einen Error2String-Befehl wie diesen verwenden, um den Fehler nachzuschlagen. In diesem Fall schlug der Befehl `deleteKey` aufgrund eines „Zugriff verweigert“-Fehlers fehl, da der Schlüssel für den aktuellen Benutzer freigegeben

wurde, der Eigentümer aber ein anderer Benutzer ist. Nur die Eigentümer eines Schlüssels haben die Berechtigung, einen Schlüssel zu löschen.

```
        Command:  Error2String -r 0xa8

        Error Code c6 maps to HSM Error: Key Access is denied
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-r

Gibt einen hexadezimalen Fehlercode an. Die hexadezimalen Anzeige `0x` ist erforderlich.

Erforderlich: Ja

exit

Der exit-Befehl in key_mgmt_util beendet key_mgmt_util. Nach erfolgreicher Beendigung kehren Sie zur Standard-Befehlszeile zurück.

Bevor Sie den Befehl key_mgmt_util ausführen, müssen Sie key_mgmt_util starten.

Syntax

```
exit
```

Parameter

Für diesen Befehl gibt es keine Parameter.

Verwandte Themen

- key_mgmt_util starten

exportPrivateKey

Der exportPrivateKey-Befehl in key_mgmt_util exportiert einen asymmetrischen privaten Schlüssel von einem HSM in eine Datei. Das HSM erlaubt keinen direkten Export von Schlüsseln im Klartext. Der Befehl umschließt den privaten Schlüssel mit einem von Ihnen angegebenen AES-Wrapping-Schlüssel, entschlüsselt die umschlossenen Bytes und kopiert den privaten Klartext-Schlüssel in eine Datei.

Der exportPrivateKey-Befehl entfernt den Schlüssel nicht aus dem HSM, ändert nicht seine Schlüsselattribute und hindert Sie nicht daran, den Schlüssel für weitere kryptografische Operationen zu verwenden. Sie können den gleichen Schlüssel mehrmals exportieren.

Sie können nur private Schlüssel exportieren, die den `OBJ_ATTR_EXTRACTABLE`-Attributwert 1 haben. Sie müssen einen AES-Wrapping-Schlüssel angeben, der `OBJ_ATTR_WRAP`- und `OBJ_ATTR_DECRYPT`-Attributwerte 1 hat. Mit dem Befehl getAttribute können Sie nach den Attributen eines Schlüssels suchen.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
exportPrivateKey -h

exportPrivateKey -k <private-key-handle
                 -w <wrapping-key-handle>
                 -out <key-file>
                 [-m <wrapping-mechanism>]
                 [-wk <wrapping-key-file>]
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit exportPrivateKey einen privaten Schlüssel aus einem HSM exportieren.

Example : Exportieren eines privaten Schlüssels

Dieser Befehl exportiert einen privaten Schlüssel mit dem Handle 15 anhand eines Verpackungsschlüssels mit dem Handle 16 in eine PEM-Datei mit dem Namen `exportKey.pem`. Wird der Befehl erfolgreich ausgeführt, gibt exportPrivateKey eine Erfolgsmeldung zurück.

```
Command: exportPrivateKey -k 15 -w 16 -out exportKey.pem

Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS

PEM formatted private key is written to exportKey.pem
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-h`

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

`-k`

Gibt das Schlüssel-Handle des zu exportierenden privaten Schlüssels an.

Erforderlich: Ja

`-w`

Gibt das Schlüssel-Handle des Verpackungsschlüssels an. Dieser Parameter muss angegeben werden. Nutzen Sie den Befehl findKey, um Schlüssel-Handles zu suchen.

Ermitteln Sie mithilfe von getAttribute den Wert des `OBJ_ATTR_WRAP`-Attributs (262), um zu bestimmen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann. Einen Verpackungsschlüssel erstellen Sie mit dem Befehl genSymKey, der zum Erstellen eines AES-Schlüssels (Typ 31) verwendet wird.

Wenn Sie den Parameter `-wk` zum Angeben eines externen Entpackungsschlüssels verwenden, wird der `-w`-Verpackungsschlüssel während des Exports zum Verpacken, nicht aber zum Entpacken des Schlüssels verwendet.

Erforderlich: Ja

`-out`

Gibt den Namen der Datei an, in die der exportierte private Schlüssel geschrieben werden soll.

Erforderlich: Ja

`-m`

Gibt den Verpackungsmechanismus an, mit dem der zu exportierende private Schlüssel verpackt werden soll. Der einzige gültige Wert ist `4`. Dieser repräsentiert den `NIST_AES_WRAP mechanism.`-Mechanismus.

Standard: 4 (`NIST_AES_WRAP`)

Erforderlich: Nein

`-wk`

Gibt den Schlüssel an, mit dem der zu exportierende Schlüssel entpackt werden soll. Geben Sie den Pfad und den Namen einer Datei an, die einen Klartext-AES-Schlüssel enthält.

Wenn Sie diesen Parameter einschließen, verwendet exportPrivateKey den Schlüssel in der `-w`-Datei, um den zu exportierenden Schlüssel zu verpacken. Zum Entpacken wird der über den `-wk`-Parameter angegebene Schlüssel verwendet.

Standard: Verwenden Sie den im `-w`-Parameter angegeben Verpackungsschlüssel für das Verpacken und Entpacken von Schlüsseln.

Erforderlich: Nein

Verwandte Themen

- importPrivateKey
- wrapKey
- unWrapKey
- genSymKey

exportPubKey

Der exportPubKey-Befehl in key_mgmt_util exportiert einen öffentlichen Schlüssel in einem HSM in eine Datei. Sie können ihn verwenden, um öffentliche Schlüssel zu exportieren, die Sie in einem HSM erstellt haben. Mit diesem Befehl ist auch das Exportieren öffentlicher Schlüssel möglich, die in ein HSM importiert wurden. Dies betrifft z. B. jene Schlüssel, deren Import über den Befehl importPubKey erfolgt ist.

Die Operation exportPubKey kopiert das Schlüsselmaterial in eine von Ihnen angegebene Datei. Der Befehl entfernt jedoch den Schlüssel nicht aus dem HSM oder ändert dessen Schlüsselattribute. Auch hindert Sie der Befehl nicht daran, den Schlüssel in weiteren kryptografischen Operationen zu verwenden. Sie können den gleichen Schlüssel mehrmals exportieren.

Sie können nur öffentliche Schlüssel exportieren, deren `OBJ_ATTR_EXTRACTABLE`-Wert auf `1` gesetzt ist. Mit dem Befehl getAttribute können Sie nach den Attributen eines Schlüssels suchen.

Vor dem Ausführen eines `key_mgmt_util`-Befehls müssen Sie key_mgmt_util starten und sich beim HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
exportPubKey -h

exportPubKey -k <public-key-handle
             -out <key-file>
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit exportPubKey einen öffentlichen Schlüssel aus einem HSM exportieren.

Example : Exportieren eines öffentlichen Schlüssels

Mit diesem Befehl wird ein öffentlicher Schlüssel mit dem Handle `10` in eine Datei namens `public.pem` exportiert. Wird der Befehl erfolgreich ausgeführt, gibt exportPubKey eine Erfolgsmeldung zurück.

```
Command: exportPubKey -k 10 -out public.pem

PEM formatted public key is written to public.pem

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-h`

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

`-k`

Gibt das Schlüssel-Handle des zu exportierenden öffentlichen Schlüssels an.

Erforderlich: Ja

`-out`

Gibt den Namen der Datei an, in die der exportierte öffentliche Schlüssel geschrieben werden soll.

Erforderlich: Ja

Verwandte Themen

- importPubKey
- Generieren von Schlüsseln

exSymKey

Der exSymKey-Befehl im Tool key_mgmt_util exportiert eine Klartextkopie eines symmetrischen Schlüssels aus dem HSM und speichert sie in einer Datei auf der Festplatte. Um eine verschlüsselte (verpackte) Kopie eines Schlüssels zu exportieren, verwenden Sie wrapKey. Um einen Klartextschlüssel zu importieren, der denen ähnelt, die anhand von `exSymKey` exportiert werden, verwenden Sie imSymKey.

Während des Exportvorgangs verwendet exSymKey einen AES-Schlüssel, den Sie angeben (der wrapping key (Verpackungsschlüssel)), um den zu exportierenden Schlüssel zu packen (verschlüsseln) und anschließend zu entpacken (entschlüsseln). Das Ergebnis des Exportvorgangs ist jedoch ein Klartextschlüssel (entpackter Schlüssel) auf einem Datenträger.

Nur der Eigentümer eines Schlüssels, d. h., der CU-Benutzer, der den Schlüssel erstellt hat, kann ihn exportieren. Benutzer, für die der Schlüssel freigegeben ist, können ihn in kryptografischen Vorgängen verwenden, können ihn aber nicht exportieren.

Der Vorgang exSymKey kopiert die Schlüsselinformationen in eine Datei, die Sie angeben, entfernt den Schlüssel jedoch nicht aus dem HSM, ändert seine key attributes (Schlüsselattribute) nicht

oder verhindert, dass Sie den Schlüssel in kryptografischen Vorgänge verwenden. Sie können den gleichen Schlüssel mehrmals exportieren.

exSymKey exportiert nur symmetrische Schlüssel. Um öffentliche Schlüssel zu exportieren, verwenden Sie exportPubKey. Um private Schlüssel zu exportieren, verwenden Sie exportPrivateKey.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
exSymKey -h

exSymKey -k <key-to-export>
         -w <wrapping-key>
         -out <key-file>
         [-m 4]
         [-wk <unwrapping-key-file> ]
```

Beispiele

Diese Beispiele zeigen, wie Sie mit exSymKey symmetrische Schlüssel, deren Eigentümer Sie sind, von Ihren HSMs exportieren.

Example : Exportieren eines symmetrischen 3DES-Schlüssels

Mit diesem Befehl wird ein symmetrischer Triple DES (3DES)-Schlüssel exportiert (Schlüssel-Handle 7). Er verwendet einen vorhandenen AES-Schlüssel (Schlüssel-Handle 6) im HSM als Verpackungsschlüssel. Anschließend wird der Klartext des 3DES-Schlüssels in die Datei `3DES.key` geschrieben.

Die Ausgabe zeigt, dass der Schlüssel 7 (der 3DES-Schlüssel) erfolgreich verpackt und entpackt und anschließend in die angegebene Datei `3DES.key` geschrieben wurde.

> **Warning**
>
> Obwohl die Ausgabe besagt, dass ein „verpackter symmetrischer Schlüssel“ in die Ausgabedatei geschrieben wurden, enthält die Ausgabedatei einen (entpackten) Klartextschlüssel.

```
        Command: exSymKey -k 7 -w 6 -out 3DES.key

       Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "3DES.key"
```

Example : Exportieren mit einem nur für die aktuelle Sitzung gültigen Verpackungsschlüssel

Das folgende Beispiel zeigt, wie Sie einen Schlüssel verwenden, der nur in dieser Sitzung als Verpackungsschlüssel vorhanden ist. Da der zu exportierende Schlüssel verpackt ist, anschließend sofort entpackt und als Klartext bereitgestellt wird, ist es nicht nötig, den Verpackungsschlüssel aufzubewahren.

Anhand dieser Befehle wird ein AES-Schlüssel mit der Schlüssel-Handle 8 aus dem HSM exportiert. Dabei wird ein AES-Sitzungsschlüssel verwendet, der speziell zu diesem Zweck erstellt wurde.

Der erste Befehl verwendet genSymKey, um einen 256-Bit-AES-Schlüssel zu erstellen. Dabei wird der `-sess`-Parameter zum Erstellen eines Schlüssels verwendet, der nur in der aktuellen Sitzung vorhanden ist.

Die Ausgabe zeigt, dass das HSM den Schlüssel 262168 erstellt.

```
        Command:  genSymKey -t 31 -s 32 -l AES-wrapping-key -sess

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 262168

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Als Nächstes wird im Beispiel sichergestellt, dass es sich bei dem Schlüssel 8, dem zu exportierenden Schlüssel, um einen symmetrischen Schlüssel handelt, der extrahierbar ist. Außerdem wird überprüft, ob der Verpackungsschlüssel, der Schlüssel 262168, ein AES-Schlüssel ist, der nur in dieser Sitzung vorhanden ist. Sie können den Befehl findKey verwenden, aber in diesem Beispiel werden die Attribute beider Schlüssel in Dateien exportiert und anschließend `grep` verwendet, um die relevanten Attributwerte in der Datei zu finden.

Diese Befehle verwenden `getAttribute` mit einem `-a` Wert von `512` (alle), um alle Attribute für die Schlüssel 8 und 262168 aufzurufen. Weitere Informationen zu den Schlüsselattributen finden Sie unter the section called “Schlüsselattributreferenz”.

```
getAttribute -o 8 -a 512 -out attributes/attr_8
getAttribute -o 262168 -a 512 -out attributes/attr_262168
```

Diese Befehle verwenden `grep`, um die Attribute des zu exportierenden Schlüssels (Schlüssel 8) und des nur für die aktuelle Sitzung gültigen Verpackungsschlüssels (Schlüssel 262168) zu überprüfen.

```
    // Verify that the key to be exported is a symmetric key.
    $  grep -A 1 "OBJ_ATTR_CLASS" attributes/attr_8
    OBJ_ATTR_CLASS
    0x04

    // Verify that the key to be exported is extractable.
    $  grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_8
    OBJ_ATTR_EXTRACTABLE
    0x00000001

    // Verify that the wrapping key is an AES key
    $  grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_262168
    OBJ_ATTR_KEY_TYPE
    0x1f

    // Verify that the wrapping key is a session key
    $  grep -A 1 "OBJ_ATTR_TOKEN" attributes/attr_262168
    OBJ_ATTR_TOKEN
    0x00

    // Verify that the wrapping key can be used for wrapping
     $  grep -A 1 "OBJ_ATTR_WRAP" attributes/attr_262168
    OBJ_ATTR_WRAP
    0x00000001
```

Schließlich verwenden wir den Befehl exSymKey, um den Schlüssel 8 mithilfe des Sitzungsschlüssels (Schlüssel 262168) als Verpackungsschlüssel zu exportieren.

Wenn die Sitzung beendet wird, wird der Schlüssel 262168 nicht länger vorhanden sein.

```
        Command:  exSymKey -k 8 -w 262168 -out aes256_H8.key
```

```
        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "aes256_H8.key"
```

Example : Verwenden eines externen Entpackungsschlüssels

Das folgende Beispiel zeigt, wie Sie mit einem externen Entpackungsschlüssel einen Schlüssel aus dem HSM exportieren.

Wenn Sie einen Schlüssel aus dem HSM exportieren, geben Sie einen AES-Schlüssel im HSM als Verpackungsschlüssel an. Standardmäßig wird dieser Schlüssel dazu verwendet, den zu exportierenden Schlüssel zu verpacken und zu entpacken. Sie können jedoch mithilfe des `-wk`-Parameters exSymKey anweisen, zum Entpacken einen externen Schlüssel in einer Datei auf einem Datenträger zu verwenden. Wenn Sie sich dafür entscheiden, verpackt der vom `-w`-Parameter angegebene Schlüssel den Ziel-Schlüssel, und der Schlüssel, der sich in der vom `-wk`-Parameter angegebenen Datei befindet, entpackt den Schlüssel.

Da der Verpackungsschlüssel ein AES-Schlüssel sein muss, und dieser symmetrisch ist, müssen der Verpackungsschlüssel im HSM und der Entpackungsschlüssel auf dem Datenträger über die gleichen Schlüsselinformationen verfügen. Hierzu müssen Sie vor dem Exportvorgang den Verpackungsschlüssel in das HSM importieren oder den Verpackungsschlüssel aus dem HSM exportieren.

In diesem Beispiel wird ein Schlüssel außerhalb des HSM erstellt und in das HSM importiert. Dabei wird eine interne Kopie des Schlüssels verwendet, um den zu exportierenden symmetrischen Schlüssel zu verpacken, und anschließend die Kopie des Schlüssels in der Datei dazu benutzt, um ihn zu entpacken.

Der erste Befehl verwendet OpenSSL, um einen 256-Bit-AES-Schlüssel zu generieren. Der Schlüssel wird in der Datei `aes256-forImport.key` gespeichert. Der OpenSSL-Befehl gibt keine Ausgabe zurück, jedoch können Sie mehrere Befehle verwenden, um zu überprüfen, ob der Vorgang erfolgreich war. Dieses Beispiel verwendet das wc-Tool (Wordcount), das bestätigt, dass die Datei 32 Byte an Daten umfasst.

```
$  openssl rand -out keys/aes256-forImport.key 32
```

```
$ wc keys/aes256-forImport.key
 0  2 32 keys/aes256-forImport.key
```

Dieser Befehl verwendet den Befehl imSymKey, um den AES-Schlüssel aus der Datei `aes256-forImport.key` in das HSM zu importieren. Wenn der Befehl abgeschlossen ist, ist der Schlüssel im HSM mit dem Schlüssel-Handle 262167 und in der Datei `aes256-forImport.key` vorhanden.

```
Command:  imSymKey -f keys/aes256-forImport.key -t 31 -l aes256-imported -w 6

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 262167

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Dieser Befehl verwendet den Schlüssel in einem Exportvorgang. Der Befehl verwendet exSymKey, um den Schlüssel 21, ein 192-Bit-AES-Schlüssel, zu exportieren. Um den Schlüssel zu verpacken, wird der Schlüssel 262167 verwendet. Dabei handelt es sich um die Kopie des Schlüssels, die in das HSM importiert wurde. Um den Schlüssel zu entpacken, werden die gleichen Schlüsselinformationen in der Datei `aes256-forImport.key` verwendet. Nach der Ausführung des Befehls wird der Schlüssel 21 in die Datei `aes192_h21.key` exportiert.

```
        Command:  exSymKey -k 21 -w 262167 -out aes192_H21.key -wk aes256-forImport.key

        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

Wrapped Symmetric Key written to file "aes192_H21.key"
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle des zu exportierenden Schlüssels an. Dieser Parameter muss angegeben werden. Geben Sie das Schlüssel-Handle eines symmetrischen Schlüssels ein, dessen Eigentümer Sie sind. Dieser Parameter muss angegeben werden. Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Verwenden Sie den Befehl getAttribute, um sicherzustellen, dass ein Schlüssel exportiert werden kann, und um den Wert des Attributs `OBJ_ATTR_EXTRACTABLE` abzurufen, der von der Konstanten `354` dargestellt wird. Zudem können Sie nur Schlüssel exportieren, deren Eigentümer Sie sind. Um den Eigentümer eines Schlüssels zu finden, verwenden Sie den Befehl getKeyInfo.

Erforderlich: Ja

-w

Gibt das Schlüssel-Handle des Verpackungsschlüssels an. Dieser Parameter muss angegeben werden. Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Ein Verpackungsschlüssel ist ein Schlüssel im HSM, der zum Verschlüsseln (Verpacken) und zum anschließenden Entschlüsseln (Entpacken) des zu exportierenden Schlüssels verwendet wird. Nur AES-Schlüssel können als Verpackungsschlüssel verwendet werden.

Sie können jeden AES-Schlüssel (in jeder beliebigen Größe) als Verpackungsschlüssel verwenden. Da der Verpackungsschlüssel den Zielschlüssel verpackt und danach sofort entpackt, können Sie einen nur für die aktuelle Sitzung gültigen AES-Schlüssel als Verpackungsschlüssel verwenden. Verwenden Sie getAttribute, um zu überprüfen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann, und um den Wert des `OBJ_ATTR_WRAP`-Attributs abzurufen, der von der Konstanten `262` dargestellt wird. Verwenden Sie genSymKey, um einen Umhüllungsschlüssel und einen AES-Schlüssel (Typ 31) zu erstellen.

Wenn Sie den Parameter `-wk` zum Angeben eines externen Entschlüsselungsschlüssels verwenden, wird der `-w`-Schlüssel während des Exportvorgangs zum Verpacken, aber nicht zum Entpacken des Schlüssels verwendet.

> **Note**
>
> Schlüssel 4 stellt einen nicht unterstützten internen Schlüssel dar. Wir empfehlen, dass Sie einen AES-Schlüssel als Umhüllungsschlüssel verwenden, den Sie erstellen und verwalten.

Erforderlich: Ja

-out

Gibt den Pfad und Namen der Ausgabedatei an. Wenn der Befehl erfolgreich ist, enthält diese Datei den exportierten Schlüssel als Klartext. Wenn die Datei bereits vorhanden ist, überschreibt der Befehl sie ohne Warnung.

Erforderlich: Ja

-m

Gibt den Verpackungsmechanismus an. Der einzige gültige Wert ist `4`, der den `NIST_AES_WRAP`-Mechanismus darstellt.

Erforderlich: Nein

Standard: `4`

-wk

Verwenden Sie den AES-Schlüssel in der angegebenen Datei, um den Schlüssel, der exportiert wird, zu entpacken. Geben Sie den Pfad und den Namen einer Datei an, die einen Klartext-AES-Schlüssel enthält.

Wenn Sie diesen Parameter angeben, verwendet `exSymKey` den Schlüssel im HSM, der durch den `-w`-Parameter angegeben wird, um den zu exportierenden Schlüssel zu verpacken, und verwendet den Schlüssel in der `-wk`-Datei, um ihn zu entpacken. Die Parameterwerte `-w` und `-wk` müssen durch denselben Klartextschlüssel aufgelöst werden.

Erforderlich: Nein

Standard: Verwenden Sie den Verpackungsschlüssel auf dem HSM zum Entpacken.

Verwandte Themen

- genSymKey
- imSymKey
- wrapKey

extractMaskedObject

Der extractMaskedObject-Befehl in key_mgmt_util extrahiert einen Schlüssel aus einem HSM und speichert ihn in einer Datei als maskiertes Objekt. Maskierte Objekte sind geklonte Objekte, die nur verwendet werden können, nachdem sie anhand des Befehls insertMaskedObject wieder in den ursprünglichen Cluster eingefügt wurden. Sie können ein maskiertes Objekt nur in den Cluster einfügen, aus dem es erstellt wurde (oder in eine geklonte Version dieses Clusters). Dies umfasst alle geklonten Clusterversionen, die durch eine regionsübergreifende Sicherungskopie und die anschließende Verwendung dieser Sicherung zum Erstellen eines neuen Clusters generiert wurden.

Maskierte Objekte sind eine effiziente Möglichkeit zum Auslagern und Synchronisieren von Schlüsseln, auch von nicht extrahierbaren Schlüsseln (d. h. Schlüssel mit einem `OBJ_ATTR_EXTRACTABLE`-Wert von `0`). Auf diese Weise können Schlüssel zwischen Clustern verschiedener Regionen auf sichere Weise synchronisiert werden, ohne die AWS CloudHSM-Konfigurationsdatei aktualisieren zu müssen.

> **Important**
>
> Nach dem Einfügen werden maskierte Objekte entschlüsselt und mit einem Schlüssel-Handle versehen, das sich vom Schlüssel-Handle des ursprünglichen Schlüssels unterscheidet. Ein maskiertes Objekt enthält alle dem ursprünglichen Schlüssel zugeordneten Metadaten, einschließlich Attributen, Eigentümerschaft, der gemeinsamen Nutzung von Informationen und Quorum-Einstellungen. Wenn Sie in einer Anwendung Schlüssel zwischen Clustern synchronisieren müssen, verwenden Sie stattdessen syncKey im cloudhsm_mgmt_util.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich beim HSM anmelden. Der Befehl extractMaskedObject kann entweder vom CU, der den Schlüssel besitzt, oder von einem beliebigen CO genutzt werden.

Syntax

```
extractMaskedObject -h

extractMaskedObject -o <object-handle>
                    -out <object-file>
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie extractMaskedObject verwenden, um einen Schlüssel als maskiertes Objekt aus einem HSM zu extrahieren.

Example : Extrahieren eines maskierten Objekts

Dieser Befehl extrahiert in einem HSM aus einem Schlüssel mit dem Handle `524295` ein maskiertes Objekt und speichert es als Datei mit dem Namen `maskedObj`. Wird der Befehl erfolgreich ausgeführt, gibt extractMaskedObject eine Erfolgsmeldung zurück.

```
Command: extractMaskedObject -o 524295 -out maskedObj

Object was masked and written to file "maskedObj"

        Cfm3ExtractMaskedObject returned: 0x00 : HSM Return: SUCCESS
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-h`

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

`-o`

Gibt das Handle des Schlüssels an, der als maskiertes Objekt extrahiert werden soll.

Erforderlich: Ja

`-out`

Gibt den Namen der Datei an, in die das maskierte Objekt gespeichert werden soll.

Erforderlich: Ja

Verwandte Themen

- insertMaskedObject
- syncKey

- Regionsübergreifendes Kopieren einer Sicherung
- Erstellen eines AWS CloudHSM-Clusters aus einer vorherigen Sicherung

findKey

Verwenden Sie den findKey-Befehl in key_mgmt_util, um nach Schlüsseln anhand der Werte der Schlüsselattribute zu suchen. Wenn ein Schlüssel allen Kriterien entspricht, die Sie festlegen, gibt findKey das Schlüssel-Handle zurück. Ohne Parameter gibt findKey die Schlüssel-Handles aller Schlüssel zurück, die Sie im HSM verwenden können. Verwenden Sie für die Suche der Attributwerte eines bestimmten Schlüssels den Befehl getAttribute.

Wie alle key_mgmt_util-Befehle ist findKey benutzerspezifisch. Der Befehl gibt nur die Schlüssel zurück, die der aktuelle Benutzer in kryptografischen Operationen verwenden kann. Dies umfasst Schlüssel, die der aktuelle Benutzer besitzt und Schlüssel, die für ihn freigegeben wurden.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
findKey -h

findKey [-c <key class>]
        [-t <key type>]
        [-l <key label>]
        [-id <key ID>]
        [-sess (0 | 1)]
        [-u <user-ids>]
        [-m <modulus>]
        [-kcv <key_check_value>]
```

Beispiele

Die Beispiele verdeutlichen, wie Sie mit findKey Schlüssel in Ihren HSMs suchen und identifizieren.

Example : Suchen nach allen Schlüsseln

Mit diesem Befehl werden alle Schlüssel für den aktuellen Benutzer im HSM gesucht. Die Ausgabe umfasst Schlüssel, die der Benutzer besitzt und die für ihn freigegeben sind, sowie alle öffentlichen Schlüssel in den HSMs.

Verwenden Sie zum Abrufen der Attribute eines Schlüssels mit einem bestimmten Schlüssel-Handle den Befehl getAttribute. Um festzustellen, ob der aktuelle Benutzer einen bestimmten Schlüssel besitzt oder teilt, verwenden Sie getKeyInfo oder findAllKeys in cloudhsm_mgmt_util.

```
Command: findKey

Total number of keys present 13

 number of keys matched from start index 0::12
6, 7, 524296, 9, 262154, 262155, 262156, 262157, 262158, 262159, 262160, 262161, 262162

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example : Suchen von Schlüsseln nach Typ, Benutzer und Sitzung

Mit diesem Befehl werden persistente AES-Schlüssel gesucht, die vom aktuellen Benutzer und von Benutzer 3 verwendet werden können. (Benutzer 3 kann möglicherweise andere Schlüssel verwenden, die der aktuelle Benutzer nicht sehen kann.)

```
Command: findKey -t 31 -sess 0 -u 3
```

Example : Suchen von Schlüsseln nach Klasse und Beschriftung

Mit diesem Befehl werden alle öffentlichen Schlüssel für den aktuellen Benutzer mit der Beschriftung `2018-sept` gesucht.

```
Command: findKey -c 2 -l 2018-sept
```

Example : Suchen von RSA-Schlüsseln nach Modulus

Mit diesem Befehl können Sie RSA-Schlüssel (Typ 0) für den aktuellen Benutzer suchen, die mit dem Modulus in der Datei `m4.txt` erstellt wurden.

```
Command: findKey -t 0 -m m4.txt
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-t

Sucht Schlüssel des angegebenen Typs. Geben Sie die Konstante ein, die die Schlüsselklasse darstellt. Geben Sie beispielsweise `-t 21` ein, um 3DES-Schlüssel zu suchen.

Zulässige Werte:

- 0: RSA
- 1: DSA
- 3: EC
- 16: GENERIC_SECRET
- 18: RC4
- 21: Triple DES (3DES)
- 31: AES

Erforderlich: Nein

-c

Sucht Schlüssel in der angegebenen Klasse. Geben Sie die Konstante ein, die die Schlüsselklasse darstellt. Geben Sie beispielsweise `-c 2` ein, um öffentliche Schlüssel zu suchen.

Gültige Werte für jeden Schlüsseltyp:

- 2: Public. Diese Klasse enthält die öffentlichen Schlüssel der öffentlich-privaten Schlüsselpaare.
- 3: Private. Diese Klasse enthält die privaten Schlüssel der öffentlich-privaten Schlüsselpaare.
- 4: Secret. Diese Klasse enthält alle symmetrischen Schlüssel.

Erforderlich: Nein

-l

Sucht Schlüssel mit der angegebenen Beschriftung. Geben Sie die genaue Beschriftung ein. Sie können keine Platzhalter oder regulären Ausdrücke im Wert `--l` verwenden.

Erforderlich: Nein

-id

Sucht den Schlüssel mit der angegebenen ID. Geben Sie die genaue ID-Zeichenfolge ein. Sie können keine Platzhalter oder regulären Ausdrücke im Wert `-id` verwenden.

Erforderlich: Nein

-sess

Sucht Schlüssel nach Sitzungsstatus. Geben Sie für die Suche nach Schlüsseln, die nur in der aktuellen Sitzung gültig sind, `1` ein. Geben Sie für die Suche nach persistenten Schlüsseln `0` ein.

Erforderlich: Nein

-u

Sucht Schlüssel, die von den angegebenen Benutzern und dem aktuellen Benutzer gemeinsam verwendet werden. Geben Sie eine durch Kommas getrennte Liste der HSM-Benutzer-IDs ein, wie etwa `-u 3` oder `-u 4,7`. Verwenden Sie zum Suchen der IDs der Benutzer in einem HSM den Befehl listUsers.

Wenn Sie eine Benutzer-ID angeben, gibt findKey den Schlüssel für diesen Benutzer zurück. Wenn Sie mehrere Benutzer-IDs angeben, gibt findKey die Schlüssel zurück, die alle angegebenen Benutzer verwenden können.

Da findKey nur Schlüssel zurückgibt, die der aktuelle Benutzer verwenden kann, sind die `-u`-Ergebnisse immer identisch oder eine Teilmenge der Schlüssel des aktuellen Benutzers. Um alle Schlüssel abzurufen, die einem beliebigen Benutzer gehören oder mit ihm geteilt werden, können Crypto Officers (COs) findAllKeys in cloudhsm_mgmt_util verwenden.

Erforderlich: Nein

-m

Sucht Schlüssel, die mit dem RSA-Modulus in der angegebenen Datei erstellt wurden. Geben Sie den Pfad zur Datei ein, in der der Modulus gespeichert ist.

-m gibt die Binärdatei an, die den RSA-Modul enthält, mit dem abgeglichen werden soll (optional).

Erforderlich: Nein

-kcv

Sucht Schlüssel mit dem angegebenen Schlüsselprüfwert.

Der Schlüsselprüfwert (KCV) ist ein 3-Byte-Hash oder eine Prüfsumme eines Schlüssels, der generiert wird, wenn das HSM einen Schlüssel importiert oder generiert. Sie können einen KCV auch außerhalb des HSM berechnen, z. B. nachdem Sie einen Schlüssel exportiert haben. Anschließend können Sie die KCVs vergleichen, um die Identität und Integrität des Schlüssels zu bestätigen. Um den KCV eines Schlüssels abzurufen, verwenden Sie getAttribute.

AWS CloudHSM verwendet die folgende Standardmethode, um einen Schlüsselprüfwert zu generieren:

- Symmetrische Schlüssel: Die ersten 3 Byte des Ergebnisses der Verschlüsselung eines Nullblocks mit dem Schlüssel.
- Asymmetrische Schlüsselpaare: Die ersten 3 Byte des SHA-1-Hashs des öffentlichen Schlüssels.
- HMAC-Schlüssel: KCV für HMAC-Schlüssel wird derzeit nicht unterstützt.

Erforderlich: Nein

Ausgabe

Die Ausgabe des Befehls findKey enthält die Gesamtanzahl der übereinstimmenden Schlüssel und ihrer Schlüssel-Handles.

```
        Command:  findKey
Total number of keys present 10

 number of keys matched from start index 0::9
6, 7, 8, 9, 10, 11, 262156, 262157, 262158, 262159

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Verwandte Themen

- findSingleKey
- getKeyInfo
- getAttribute

- findAllKeys in cloudhsm_mgmt_util
- Schlüsselattributreferenz

findSingleKey

Der findSingleKey-Befehl im Tool key_mgmt_util prüft, ob ein Schlüssel auf allen HSMs im Cluster vorhanden ist.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
findSingleKey -h

findSingleKey -k <key-handle>
```

Beispiel

Example

Mit diesem Befehl wird überprüft, ob der Schlüssel 252136 auf allen drei HSMs im Cluster vorhanden ist.

```
Command: findSingleKey -k 252136
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle eines Schlüssels im HSM an. Dieser Parameter muss angegeben werden.

Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Erforderlich: Ja

Verwandte Themen

- findKey
- getKeyInfo
- getAttribute

genDSAKeyPair

Der genDSAKeyPair-Befehl in key_mgmt_util tool generiert ein Digital Signing Algorithm (DSA)-Schlüsselpaar in Ihren HSMs. Sie müssen die Modullänge angeben, der Befehl generiert den Modulwert. Sie können auch eine ID zuweisen, den Schlüssel für andere HSM-Benutzer freigeben sowie nicht extrahierbare Schlüssel und Schlüssel, die bei Sitzungsende ablaufen, erstellen. Wenn der Befehl erfolgreich ausgeführt wurde, werden die Schlüssel-Handles zurückgegeben, die das HSM den öffentlichen und privaten ECC-Schlüsseln zuweist. Sie können die Schlüssel-Handles nutzen, damit die Schlüssel für andere Befehle identifizierbar sind.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

> **Tip**
>
> Um die Attribute eines von Ihnen erstellten Schlüssels wie Typ, Länge, Bezeichnung und ID zu finden, verwenden Sie GetAttribute. Um nach den Schlüsseln für einen bestimmten Benutzer zu suchen, verwenden Sie getKeyInfo. Verwenden Sie FindKey, um Schlüssel anhand ihrer Attributwerte zu finden.

Syntax

```
genDSAKeyPair -h
```

```
genDSAKeyPair -m <modulus length>
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

Beispiele

Diese Beispiele verdeutlichen, wie mit genDSAKeyPair ein DSA-Schlüsselpaar erstellt wird.

Example : Erstellen eines DSA-Schlüsselpaares

Mit diesem Befehl wird ein DSA-Schlüsselpaar mit einem DSA-Label erstellt. Die Ausgabe zeigt, dass das Schlüssel-Handle des öffentlichen Schlüssels 19 ist, und dass das Schlüssel-Handle des privaten Schlüssels 21 lautet.

```
Command: genDSAKeyPair -m 2048 -l DSA

        Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 19    private key handle: 21

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : Erstellen eines DSA-Schlüsselpaares, das nur für eine Sitzung gültig ist

Mit diesem Befehl wird ein DSA-Schlüsselpaar erstellt, das nur für die aktuelle Sitzung gültig ist. Der Befehl weist zusätzlich zu der erforderlichen (nicht eindeutigen) Bezeichnung eine eindeutige ID von DSA_temp_pair zu. Sie können ein solches Schlüsselpaar erstellen, um ein nur für diese Sitzung gültiges Token zu signieren und zu überprüfen. Die Ausgabe zeigt, dass das Schlüssel-Handle des öffentlichen Schlüssels 12 ist, und dass das Schlüssel-Handle des privaten Schlüssels 14 lautet.

```
Command: genDSAKeyPair -m 2048 -l DSA-temp -id DSA_temp_pair -sess

        Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS
```

```
        Cfm3GenerateKeyPair:    public key handle: 12    private key handle: 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Um zu überprüfen, ob das Schlüsselpaar nur in der Sitzung vorhanden ist, verwenden Sie den `-sess`-Parameter von findKey mit einem Wert von 1 (true).

```
  Command: findKey -sess 1

  Total number of keys present 2

 number of keys matched from start index 0::1
12, 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example : Erstellen eines freigegebenen, unextrahierbaren DSA-Schlüsselpaares

Mit diesem Befehl wird ein DSA-Schlüsselpaar erstellt. Der private Schlüssel wird mit drei anderen Benutzern geteilt und kann nicht aus dem HSM exportiert werden. Öffentliche Schlüssel können von jedem Benutzer verwendet und immer extrahiert werden.

```
        Command:  genDSAKeyPair -m 2048 -l DSA -id DSA_shared_pair -nex -u 3,5,6

        Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 11    private key handle: 19

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : Erstellen eines Quorum-kontrollierten Schlüsselpaares

Mit diesem Befehl wird ein DSA-Schlüsselpaar mit der Bezeichnung `DSA-mV2` erstellt. Der Befehl verwendet den `-u`-Parameter, um den privaten Schlüssel für die Benutzer 4 und 6 freizugeben. Er verwendet den `-m_value`-Parameter, damit für alle kryptografischen Vorgänge, die den privaten

Schlüssel verwenden, ein Quorum aus mindestens zwei Genehmigungen erforderlich ist. Der Befehl verwendet auch den `-attest`-Parameter, um die Integrität der Firmware zu überprüfen, auf der das Schlüsselpaar generiert wird.

Die Ausgabe zeigt, dass der Befehl einen öffentlichen Schlüssel mit dem Schlüssel-Handle 12 und einen privaten Schlüssel mit dem Schlüssel-Handle 17 generiert, und dass die Bescheinigungsprüfung auf der Cluster-Firmware erfolgreich bestanden wurde.

```
Command:  genDSAKeyPair -m 2048 -l DSA-mV2 -m_value 2 -u 4,6 -attest

Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 12    private key handle: 17

Attestation Check : [PASS]

Cluster Error Status
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Dieser Befehl verwendet getKeyInfo für den privaten Schlüssel (Schlüssel-Handle 17). Die Ausgabe bestätigt, dass es sich bei dem Eigentümer um den aktuellen Benutzer (Benutzer 3) handelt, und dass der Schlüssel mit den Benutzern 4 und 6 (und keinen anderen) geteilt wird. Die Ausgabe zeigt zudem, dass die Quorum-Authentifizierung aktiviert ist und die Quorumgröße 2 beträgt.

```
Command:  getKeyInfo -k 17

Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

Owned by user 3

also, shared to following 2 user(s):

        4
        6
 2 Users need to approve to use/manage this key
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-m

Gibt die Länge des Moduls in Bits an. Der einzige gültige Wert ist `2048`.

Erforderlich: Ja

-l

Gibt eine benutzerdefinierte Bezeichnung für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein. Dieselbe Bezeichnung gilt für beide Schlüssel im Paar. Die maximal zulässige Größe für `label` beträgt 127 Zeichen.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-id

Gibt einen benutzerdefinierten Bezeichner für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge. Die von Ihnen angegebene ID gilt für beide Schlüssel im Paar.

Standard : Kein ID-Wert.

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl der HSMs an, in denen der Schlüssel synchronisiert wird, bevor der Wert des Parameters `-timeout` verfällt. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Zur Beschleunigung Ihres Prozesses legen Sie den Wert von `min_srv` auf weniger als die Anzahl

der HSMs im Cluster fest und stellen einen niedrigen Zeitüberschreitungswert ein. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-m_value

Gibt die Anzahl der Benutzer an, die jede kryptografische Operation genehmigen müssen, die den privaten Schlüssel des Paares verwendet. Geben Sie einen Wert von `0` bis `8` ein.

Dieser Parameter legt eine Quorum-Authentifizierungsanforderung für den privaten Schlüssel fest. Der Standardwert, `0`, deaktiviert die Quorum-Authentifizierungsfunktion für den Schlüssel. Wenn die Quorumauthentifizierung aktiviert ist, muss die angegebene Anzahl von Benutzern ein Token signieren, um kryptografische Operationen, bei denen der private Schlüssel verwendet wird, sowie Operationen, bei denen der private Schlüssel gemeinsam genutzt oder die gemeinsame Nutzung aufgehoben wird, zu genehmigen.

Um den `m_value` eines Schlüssels zu finden, verwenden Sie getKeyInfo.

Dieser Parameter ist nur gültig, wenn der `-u`-Parameter im Befehl das Schlüsselpaar für ausreichend Benutzer freigibt, um die `m_value`-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

-nex

Macht den privaten Schlüssel nicht extrahierbar. Der generierte private Schlüssel kann nicht aus dem HSM exportiert werden. Öffentliche Schlüssel sind immer extrahierbar.

Standard: Sowohl der öffentliche als auch der private Schlüssel im Schlüsselpaar können extrahiert werden.

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie setAttribute.

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv`-Parameter angegebenen Anzahl von HSMs synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-u

Teilt den privaten Schlüssel des Paares mit den angegebenen Benutzern. Dieser Parameter erteilt anderen HSM-Crypto-Benutzern die Berechtigung zur Verwendung des privaten Schlüssels in kryptographischen Vorgängen. Öffentliche Schlüssel können von jedem Benutzer verwendet werden, ohne sie zu teilen.

Geben Sie eine durch Kommas getrennte Liste der HSM-Benutzer-IDs ein, wie etwa `-u 5,6`. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. Verwenden Sie listUsers, um im HSM nach den HSM-Benutzer-IDs von CUs zu suchen. Nutzen Sie zum Freigeben oder zum Aufheben der Freigabe vorhandener Schlüssel shareKey in cloudhsm_mgmt_util.

Standard: Nur der aktuelle Benutzer kann den privaten Schlüssel verwenden.

Erforderlich: Nein

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

Verwandte Themen

- genRSAKeyPair
- genSymKey
- genECCKeyPair

genECCKeyPair

Der `genECCKeyPair`- Befehl im Tool key_mgmt_util erzeugt ein ECC-Schlüsselpaar (Elliptic Curve Cryptography) in Ihren HSMs. Beim Ausführen des `genECCKeyPair`-Befehls müssen Sie die elliptische Kurvenkennung und eine Beschriftung für das Schlüsselpaar angeben. Sie können den privaten Schlüssel auch mit anderen CU-Benutzern teilen sowie nicht extrahierbare Schlüssel, Quorum-kontrollierte Schlüssel und Schlüssel, die bei Sitzungsende ablaufen, erstellen. Wenn der Befehl erfolgreich ausgeführt wurde, werden die Schlüssel-Handle zurückgegeben, die das HSM zu den öffentlichen und privaten ECC-Schlüsseln zuweist. Sie können die Schlüssel-Handles nutzen, damit die Schlüssel für andere Befehle identifizierbar sind.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

> **Tip**
>
> Um die Attribute eines von Ihnen erstellten Schlüssels wie Typ, Länge, Bezeichnung und ID zu finden, verwenden Sie GetAttribute. Um nach den Schlüsseln für einen bestimmten Benutzer zu suchen, verwenden Sie getKeyInfo. Verwenden Sie FindKey, um Schlüssel anhand ihrer Attributwerte zu finden.

Syntax

```
genECCKeyPair -h

genECCKeyPair -i <EC curve id>
```

```
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

Beispiele

Diese Beispiele verdeutlichen, wie mit genECCKeyPair ECC-Schlüsselpaare in Ihren HSMs erstellt werden.

Example : Erstellen und Untersuchen eines ECC-Schlüsselpaars

Mit diesem Befehl wird ein ECC-Schlüsselpaar mithilfe einer elliptischen Kurve vom Typ NID_secp384r1 und der Bezeichnung `ecc14` erstellt. Die Ausgabe zeigt, dass das Schlüssel-Handle des privaten Schlüssels 262177 ist und das Schlüssel-Handle des öffentlichen Schlüssels 262179. Die Bezeichnung gilt sowohl für den öffentlichen als auch den privaten Schlüssel.

```
Command: genECCKeyPair -i 14 -l ecc14

        Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 262179    private key handle: 262177

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Nach dem Generieren des Schlüssels können Sie seine Attribute untersuchen. Verwenden Sie getAttribute, um alle Attribute (dargestellt durch die Konstante 512) des neuen privaten ECC-Schlüssels in die Datei `attr_262177` zu schreiben.

```
Command: getAttribute -o 262177 -a 512 -out attr_262177
got all attributes of size 529 attr cnt 19
Attributes dumped into attr_262177
```

```
        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

Verwenden Sie dann den `cat`-Befehl, um den Inhalt der `attr_262177`-Attributdatei anzuzeigen. Die Ausgabe zeigt, dass der Schlüssel ein privater Ellipsenkurvenschlüssel ist, der zum Signieren, aber nicht zum Verschlüsseln, Entschlüsseln, Verpacken, Entpacken oder Verifizieren verwendet werden kann. Der Schlüssel ist persistent und exportierbar.

```
$  cat attr_262177

OBJ_ATTR_CLASS
0x03
OBJ_ATTR_KEY_TYPE
0x03
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x00
OBJ_ATTR_DECRYPT
0x00
OBJ_ATTR_WRAP
0x00
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x01
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
ecc2
OBJ_ATTR_ID

OBJ_ATTR_VALUE_LEN
0x0000008a
OBJ_ATTR_KCV
0xbbb32a
```

```
OBJ_ATTR_MODULUS
044a0f9d01d10f7437d9fa20995f0cc742552e5ba16d3d7e9a65a33e20ad3e569e68eb62477a9960a87911e6121d112
OBJ_ATTR_MODULUS_BITS
0x0000019f
```

Example Verwenden einer ungültigen EEC-Kurve

Mit diesem Befehl wird ein ECC-Schlüsselpaar mithilfe einer NID_X9_62_prime192v1-Kurve erstellt. Da diese elliptische Kurve für FIPS-Modus-HSMs nicht gültig ist, schlägt der Befehl fehl. Die Nachricht gibt an, dass ein Server im Cluster nicht verfügbar ist. Dies weist aber nicht unbedingt auf ein Problem mit den HSMs im Cluster hin.

```
Command:  genECCKeyPair -i 1 -l ecc1

       Cfm3GenerateKeyPair returned: 0xb3 : HSM Error: This operation violates the
 current configured/FIPS policies

       Cluster Error Status
       Node id 0 and err state 0x30000085 : HSM CLUSTER ERROR: Server in cluster is
 unavailable
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-i

Gibt die ID für die elliptische Kurve an. Geben Sie eine ID ein.

Zulässige Werte:

- 2: NID_X9_62_prime256v1
- 14: NID_secp384r1
- 16: NID_secp256k1

Erforderlich: Ja

-l

Gibt eine benutzerdefinierte Bezeichnung für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein. Dieselbe Bezeichnung gilt für beide Schlüssel im Paar. Die maximal zulässige Größe für `label` beträgt 127 Zeichen.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-id

Gibt einen benutzerdefinierten Bezeichner für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge. Die von Ihnen angegebene ID gilt für beide Schlüssel im Paar.

Standard : Kein ID-Wert.

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl der HSMs an, in denen der Schlüssel synchronisiert wird, bevor der Wert des Parameters `-timeout` verfällt. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Zur Beschleunigung Ihres Prozesses legen Sie den Wert von `min_srv` auf weniger als die Anzahl der HSMs im Cluster fest und stellen einen niedrigen Zeitüberschreitungswert ein. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-m_value

Gibt die Anzahl der Benutzer an, die jede kryptografische Operation genehmigen müssen, die den privaten Schlüssel des Paares verwendet. Geben Sie einen Wert von `0` bis `8` ein.

Dieser Parameter legt eine Quorum-Authentifizierungsanforderung für den privaten Schlüssel fest. Der Standardwert, `0`, deaktiviert die Quorum-Authentifizierungsfunktion für den Schlüssel. Wenn

die Quorumauthentifizierung aktiviert ist, muss die angegebene Anzahl von Benutzern ein Token signieren, um kryptografische Operationen, bei denen der private Schlüssel verwendet wird, sowie Operationen, bei denen der private Schlüssel gemeinsam genutzt oder die gemeinsame Nutzung aufgehoben wird, zu genehmigen.

Um den `m_value` eines Schlüssels zu finden, verwenden Sie getKeyInfo.

Dieser Parameter ist nur gültig, wenn der `-u`-Parameter im Befehl das Schlüsselpaar für ausreichend Benutzer freigibt, um die `m_value`-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

-nex

Macht den privaten Schlüssel nicht extrahierbar. Der generierte private Schlüssel kann nicht aus dem HSM exportiert werden. Öffentliche Schlüssel sind immer extrahierbar.

Standard: Sowohl der öffentliche als auch der private Schlüssel im Schlüsselpaar können extrahiert werden.

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie setAttribute.

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv`-Parameter angegebenen Anzahl von HSMs synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-u

Teilt den privaten Schlüssel des Paares mit den angegebenen Benutzern. Dieser Parameter erteilt anderen HSM-Crypto-Benutzern die Berechtigung zur Verwendung des privaten Schlüssels in kryptographischen Vorgängen. Öffentliche Schlüssel können von jedem Benutzer verwendet werden, ohne sie zu teilen.

Geben Sie eine durch Kommas getrennte Liste der HSM-Benutzer-IDs ein, wie etwa `-u 5,6`. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. Verwenden Sie listUsers, um im HSM nach den HSM-Benutzer-IDs von CUs zu suchen. Nutzen Sie zum Freigeben oder zum Aufheben der Freigabe vorhandener Schlüssel shareKey in cloudhsm_mgmt_util.

Standard: Nur der aktuelle Benutzer kann den privaten Schlüssel verwenden.

Erforderlich: Nein

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

Verwandte Themen

- genSymKey
- genRSAKeyPair
- genDSAKeyPair

genRSAKeyPair

Der genRSAKeyPair-Befehl im key_mgmt_util-Tool generiert ein asymmetrisches RSA-Schlüsselpaar. Sie geben den Schlüsseltyp, die Modullänge und einen öffentlichen Exponenten

an. Der Befehl generiert ein Modul der angegebenen Länge und erstellt das Schlüsselpaar. Sie können eine ID zuweisen, den Schlüssel mit anderen HSM-Benutzern teilen und nicht extrahierbare Schlüssel sowie Schlüssel, die bei Sitzungsende ablaufen, erstellen. Wenn der Befehl erfolgreich ausgeführt wurde, wird ein Schlüssel-Handle zurückgegeben, das das HSM zum Schlüssel zuweist. Sie können das Schlüssel-Handle nutzen, damit der Schlüssel für andere Befehle identifizierbar ist.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

> **Tip**
>
> Um die Attribute eines von Ihnen erstellten Schlüssels wie Typ, Länge, Bezeichnung und ID zu finden, verwenden Sie GetAttribute. Um nach den Schlüsseln für einen bestimmten Benutzer zu suchen, verwenden Sie getKeyInfo. Verwenden Sie FindKey, um Schlüssel anhand ihrer Attributwerte zu finden.

Syntax

```
genRSAKeyPair -h

genRSAKeyPair -m <modulus length>
              -e <public exponent>
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

Beispiele

Diese Beispiele verdeutlichen, wie mit genRSAKeyPair asymmetrische Schlüsselpaare in Ihren HSMs erstellt werden.

Example : Erstellen und Untersuchen eines RSA-Schlüsselpaars

Mit diesem Befehl wird ein RSA-Schlüsselpaar mit einem 2048-Bit-Modul und einem Exponenten von 65537 erstellt. Die Ausgabe zeigt, dass das Schlüssel-Handle des öffentlichen Schlüssels `2100177` ist und das Schlüssel-Handle des privaten Schlüssels `2100426`.

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa_test

Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

       Cfm3GenerateKeyPair:    public key handle: 2100177    private key handle:
 2100426

       Cluster Status:
       Node id 0 status: 0x00000000 : HSM Return: SUCCESS
       Node id 1 status: 0x00000000 : HSM Return: SUCCESS
```

Der nächste Befehl nutzt getAttribute, um die Attribute des öffentlichen Schlüssels abzurufen, den wir gerade erstellt haben. Die Ausgabe wird in die Datei `attr_2100177` geschrieben. Auf diesen folgt ein cat-Befehl, der den Inhalt der Attribut-Datei abruft. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Die resultierenden Hexadezimalwerte bestätigen, dass es sich um einen öffentliche Schlüssel (`OBJ_ATTR_CLASS 0x02`) mit einem RSA-Typ von (`OBJ_ATTR_KEY_TYPE 0x00`) handelt. Sie können mit diesem öffentlichen Schlüssel (`OBJ_ATTR_ENCRYPT 0x01`) verschlüsseln, aber nicht (`OBJ_ATTR_DECRYPT 0x00`) entschlüsseln. Die Ergebnisse können auch die Schlüssellänge (512, `0x200`), das Modul, die Modullänge (2048, `0x800`) und den öffentlichen Exponenten (65541, `0x10001`) enthalten.

```
Command:  getAttribute -o 2100177 -a 512 -out attr_2100177

Attribute size: 801, count: 26
Written to: attr_2100177 file

       Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS

$  cat attr_2100177
OBJ_ATTR_CLASS
0x02
OBJ_ATTR_KEY_TYPE
0x00
```

```
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x00
OBJ_ATTR_WRAP
0x01
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x01
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x00
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
rsa_test
OBJ_ATTR_ID

OBJ_ATTR_VALUE_LEN
0x00000200
OBJ_ATTR_KCV
0xc51c18
OBJ_ATTR_MODULUS
0xbb9301cc362c1d9724eb93da8adab0364296bde7124a241087d9436b9be57e4f7780040df03c2c
1c0fe6e3b61aa83c205280119452868f66541bbbffacbbe787b8284fc81deaeef2b8ec0ba25a077d
6983c77a1de7b17cbe8e15b203868704c6452c2810344a7f2736012424cf0703cf15a37183a1d2d0
97240829f8f90b063dd3a41171402b162578d581980976653935431da0c1260bfe756d85dca63857
d9f27a541676cb9c7def0ef6a2a89c9b9304bcac16fdf8183c0a555421f9ad5dfeb534cf26b65873
970cdf1a07484f1c128b53e10209cc6f7ac308669112968c81a5de408e7f644fe58b1a9ae1286fec
b3e4203294a96fae06f8f0db7982cb5d7f
OBJ_ATTR_MODULUS_BITS
0x00000800
OBJ_ATTR_PUBLIC_EXPONENT
0x010001
OBJ_ATTR_TRUSTED
0x00
```

```
OBJ_ATTR_WRAP_WITH_TRUSTED
0x00
OBJ_ATTR_DESTROYABLE
0x01
OBJ_ATTR_DERIVE
0x00
OBJ_ATTR_ALWAYS_SENSITIVE
0x00
OBJ_ATTR_NEVER_EXTRACTABLE
0x00
```

Example : Generieren eines freigegebenen RSA-Schlüsselpaars

Dieser Befehl generiert ein RSA-Schlüsselpaar und teilt den privaten Schlüssel mit Benutzer 4, einem anderen CU auf dem HSM. Der Befehl verwendet den Parameter `m_value`, damit mindestens zwei Genehmigungen erforderlich sind, ehe der private Schlüssel des Paares in einer kryptografischen Operation verwendet werden kann. Wenn Sie den Parameter `m_value` verwenden, müssen Sie auch `-u` im Befehl nutzen. Dabei darf der `m_value` nicht die Gesamtanzahl an Benutzern übersteigen (Anzahl der Werte in `-u` + Eigentümer).

```
Command:  genRSAKeyPair -m 2048 -e 65537 -l rsa_mofn -id rsa_mv2 -u 4 -m_value 2

       Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

       Cfm3GenerateKeyPair:    public key handle: 27    private key handle: 28

       Cluster Error Status
       Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
       Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-m

Gibt die Länge des Moduls in Bits an. Der minimale Wert beträgt 2048.

Erforderlich: Ja

-e

Gibt den öffentlichen Exponenten an. Bei diesem Wert muss es sich eine ungerade Zahl gleich oder größer als 65537 handeln.

Erforderlich: Ja

-l

Gibt eine benutzerdefinierte Bezeichnung für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein. Dieselbe Bezeichnung gilt für beide Schlüssel im Paar. Die maximal zulässige Größe für `label` beträgt 127 Zeichen.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-id

Gibt einen benutzerdefinierten Bezeichner für das Schlüsselpaar an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge. Die von Ihnen angegebene ID gilt für beide Schlüssel im Paar.

Standard : Kein ID-Wert.

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl der HSMs an, in denen der Schlüssel synchronisiert wird, bevor der Wert des Parameters `-timeout` verfällt. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Zur Beschleunigung Ihres Prozesses legen Sie den Wert von `min_srv` auf weniger als die Anzahl der HSMs im Cluster fest und stellen einen niedrigen Zeitüberschreitungswert ein. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-m_value

Gibt die Anzahl der Benutzer an, die jede kryptografische Operation genehmigen müssen, die den privaten Schlüssel des Paares verwendet. Geben Sie einen Wert von `0` bis `8` ein.

Dieser Parameter legt eine Quorum-Authentifizierungsanforderung für den privaten Schlüssel fest. Der Standardwert, `0`, deaktiviert die Quorum-Authentifizierungsfunktion für den Schlüssel. Wenn die Quorumauthentifizierung aktiviert ist, muss die angegebene Anzahl von Benutzern ein Token signieren, um kryptografische Operationen, bei denen der private Schlüssel verwendet wird, sowie Operationen, bei denen der private Schlüssel gemeinsam genutzt oder die gemeinsame Nutzung aufgehoben wird, zu genehmigen.

Um den `m_value` eines Schlüssels zu finden, verwenden Sie getKeyInfo.

Dieser Parameter ist nur gültig, wenn der `-u`-Parameter im Befehl das Schlüsselpaar für ausreichend Benutzer freigibt, um die `m_value`-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

-nex

Macht den privaten Schlüssel nicht extrahierbar. Der generierte private Schlüssel kann nicht aus dem HSM exportiert werden. Öffentliche Schlüssel sind immer extrahierbar.

Standard: Sowohl der öffentliche als auch der private Schlüssel im Schlüsselpaar können extrahiert werden.

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie setAttribute.

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv`-Parameter angegebenen Anzahl von HSMs synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-u

Teilt den privaten Schlüssel des Paares mit den angegebenen Benutzern. Dieser Parameter erteilt anderen HSM-Crypto-Benutzern die Berechtigung zur Verwendung des privaten Schlüssels in kryptographischen Vorgängen. Öffentliche Schlüssel können von jedem Benutzer verwendet werden, ohne sie zu teilen.

Geben Sie eine durch Kommas getrennte Liste der HSM-Benutzer-IDs ein, wie etwa `-u 5,6`. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. Verwenden Sie listUsers, um im HSM nach den HSM-Benutzer-IDs von CUs zu suchen. Nutzen Sie zum Freigeben oder zum Aufheben der Freigabe vorhandener Schlüssel shareKey in cloudhsm_mgmt_util.

Standard: Nur der aktuelle Benutzer kann den privaten Schlüssel verwenden.

Erforderlich: Nein

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

Verwandte Themen

- genSymKey

- genDSAKeyPair
- genECCKeyPair

genSymKey

Der genSymKey-Befehl im key_mgmt_util-Tool generiert einen symmetrischen Schlüssel in Ihren HSMs. Sie können Schlüsseltyp und Größe festlegen, eine ID und Bezeichnung zuweisen und den Schlüssel für andere HSM-Benutzer freigeben. Sie können auch nicht extrahierbare Schlüssel erstellen sowie Schlüssel, die ablaufen, wenn die Sitzung endet. Wenn der Befehl erfolgreich ausgeführt wurde, wird ein Schlüssel-Handle zurückgegeben, das das HSM zum Schlüssel zuweist. Sie können das Schlüssel-Handle nutzen, damit der Schlüssel für andere Befehle identifizierbar ist.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
genSymKey -h

genSymKey -t <key-type>
          -s <key-size>
          -l <label>
          [-id <key-ID>]
          [-min_srv <minimum-number-of-servers>]
          [-m_value <0..8>]
          [-nex]
          [-sess]
          [-timeout <number-of-seconds> ]
          [-u <user-ids>]
          [-attest]
```

Beispiele

Diese Beispiele verdeutlichen, wie mit genSymKey symmetrische Schlüssel in Ihren HSMs erstellt werden.

> **Tip**
>
> Um die Schlüssel, die Sie mit diesen Beispielen erstellen, für HMAC-Operationen zu verwenden, müssen Sie nach dem Generieren des Schlüssels den Wert `OBJ_ATTR_SIGN`

und OBJ_ATTR_VERIFY auf TRUE festlegen. Verwenden Sie setAttribute in CloudHSM Management Utility (CMU), um diese Werte festzulegen. Weitere Informationen finden Sie unter setAttribute.

Example : Generieren eines AES-Schlüssels

Mit diesem Befehl wird ein 256-Bit-AES-Schlüssel mit der Bezeichnung `aes256` erstellt. Die Ausgabe zeigt, dass das Schlüssel-Handle des neuen Schlüssels 6 ist.

```
Command: genSymKey -t 31 -s 32 -l aes256

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 6

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : Erstellen eines Sitzungsschlüssels

Mit diesem Befehl wird ein nicht extrahierbarer 192-Bit-AES-Schlüssel erstellt, der nur für die aktuelle Sitzung gültig ist. Sie können einen solchen Schlüssel erstellen, um einen Schlüssel, der exportiert wird, zu verpacken (und dann sofort zu entpacken).

```
Command: genSymKey -t 31 -s 24 -l tmpAES -id wrap01 -nex -sess
```

Example : Schnelles Zurückgeben

Mit diesem Befehl wird ein generischer 512-Byte-Schlüssel mit der Bezeichnung `IT_test_key` erstellt. Der Befehl wartet nicht, bis der Schlüssel für alle HSMs in dem Cluster synchronisiert wird. Stattdessen wird er zurückgegeben, sobald der Schlüssel in einem der HSMs erstellt ist (`-min_srv 1`) oder nach Ablauf von 1 Sekunde (`-timeout 1`), je nachdem, was kürzer ist. Falls der Schlüssel vor Ablauf des Zeitlimits nicht für die vorgegebene Mindestanzahl von HSMs synchronisiert wird, wird er nicht erstellt. Sie können einen solchen Befehl in einem Skript verwenden, das zahlreiche Schlüssel erstellt, wie die `for`-Schleife im folgenden Beispiel.

```
Command: genSymKey -t 16 -s 512 -l IT_test_key -min_srv 1 -timeout 1

$  for i in {1..30};
```

```
    do /opt/cloudhsm/bin/key_mgmt_util singlecmd loginHSM -u CU -s example_user -p
example_pwd genSymKey -l aes -t 31 -s 32 -min_srv 1 -timeout 1;
done;
```

Example : Erstellen eines allgemeinen Schlüssels mit Quorum-Autorisierung

Mit diesem Befehl wird ein allgemeiner geheimer 2048-Bit-Schlüssel mit der Bezeichnung `generic-mV2` erstellt. Der Befehl verwendet den `-u`-Parameter, um den Schlüssel mit einem anderen CU, Benutzer 6, zu teilen. Er verwendet den `-m_value`-Parameter, damit für alle kryptografischen Vorgänge, die den Schlüssel verwenden, ein Quorum aus mindestens zwei Genehmigungen erforderlich ist. Der Befehl verwendet auch den `-attest`-Parameter, um die Integrität der Firmware zu überprüfen, auf der der Schlüssel generiert wird.

Die Ausgabe zeigt, dass der Befehl einen Schlüssel mit dem Schlüssel-Handle 9 generiert hat und dass die Bescheinigungsprüfung auf der Cluster-Firmware erfolgreich bestanden wurde.

```
               Command:  genSymKey -t 16 -s 2048 -l generic-mV2 -m_value 2 -u 6 -
attest

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 9

        Attestation Check : [PASS]

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : Erstellen und Untersuchen eines Schlüssels

Mit diesem Befehl wird ein Triple-DES-Schlüssel mit der Bezeichnung `3DES_shared` und der ID `IT-02` erstellt. Der Schlüssel kann von dem aktuellen Benutzer und den Benutzern 4 und 5 verwendet werden. Der Befehl schlägt fehl, wenn die ID im Cluster nicht eindeutig ist oder wenn der aktuelle Benutzer der Benutzer 4 oder 5 ist.

Die Ausgabe zeigt, dass der neue Schlüssel das Schlüssel-Handle 7 aufweist.

```
Command: genSymKey -t 21 -s 24 -l 3DES_shared -id IT-02 -u 4,5
```

```
      Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

       Symmetric Key Created.  Key Handle: 7

       Cluster Error Status
       Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Um zu überprüfen, ob der neue 3DES-Schlüssel im Besitz des aktuellen Benutzers ist und gemeinsam mit den Benutzern 4 und 5 genutzt wird, verwenden Sie getKeyInfo. Der Befehl verwendet das Handle, das dem neuen Schlüssel zugewiesen wurde (`Key Handle: 7`).

Die Ausgabe bestätigt, dass der Schlüssel im Besitz des Benutzers 3 ist und gemeinsam mit den Benutzern 4 und 5 verwendet wird.

```
Command:  getKeyInfo -k 7

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 2 user(s):

                4, 5
```

Verwenden Sie getAttribute, um die anderen Eigenschaften des Schlüssels zu bestätigen. Der erste Befehl verwendet `getAttribute`, um alle Attribute (`-a 512`) von Schlüssel-Handle 7 (`-o 7`) abzurufen. Sie werden in die Datei `attr_7` geschrieben. Der zweite Befehl verwendet `cat`, um den Inhalt der `attr_7`-Datei abzurufen.

Mit diesem Befehl wird bestätigt, dass Schüssel 7 ein symmetrischer 192-Bit- (`OBJ_ATTR_VALUE_LEN 0x00000018` oder 24-Byte)-3DES (`OBJ_ATTR_KEY_TYPE 0x15`)-Schlüssel (`OBJ_ATTR_CLASS 0x04`) mit der Bezeichnung `3DES_shared` (`OBJ_ATTR_LABEL 3DES_shared`) und der ID `IT_02` ist (`OBJ_ATTR_ID IT-02`). Der Schlüssel ist persistent (`OBJ_ATTR_TOKEN 0x01`) und extrahierbar (`OBJ_ATTR_EXTRACTABLE 0x01`) und kann für das Verschlüsseln, Entschlüsseln und Verpacken verwendet werden.

> **Tip**
>
> Um die Attribute eines von Ihnen erstellten Schlüssels wie Typ, Länge, Bezeichnung und ID zu finden, verwenden Sie GetAttribute. Um nach den Schlüsseln für einen bestimmten

Benutzer zu suchen, verwenden Sie getKeyInfo. Verwenden Sie FindKey, um Schlüssel anhand ihrer Attributwerte zu finden.

Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

```
Command:  getAttribute -o 7 -a 512 -out attr_7

got all attributes of size 444 attr cnt 17
Attributes dumped into attr_7 file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS


$  cat attr_7

OBJ_ATTR_CLASS
0x04
OBJ_ATTR_KEY_TYPE
0x15
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x01
OBJ_ATTR_WRAP
0x00
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
```

```
3DES_shared
OBJ_ATTR_ID
IT-02
OBJ_ATTR_VALUE_LEN
0x00000018
OBJ_ATTR_KCV
0x59a46e
```

> **Tip**
>
> Um die Schlüssel, die Sie mit diesen Beispielen erstellen, für HMAC-Operationen zu verwenden, müssen Sie nach dem Generieren des Schlüssels den Wert `OBJ_ATTR_SIGN` und `OBJ_ATTR_VERIFY` auf `TRUE` festlegen. Verwenden Sie setAttribute in CMU, um diese Werte festzulegen. Weitere Informationen finden Sie unter setAttribute.

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-t

Gibt den Typ des symmetrischen Schlüssels an. Geben Sie die Konstante ein, die den Schlüsseltyp darstellt. Zum Erstellen eines AES-Schlüssels geben Sie beispielsweise `-t 31` ein.

Zulässige Werte:

- 16: GENERIC_SECRET. Ein allgemeiner geheimer Schlüssel ist ein Byte-Array, das keinem speziellen Standard entspricht, wie etwa den Anforderungen an einen AES-Schlüssel.
- 18: RC4. RC4-Schlüssel sind in FIPS-Modus-HSMs nicht gültig.
- 21: Triple DES (3DES). Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.
- 31: AES

Erforderlich: Ja

-s

Gibt die Schlüsselgröße in Byte an. Um beispielsweise einen 192-Bit-Schlüssel zu erstellen, geben Sie 24 ein.

Gültige Werte für jeden Schlüsseltyp:

- AES: 16 (128 Bit), 24 (192 Bit), 32 (256 Bit)
- 3DES: 24 (192 Bit)
- Allgemeiner geheimer Schlüssel: <3584 (28672 Bit)

Erforderlich: Ja

-l

Gibt eine benutzerdefinierte Bezeichnung für den Schlüssel an. Geben Sie eine Zeichenfolge ein.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

-id

Gibt einen benutzerdefinierten Bezeichner für den Schlüssel an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge.

Standard : Kein ID-Wert.

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl der HSMs an, in denen der Schlüssel synchronisiert wird, bevor der Wert des Parameters `-timeout` verfällt. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Zur Beschleunigung Ihres Prozesses legen Sie den Wert von `min_srv` auf weniger als die Anzahl der HSMs im Cluster fest und stellen einen niedrigen Zeitüberschreitungswert ein. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-m_value

Gibt die Anzahl der Benutzer an, die einen kryptografischen Vorgang genehmigen müssen, der den Schlüssel verwendet. Geben Sie einen Wert von `0` bis `8` ein.

Dieser Parameter legt eine Quorum-Authentifizierungsanforderung für den Schlüssel fest. Der Standardwert, `0`, deaktiviert die Quorum-Authentifizierungsfunktion für den Schlüssel. Wenn die Quorumauthentifizierung aktiviert ist, muss die angegebene Anzahl von Benutzern ein Token signieren, um kryptografische Operationen, bei denen der Schlüssel verwendet wird, sowie Operationen, bei denen der Schlüssel gemeinsam genutzt oder die gemeinsame Nutzung aufgehoben wird, zu genehmigen.

Um den `m_value` eines Schlüssels zu finden, verwenden Sie getKeyInfo.

Dieser Parameter ist nur gültig, wenn der `-u`-Parameter im Befehl den Schlüssel für ausreichend Benutzer freigibt, um die `m_value`-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

-nex

Macht den Schlüssel nicht extrahierbar. Der generierte Schlüssel kann nicht aus dem HSM exportiert werden.

Standard: Der Schlüssel ist extrahierbar.

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie setAttribute.

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv`-Parameter angegebenen Anzahl von HSMs synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-u

Gibt den Schlüssel für die angegebenen Benutzer frei. Dieser Parameter erteilt anderen HSM-Crypto-Benutzern die Berechtigung zur Verwendung dieses Schlüssels in kryptographischen Vorgängen.

Geben Sie eine durch Kommas getrennte Liste der HSM-Benutzer-IDs ein, wie etwa `-u 5,6`. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. Verwenden Sie listUsers, um im HSM nach den HSM-Benutzer-IDs von CUs zu suchen. Nutzen Sie zum Freigeben oder zum Aufheben der Freigabe vorhandener Schlüssel shareKey in cloudhsm_mgmt_util.

Standard: Nur der aktuelle Benutzer kann den Schlüssel verwenden.

Erforderlich: Nein

Verwandte Themen

- exSymKey
- genRSAKeyPair

- genDSAKeyPair
- genECCKeyPair
- setAttribute

getAttribute

Der getAttribute-Befehl in key_mgmt_util schreibt einen oder alle Attributwerte für einen AWS CloudHSM-Schlüssel in eine Datei. Wenn das von Ihnen angegebene Attribut für den Schlüsseltyp nicht existiert (z. B. der Modulus eines AES-Schlüssels), gibt getAttribute einen Fehler zurück.

Schlüsselattribute sind Eigenschaften eines Schlüssels. Dazu zählen Merkmale wie der Schlüsseltyp, Klasse, Beschriftung und ID sowie Werte, die Aktionen darstellen, die Sie mit dem Schlüssel ausführen können, wie verschlüsseln, entschlüsseln, packen, signieren und überprüfen.

Sie können getAttribute nur für Schlüssel ausführen, die Ihnen gehören oder die für Sie freigegeben wurden. Sie können diesen Befehl oder den Befehl getAttribute in cloudhsm_mgmt_util ausführen, der einen Attributwert eines Schlüssels von allen HSMs in einem Cluster abruft und ihn in stdout oder in eine Datei schreibt.

Zum Abrufen einer Liste der Attribute und Konstanten, die sie darstellen, verwenden Sie den Befehl listAttributes. Um die Attributwerte von vorhandenen Schlüsseln zu ändern, verwenden Sie setAttribute in key_mgmt_util und setAttribute in cloudhsm_mgmt_util. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
getAttribute -h

getAttribute -o <key handle>
             -a <attribute constant>
             -out <file>
```

Beispiele

Diese Beispiele zeigen, wie Sie getAttribute verwenden, um die Attribute von Schlüsseln in Ihren HSMs abzurufen.

Example : Ruft den Schlüsseltyp ab

In diesem Beispiel wird der Schlüsseltyp abgerufen (z. B. ein AES- oder 3DES-Schlüssel, ein generischer Schlüssel oder ein RSA-Schlüsselpaar oder elliptisches Kurvenschlüsselpaar).

Der erste Befehl führt listAttributes aus, das die Schlüsselattribute und deren Konstanten abruft. Die Ausgabe zeigt, dass die Konstante für den Schlüsseltyp 256 lautet. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

```
Command: listAttributes

Description
===========
The following are all of the possible attribute values for getAttributes.

      OBJ_ATTR_CLASS                = 0
      OBJ_ATTR_TOKEN                = 1
      OBJ_ATTR_PRIVATE              = 2
      OBJ_ATTR_LABEL                = 3
      OBJ_ATTR_KEY_TYPE             = 256
      OBJ_ATTR_ID                   = 258
      OBJ_ATTR_SENSITIVE            = 259
      OBJ_ATTR_ENCRYPT              = 260
      OBJ_ATTR_DECRYPT              = 261
      OBJ_ATTR_WRAP                 = 262
      OBJ_ATTR_UNWRAP               = 263
      OBJ_ATTR_SIGN                 = 264
      OBJ_ATTR_VERIFY               = 266
      OBJ_ATTR_LOCAL                = 355
      OBJ_ATTR_MODULUS              = 288
      OBJ_ATTR_MODULUS_BITS         = 289
      OBJ_ATTR_PUBLIC_EXPONENT      = 290
      OBJ_ATTR_VALUE_LEN            = 353
      OBJ_ATTR_EXTRACTABLE          = 354
      OBJ_ATTR_KCV                  = 371
```

Der zweite Befehl führt getAttribute aus. Er fordert den Schlüsseltyp (Attribut 256) für das Schlüsselhandle 524296 an und schreibt ihn in die `attribute.txt`-Datei.

```
Command: getAttribute -o 524296 -a 256 -out attribute.txt
Attributes dumped into attribute.txt file
```

Mit dem letzten Befehl wird der Inhalt der Schlüsseldatei abgerufen. Die Ausgabe zeigt, dass es sich bei dem Schlüsseltyp um `0x15` oder 21 handelt – ein Triple DES-Schlüssel (3DES). Definitionen der Klassen- und Typwerte finden Sie in der Schlüsselattributreferenz.

```
$  cat attribute.txt
OBJ_ATTR_KEY_TYPE
0x00000015
```

Example : Ruft alle Attribute eines Schlüssels ab

Dieser Befehl ruft alle Attribute des Schlüssels mit dem Schlüsselhandle 6 ab und schreibt sie in die `attr_6`-Datei. Er verwendet den Attributwert 512, der alle Attribute repräsentiert.

```
Command: getAttribute -o 6 -a 512 -out attr_6

got all attributes of size 444 attr cnt 17
Attributes dumped into attribute.txt file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS>
```

Dieser Befehl zeigt den Inhalt einer Beispiel-Attributdatei mit allen Attributwerten an. Neben den Werten zeigt er auch an, dass es sich bei dem Schlüssel um einen 256-Bit-AES-Schlüssel mit der ID `test_01` von und der Bezeichnung `aes256` handelt. Der Schlüssel ist extrahierbar und persistent, d. h., es handelt sich nicht um einen Session-only-Schlüssel. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

```
$  cat attribute.txt

OBJ_ATTR_CLASS
0x04
OBJ_ATTR_KEY_TYPE
0x15
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x01
OBJ_ATTR_WRAP
```

```
0x01
OBJ_ATTR_UNWRAP
0x01
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
aes256
OBJ_ATTR_ID
test_01
OBJ_ATTR_VALUE_LEN
0x00000020
OBJ_ATTR_KCV
0x1a4b31
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-o

Gibt das Schlüssel-Handle des Zielschlüssels an. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Verwenden Sie zum Abrufen des Schlüssel-Handles eines Schlüssels findKey.

Sie müssen außerdem Besitzer des angegebenen Schlüssels sein oder er muss für Sie freigegeben sein. Um die Benutzer eines Schlüssels zu finden, verwenden Sie getKeyInfo.

Erforderlich: Ja

-a

Bezeichnet das Attribut. Geben Sie eine Konstante, die ein Attribut darstellt, oder den Wert `512` ein, der alle Attribute repräsentiert. Zum Abrufen des Schlüsseltyps geben Sie z. B. `256` ein. Dies ist die Konstante für das Attribut `OBJ_ATTR_KEY_TYPE`.

Verwenden Sie listAttributes, um die Attribute und deren Konstanten aufzulisten. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Erforderlich: Ja

-out

Schreibt die Ausgabe in die angegebene Datei. Geben Sie einen Dateipfad ein. Sie können die Ausgabe nicht in die Datei `stdout` schreiben.

Wenn die angegebene Datei vorhanden ist, überschreibt getAttribute die Datei ohne Warnung.

Erforderlich: Ja

Verwandte Themen

- getAttribute in cloudhsm_mgmt_util
- listAttributes
- setAttribute
- findKey
- Schlüsselattributreferenz

getCaviumPrivKey

Der Befehl getCaviumPrivKey in key_mgmt_util exportiert einen privaten Schlüssel von einem HSM im gefälschten PEM-Format. Die gefälschte PEM-Datei enthält nicht das Material des tatsächlichen privaten Schlüssels, dafür jedoch Verweise auf den privaten Schlüssel im HSM. Die gefälschte PEM-Datei ermöglicht eine SSL-/TLS-Auslagerung von Ihrem Webserver nach AWS CloudHSM. Weitere Informationen erhalten Sie unter SSL-/TLS-Auslagerung unter Linux.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
getCaviumPrivKey -h

getCaviumPrivKey -k <private-key-handle
                 -out <fake-PEM-file>
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit getCaviumPrivKey einen privaten Schlüssel im gefälschten PEM-Format exportieren.

Example : Exportieren einer gefälschten PEM-Datei

Dieser Befehl erstellt und exportiert eine gefälschte PEM-Version eines privaten Schlüssels mit dem Handle 15 und speichert sie in einer Datei namens `cavKey.pem`. Wird der Befehl erfolgreich ausgeführt, gibt exportPrivateKey eine Erfolgsmeldung zurück.

```
Command: getCaviumPrivKey -k 15 -out cavKey.pem

Private Key Handle is written to cavKey.pem in fake PEM format

        getCaviumPrivKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-h`

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

`-k`

Gibt das Schlüssel-Handle des privaten Schlüssels an, der im gefälschten PEM-Format exportiert werden soll.

Erforderlich: Ja

`-out`

Gibt den Namen der Datei an, in die der gefälschte PEM-Schlüssel geschrieben werden soll.

Erforderlich: Ja

Verwandte Themen

- importPrivateKey
- SSL/TLS-Auslagerung unter Linux

getCert

Der getCert-Befehl in key_mgmt_util ruft die Partitionszertifikate eines HSM ab und speichert sie in einer Datei. Wenn Sie den Befehl ausführen, geben Sie den Typ des abzurufenden Zertifikats an. Verwenden Sie dazu eine der entsprechenden ganzzahligen Werte, wie im folgenden Abschnitt Parameter beschrieben. Weitere Informationen über die Rolle der einzelnen Zertifikate finden Sie unter Überprüfen der HSM-Identität.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
getCert -h

getCert -f <file-name>
        -t <certificate-type>
```

Beispiel

Dieses Beispiel zeigt, wie man getCert verwendet, um das Kundenstammzertifikat eines Clusters abzurufen und als Datei zu speichern.

Example : Abrufen eines Kundenstammzertifikats

Dieser Befehl exportiert ein Kundenstammzertifikat (dargestellt durch die Ganzzahl 4) und speichert es in einer Datei namens `userRoot.crt`. Wird der Befehl erfolgreich ausgeführt, gibt getCert eine Erfolgsmeldung zurück.

```
Command: getCert -f userRoot.crt -s 4
```

```
Cfm3GetCert() returned 0 :HSM Return: SUCCESS
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-f

Gibt den Namen der Datei an, in der das abgerufene Zertifikat gespeichert wird.

Erforderlich: Ja

-s

Eine Ganzzahl, die den Typ des abzurufenden Partitionszertifikats angibt. Die Ganzzahlen und ihre entsprechenden Zertifikattypen sind:

- 1 – Hersteller-Stammzertifikat
- 2 – Hersteller-Hardwarezertifikat
- 4 – Kunden-Stammzertifikat
- 8 – Clusterzertifikat (signiert vom Kunden-Stammzertifikat)
- 16 – Clusterzertifikat (verkettet mit dem Hersteller-Stammzertifikat)

Erforderlich: Ja

Verwandte Themen

- HSM-Identität überprüfen
- getCert (in cloudhsm_mgmt_util)

getKeyInfo

Der getKeyInfo-Befehl in key_mgmt_util gibt die HSM-Benutzer-IDs der Benutzer zurück, die den Schlüssel verwenden können, einschließlich Eigentümer und Crypto-Benutzer (CUs), mit denen

der Schlüssel geteilt wird. Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, gibt getKeyInfo auch die Anzahl der Benutzer zurück, die kryptografische Operationen genehmigen müssen, von denen der Schlüssel verwendet wird. Sie können getKeyInfo nur auf Schlüsseln ausführen, die Ihnen gehören oder die für Sie freigegeben wurden.

Wenn Sie getKeyInfo auf öffentliche Schlüsseln ausführen, gibt getKeyInfo nur den Schlüsseleigentümer zurück, auch wenn alle HSM-Benutzer den öffentlichen Schlüssel verwenden können. Verwenden Sie zur Suche nach den HSM-Benutzer-IDs der Benutzer in Ihren HSMs listUsers. Um nach den Schlüsseln für einen bestimmten Benutzer zu suchen, verwenden Sie findKey `-u`.

Ihren gehören die Schlüssel, die Sie erstellen. Sie können einen Schlüssel für andere Benutzer freigeben, wenn Sie ihn erstellen. Nutzen Sie dann zum Freigeben oder zum Aufheben der Freigabe shareKey in cloudhsm_mgmt_util.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
getKeyInfo -h

getKeyInfo -k <key-handle>
```

Beispiele

Diese Beispiele veranschaulichen die Verwendung von getKeyInfo zum Abrufen von Informationen über die Benutzer eines Schlüssels.

Example : Abfragen der Benutzer eines symmetrischen Schlüssels

Über diesen Befehl erhalten Sie die Benutzer, die den (symmetrischen) AES-Schlüssel mit dem Schlüssel-Handle 9 verwenden können. Die Ausgabe zeigt, dass Benutzer 3 den Schlüssel besitzt und mit Benutzer 4 geteilt hat.

```
Command:  getKeyInfo -k 9

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3
```

```
       also, shared to following 1 user(s):

                4
```

Example : Abfragen der Benutzer eines asymmetrischen Schlüsselpaares

Diese Befehle verwenden getKeyInfo, um die Benutzer abzurufen, die die Schlüssel in einem (asymmetrischen) RSA-Schlüsselpaar verwenden können. Das Schlüssel-Handle des öffentlichen Schlüssels ist 21. Das Schlüssel-Handle des privaten Schlüssels ist 20.

Wenn Sie getKeyInfo auf dem privaten Schlüssel (20) ausführen, werden der Schlüsselbesitzer (3) und die Crypto-Benutzer (CUs, Crypto Users) 4 und 5 zurückgegeben, mit denen der Schlüssel geteilt wird.

```
Command:  getKeyInfo -k 20

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 2 user(s):

                4
                5
```

Wenn Sie getKeyInfo auf dem öffentlichen Schlüssel (21) ausführen, wird nur der Schlüsseleigentümer (3) zurückgegeben.

```
Command:  getKeyInfo -k 21

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3
```

Um zu bestätigen, dass Benutzer 4 den öffentlichen Schlüssel (und alle öffentlichen Schlüssel im HSM) verwenden kann, verwenden Sie den -u-Parameter von findKey.

Die Ausgabe zeigt, dass Benutzer 4 sowohl den öffentlichen (21) als auch den privaten Schlüssel (20) im Schlüsselpaar verwenden kann. Benutzer 4 kann zudem alle anderen öffentlichen und privaten Schlüssel nutzen, die sie erstellt haben oder die für sie freigegeben wurden.

```
Command:  findKey -u 4
Total number of keys present 8

 number of keys matched from start index 0::7
11, 12, 262159, 262161, 262162, 19, 20, 21

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example : Abrufen des Quorum-Authentifizierungswerts (m_value) für einen Schlüssel

Mit diesem Beispiel wird gezeigt, wie der `m_value` für einen Schlüssel abgerufen werden kann, d. h. die Anzahl der Benutzer im Quorum, die alle kryptografischen Vorgänge genehmigen müssen, bei denen der Schlüssel verwendet wird.

Wenn die Quorum-Authentifizierung für einen Schlüssel aktiviert ist, muss das Quorum der Benutzer alle kryptografischen Operationen genehmigen, bei denen der Schlüssel verwendet wird. Zum Aktivieren der Quorum-Authentifizierung und Festlegen der Quorum-Größe nutzen Sie beim Erstellen des Schlüssels den Parameter `-m_value`.

Dieser Befehl verwendet genRSAKeyPair zum Erstellen eines RSA-Schlüsselpaares, das mit Benutzer 4 geteilt wird. Er verwendet den Parameter `m_value` zur Aktivierung der Quorum-Authentifizierung für den privaten Schlüssel im Paar und zur Festlegung der Quorum-Größe auf zwei Benutzer. Die Anzahl der Benutzer muss groß genug sein, um die erforderlichen Genehmigungen bereitstellen zu können.

Die Ausgabe zeigt, dass der Befehl den öffentlichen Schlüssel 27 und den privaten Schlüssel 28 erstellt hat.

```
Command:  genRSAKeyPair -m 2048 -e 195193 -l rsa_mofn -id rsa_mv2 -u 4 -m_value 2

        Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 27    private key handle: 28

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Dieser Befehl nutzt getKeyInfo zum Abrufen von Informationen über die Benutzer des privaten Schlüssels. Die Ausgabe zeigt, dass der Schlüssel im Besitz des Benutzers 3 ist und gemeinsam mit Benutzer 4 verwendet wird. Sie zeigt auch, dass ein Quorum von zwei Benutzern jede kryptografische Operation genehmigen muss, bei der der Schlüssel verwendet wird.

```
Command:  getKeyInfo -k 28

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

        Owned by user 3

        also, shared to following 1 user(s):

                4
         2 Users need to approve to use/manage this key
```

Parameter

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-k

Gibt das Schlüssel-Handle eines Schlüssels im HSM an. Geben Sie das Schlüssel-Handle eines Schlüssels ein, den Sie besitzen oder teilen. Dieser Parameter muss angegeben werden.

Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Erforderlich: Ja

Verwandte Themen

- getKeyInfo in cloudhsm_mgmt_util
- listUsers
- findKey
- findAllKeys in cloudhsm_mgmt_util

help

Der help-Befehl in key_mgmt_util zeigt Informationen über alle verfügbaren key_mgmt_util-Befehle an.

Vor dem Ausführen von help müssen Sie key_mgmt_util starten.

Syntax

```
help
```

Beispiel

Dieses Beispiel zeigt die Ausgabe des `help`-Befehls.

Example

```
Command:  help

Help Commands Available:

Syntax: <command> -h


   Command               Description
   =======               ===========

   exit                   Exits this application
   help                   Displays this information

        Configuration and Admin Commands
   getHSMInfo             Gets the HSM Information
   getPartitionInfo       Gets the Partition Information
   listUsers              Lists all users of a partition
   loginStatus            Gets the Login Information
   loginHSM               Login to the HSM
   logoutHSM              Logout from the HSM

        M of N commands
   getToken               Initiate an MxN service and get Token
   delToken               delete Token(s)
   approveToken           Approves an MxN service
   listTokens             List all Tokens in the current partition
```

```
     Key Generation Commands

     Asymmetric Keys:
genRSAKeyPair          Generates an RSA Key Pair
genDSAKeyPair          Generates a DSA Key Pair
genECCKeyPair          Generates an ECC Key Pair

     Symmetric Keys:
genPBEKey              Generates a PBE DES3 key
genSymKey              Generates a Symmetric keys

     Key Import/Export Commands
createPublicKey        Creates an RSA public key
importPubKey           Imports RSA/DSA/EC Public key
exportPubKey           Exports RSA/DSA/EC Public key
importPrivateKey       Imports RSA/DSA/EC private key
exportPrivateKey       Exports RSA/DSA/EC private key
imSymKey               Imports a Symmetric key
exSymKey               Exports a Symmetric key
wrapKey                Wraps a key from from HSM using the specified handle
unWrapKey              UnWraps a key into HSM using the specified handle

     Key Management Commands
deleteKey              Delete Key
setAttribute           Sets an attribute of an object
getKeyInfo             Get Key Info about shared users/sessions
findKey                Find Key
findSingleKey          Find single Key
getAttribute           Reads an attribute from an object

     Certificate Setup Commands
getCert                Gets Partition Certificates stored on HSM

     Key Transfer Commands
insertMaskedObject     Inserts a masked object
extractMaskedObject    Extracts a masked object

     Management Crypto Commands
sign                   Generates a signature
verify                 Verifies a signature
aesWrapUnwrap          Does NIST AES Wrap/Unwrap

     Helper Commands
```

```
   Error2String           Converts Error codes to Strings
                          save key handle in fake PEM format
   getCaviumPrivKey       Saves an RSA private key handle
                          in fake PEM format
   IsValidKeyHandlefile   Checks if private key file has
                          an HSM key handle or a real key
   listAttributes         List all attributes for getAttributes
   listECCCurveIds        List HSM supported ECC CurveIds
```

Parameter

Für diesen Befehl gibt es keine Parameter.

Verwandte Themen

- loginHSM und logoutHSM

importPrivateKey

Der importPrivateKey-Befehl in key_mgmt_util importiert einen asymmetrischen privaten Schlüssel aus einer Datei in ein HSM. Das HSM erlaubt keinen direkten Import von Schlüsseln im Klartext. Der Befehl verschlüsselt den privaten Schlüssel mit einem von Ihnen angegebenen AES-Wrapping-Schlüssel und entpackt den Schlüssel innerhalb des HSM. Wenn Sie versuchen, einem Zertifikat einen AWS CloudHSM-Schlüssel zuzuordnen, finden Sie weitere Informationen in diesem Thema.

> **Note**
>
> Sie können einen passwortgeschützten PEM-Schlüssel nicht mithilfe eines symmetrischen oder privaten Schlüssels importieren.

Sie müssen einen AES-Wrapping-Schlüssel angeben, der `OBJ_ATTR_UNWRAP`- und `OBJ_ATTR_ENCRYPT`-Attributwerte 1 hat. Mit dem Befehl getAttribute können Sie nach den Attributen eines Schlüssels suchen.

> **Note**
>
> Dieser Befehl bietet keine Option, um den importierten Schlüssel als nicht exportierbar zu markieren.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
importPrivateKey -h

importPrivateKey -l <label>
                 -f <key-file>
                 -w <wrapping-key-handle>
                 [-sess]
                 [-id <key-id>]
                 [-m_value <0...8>]
                 [min_srv <minimum-number-of-servers>]
                 [-timeout <number-of-seconds>]
                 [-u <user-ids>]
                 [-wk <wrapping-key-file>]
                 [-attest]
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit importPrivateKey einen privaten Schlüssel in ein HSM importieren.

Example : Importieren eines privaten Schlüssels

Dieser Befehl importiert den privaten Schlüssel aus einer Datei mit dem Namen `rsa2048.key`, der Bezeichnung `rsa2048-imported` und einem Verpackungsschlüssel mit dem Handle `524299`. Wenn der Befehl erfolgreich ausgeführt wurde, gibt importPrivateKey ein Schlüssel-Handle für den importierten Schlüssel sowie eine Erfolgsmeldung zurück.

```
Command: importPrivateKey -f rsa2048.key -l rsa2048-imported -w 524299

BER encoded key length is 1216

Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Private Key Unwrapped.  Key Handle: 524301
```

```
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-h`

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

`-l`

Gibt die benutzerdefinierte Bezeichnung des privaten Schlüssels an.

Erforderlich: Ja

`-f`

Gibt den Dateinamen des zu importierenden Schlüssels an.

Erforderlich: Ja

`-w`

Gibt das Schlüssel-Handle des Verpackungsschlüssels an. Dieser Parameter muss angegeben werden. Nutzen Sie den Befehl findKey, um Schlüssel-Handles zu suchen.

Ermitteln Sie mithilfe von getAttribute den Wert des `OBJ_ATTR_WRAP`-Attributs (262), um zu bestimmen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann. Einen Verpackungsschlüssel erstellen Sie mit dem Befehl genSymKey, der zum Erstellen eines AES-Schlüssels (Typ 31) verwendet wird.

Wenn Sie den Parameter `-wk` zum Angeben eines externen Entpackungsschlüssels verwenden, wird der `-w`-Verpackungsschlüssel während des Imports zum Verpacken, nicht aber zum Entpacken des Schlüssels verwendet.

Erforderlich: Ja

`-sess`

Gibt den importierten Schlüssel als Sitzungsschlüssel an.

Standard: Der importierte Schlüssel wird im Cluster als persistenter Schlüssel (Token) bereitgehalten.

Erforderlich: Nein

`-id`

Gibt die ID des zu importierenden Schlüssels an.

Standard : Kein ID-Wert.

Erforderlich: Nein

`-m_value`

Gibt die Anzahl der Benutzer an, die einen kryptografischen Vorgang genehmigen müssen, der den importierten Schlüssel verwendet. Geben Sie einen Wert zwischen **`0`** und **`8`** ein.

Dieser Parameter ist nur gültig, wenn der `-u`-Parameter im Befehl den Schlüssel für ausreichend Benutzer freigibt, um die `m_value`-Anforderung zu erfüllen.

Standard: 0

Erforderlich: Nein

`-min_srv`

Gibt die Mindestanzahl der HSMs an, in denen der importierte Schlüssel synchronisiert wird, bevor der Wert des Parameters `-timeout` verfällt. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Zur Beschleunigung Ihres Prozesses legen Sie den Wert von `min_srv` auf weniger als die Anzahl der HSMs im Cluster fest und stellen einen niedrigen Zeitüberschreitungswert ein. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

`-timout`

Gibt an, wie viele Sekunden bei Einschluss des Parameters `min-serv` gewartet wird, bis der Schlüssel in verschiedenen HSMs synchronisiert wird. Wenn keine Anzahl angegeben ist, wird die Abfrage ohne zeitliche Einschränkung fortgesetzt.

Standard: keine Einschränkung

Erforderlich: Nein

`-u`

Gibt die Liste der Benutzer an, für die der importierte private Schlüssel freigegeben werden soll. Dieser Parameter erteilt anderen HSM-Crypto-Benutzern (CU) die Berechtigung, den importierten Schlüssel für kryptografische Vorgänge zu verwenden.

Geben Sie eine durch Kommas getrennte Liste der HSM-Benutzer-IDs ein, wie etwa `-u 5,6`. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. Verwenden Sie listUsers, um im HSM nach den HSM-Benutzer-IDs von CUs zu suchen.

Standard: Nur der aktuelle Benutzer kann den importierten Schlüssel verwenden.

Erforderlich: Nein

`-wk`

Gibt den Schlüssel an, mit dem der Schlüssel, der importiert wird, verpackt werden soll. Geben Sie den Pfad und den Namen einer Datei an, die einen Klartext-AES-Schlüssel enthält.

Wenn Sie diesen Parameter einschließen, verwendet importPrivateKey den Schlüssel in der `-wk`-Datei, um den zu importierenden Schlüssel zu verpacken. Außerdem wird der vom `-w`-Parameter angegebene Schlüssel zum Entpacken verwendet.

Standard: Verwenden Sie den im `-w`-Parameter angegeben Verpackungsschlüssel für das Verpacken und Entpacken von Schlüsseln.

Erforderlich: Nein

`-attest`

Überprüft die Firmware-Antwort, um sicherzustellen, dass die Firmware, auf der der Cluster ausgeführt wird, nicht beeinträchtigt wurde.

Erforderlich: Nein

Verwandte Themen

- wrapKey
- unWrapKey
- genSymKey
- exportPrivateKey

importPubKey

Der importPubKey-Befehl in key_mgmt_util importiert einen öffentlichen Schlüssel im PEM-Format in ein HSM. Sie können ihn verwenden, um öffentliche und außerhalb des HSM erstellte Schlüssel zu importieren. Sie können den Befehl auch verwenden, um Schlüssel zu importieren, die aus einem HSM exportiert wurden (beispielsweise die, die vom Befehl exportPubKey exportiert wurden).

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
importPubKey -h

importPubKey -l <label>
             -f <key-file>
             [-sess]
             [-id <key-id>]
             [min_srv <minimum-number-of-servers>]
             [-timeout <number-of-seconds>]
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit importPubKey einen öffentlichen Schlüssel in ein HSM importieren.

Example : Importieren eines öffentlichen Schlüssels

Mit diesem Befehl wird ein öffentlicher Schlüssel aus einer Datei mit dem Namen `public.pem` unter der Bezeichnung `importedPublicKey` importiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt importPubKey ein Schlüssel-Handle für den importierten Schlüssel sowie eine Erfolgsmeldung zurück.

```
Command: importPubKey -l importedPublicKey -f public.pem

Cfm3CreatePublicKey returned: 0x00 : HSM Return: SUCCESS

Public Key Handle: 262230

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

-h

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

-l

Gibt die benutzerdefinierte Bezeichnung für den öffentlichen Schlüssel an.

Erforderlich: Ja

-f

Gibt den Dateinamen des zu importierenden Schlüssels an.

Erforderlich: Ja

-sess

Bezeichnet den importierten Schlüssel als Sitzungsschlüssel.

Standard: Der importierte Schlüssel wird im Cluster als persistenter Schlüssel (Token) bereitgehalten.

Erforderlich: Nein

-id

Gibt die ID des zu importierenden Schlüssels an.

Standard : Kein ID-Wert.

Erforderlich: Nein

`-min_srv`

Gibt die Mindestanzahl der HSMs an, mit denen der importierte Schlüssel synchronisiert wird, bevor der Wert des Parameters `-timeout` verfällt. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Zur Beschleunigung Ihres Prozesses legen Sie den Wert von `min_srv` auf weniger als die Anzahl der HSMs im Cluster fest und stellen einen niedrigen Zeitüberschreitungswert ein. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

`-timeout`

Gibt an, wie viele Sekunden bei Einschluss des Parameters `min-serv` gewartet wird, bis der Schlüssel in verschiedenen HSMs synchronisiert wird. Wenn keine Anzahl angegeben ist, wird die Abfrage ohne zeitliche Einschränkung fortgesetzt.

Standard: keine Einschränkung

Erforderlich: Nein

Verwandte Themen

- exportPubKey
- Generieren von Schlüsseln

imSymKey

Der imSymKey-Befehl im Tool key_mgmt_util importiert eine Klartextkopie eines symmetrischen Schlüssels aus einer Datei in das HSM. Sie können diese zum Importieren von Schlüsseln verwenden, die Sie mit einer beliebigen Methode außerhalb des HSM erstellt haben, sowie von Schlüsseln, die aus einem HSM exportiert wurden, z. B. die Schlüssel, die mit dem Befehl exSymKey in eine Datei geschrieben werden.

Während des Importvorgangs verwendet imSymKey einen AES-Schlüssel, den Sie auswählen (Schlüssel zum Packen), um den zu importierenden Schlüssel zu packen (verschlüsseln) und zu entpacken (entschlüsseln). Allerdings funktioniert imSymKey nur für Dateien, die Klartextschlüssel enthalten. Zum Exportieren und Importieren von verschlüsselten Schlüsseln, verwenden Sie die Befehle wrapKey und unWrapKey.

Darüber hinaus exportiert der Befehl imSymKey nur symmetrische Schlüssel. Um öffentliche Schlüssel zu importieren, verwenden Sie importPubKey. Um private Schlüssel zu importieren, verwenden Sie importPrivateKey oder wrapKey.

> **Note**
>
> Sie können einen passwortgeschützten PEM-Schlüssel nicht mithilfe eines symmetrischen oder privaten Schlüssels importieren.

Importierte Schlüssel können genauso wie im HSM generierte Schlüssel verwendet werden. Der Wert des Attributs OBJ_ATTR_LOCAL ist allerdings Null, was darauf hinweist, dass es nicht lokal generiert wurde. Mit dem folgenden Befehl können Sie einen symmetrischen Schlüssel freigeben, während Sie ihn importieren. Mit dem Befehl `shareKey` in cloudhsm_mgmt_util können Sie den Schlüssel freigeben, nachdem er importiert wurde.

```
imSymKey -l aesShared -t 31 -f kms.key -w 3296 -u 5
```

Vergewissern Sie sich, dass Sie die Schlüsseldatei nach dem Importieren des Schlüssels markieren oder entfernen. Dieser Befehl verhindert nicht das wiederholte Importieren der gleichen Schlüsselinformationen. Das Ergebnis, mehrere Schlüssel mit eindeutigen Schlüssel-Handles und den gleichen Schlüsselinformationen, erschwert es, die Verwendung der Schlüsselinformationen nachzuverfolgen und eine Überschreitung der kryptografischen Grenzwerte zu verhindern.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
imSymKey -h

imSymKey -f <key-file>
         -w <wrapping-key-handle>
```

```
         -t <key-type>
         -l <label>
         [-id <key-ID>]
         [-sess]
         [-wk <wrapping-key-file> ]
         [-attest]
         [-min_srv <minimum-number-of-servers>]
         [-timeout <number-of-seconds> ]
         [-u <user-ids>]
```

Beispiele

Diese Beispiele verdeutlichen, wie mit imSymKey symmetrische Schlüssel in Ihre HSMs importiert werden.

Example : Importieren eines symmetrischen AES-Schlüssels

Dieses Beispiel verwendet imSymKey zum Importieren eines symmetrischen AES-Schlüssels in die HSMs.

Der erste Befehl verwendet OpenSSL, um einen symmetrischen 256-Bit-AES-Schlüssel nach dem Zufallsprinzip zu generieren. Der Schlüssel wird in der Datei `aes256.key` gespeichert.

```
$  openssl rand -out aes256-forImport.key 32
```

Der zweite Befehl verwendet imSymKey zum Importieren des AES-Schlüssels aus der Datei `aes256.key` in die HSMs. Dabei wird Schlüssel 20, ein AES-Schlüssel im HSM, als Schlüssel zum Packen eingesetzt und die Beschriftung `imported` festgelegt. Im Gegensatz zur ID muss die Beschriftung im Cluster nicht eindeutig sein. Der Wert des Parameters (Typs) `-t` ist `31`, der AES darstellt.

Die Ausgabe zeigt, dass der Schlüssel in der Datei verpackt und entpackt und dann in das HSM importiert wurde, wo ihm das Schlüssel-Handle 262180 zugewiesen wurde.

```
Command:  imSymKey -f aes256.key -w 20 -t 31 -l imported

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
        Symmetric Key Unwrapped.  Key Handle: 262180

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Der nächste Befehl verwendet getAttribute, um das Attribut OBJ_ATTR_LOCAL (Attribut 355) des neu importierten Schlüssels abzurufen und in der Datei `attr_262180` zu speichern.

```
Command:  getAttribute -o 262180 -a 355 -out attributes/attr_262180
Attributes dumped into attributes/attr_262180_imported file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

Wenn Sie die Attributdatei untersuchen, sehen Sie, dass der Wert des Attributs `OBJ_ATTR_LOCAL` null ist. Dies bedeutet, dass die Schlüsselinformationen nicht im HSM generiert wurden.

```
$  cat attributes/attr_262180_local
OBJ_ATTR_LOCAL
0x00000000
```

Example : Verschieben eines symmetrischen Schlüssels zwischen Clustern

Dieses Beispiel zeigt, wie man exSymKey und imSymKey verwendet, um einen Klartext-AES-Schlüssel zwischen Clustern zu verschieben. Sie können einen Prozess wie diesen zum Erstellen einer AES-Verpackung verwenden, die in den HSMs beider Cluster vorhanden ist. Sobald Sie den freigegebenen Schlüssel zum Packen erstellt haben, können Sie mit wrapKey und unWrapKey verschlüsselte Schlüssel zwischen den Clustern verschieben.

Der CU-Benutzer, der diese Operation ausführt, muss über die Berechtigung zum Anmelden bei den HSMs in beiden Clustern verfügen.

Der erste Befehl verwendet exSymKey, um den Schlüssel 14, einen 32-Bit-AES-Schlüssel, aus dem Cluster 1 in die Datei `aes.key` zu exportieren. Er nutzt Schlüssel 6, einen AES-Schlüssel in den HSMs in Cluster 1, als Schlüssel zum Packen.

```
Command: exSymKey -k 14 -w 6 -out aes.key

        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "aes.key"
```

Der Benutzer meldet sich dann bei key_mgmt_util in Cluster 2 an und führt einen imSymKey-Befehl zum Importieren des Schlüssels in der Datei `aes.key` in die HSMs von Cluster 2 aus. Dieser Befehl verwendet Schlüssel 252152, einen AES-Schlüssel in den HSMs von Cluster 2, als Schlüssel zum Packen.

Da die Wrapping-Schlüssel, die exSymKey und imSymKey verwenden, die Zielschlüssel ver- und entpacken, müssen die Wrapping-Schlüssel auf den verschiedenen Clustern nicht identisch sein.

Die Ausgabe zeigt, dass der Schlüssel erfolgreich in Cluster 2 importiert und das Schlüssel-Handle 21 zugwiesen wurde.

```
Command:  imSymKey -f aes.key -w 262152 -t 31 -l xcluster

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 21

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Um nachzuweisen, dass Schlüssel 14 in Cluster 1 und Schlüssel 21 in Cluster 2 über dieselben Schlüsselinformationen verfügen, rufen Sie den Schlüsselprüfwert (Key Check Value, KCV) der einzelnen Schlüssel ab. Bei identischen KCV-Werten sind die Schlüsselinformationen gleich.

Im folgenden Befehl wird getAttribute in Cluster 1 verwendet, um den Wert des KCV-Attributs (Attribut 371) von Schlüssel 14 in die Datei `attr_14_kcv` zu schreiben. Anschließend wird mit dem Befehl cat der Inhalt der Datei `attr_14_kcv` abgerufen.

```
Command:  getAttribute -o 14 -a 371 -out attr_14_kcv
```

```
Attributes dumped into attr_14_kcv file

$  cat attr_14_kcv
OBJ_ATTR_KCV
0xc33cbd
```

Dieser ganz ähnliche Befehl verwendet getAttribute in Cluster 2, um den Wert des KCV-Attributs (Attribut 371) von Schlüssel 21 in die Datei `attr_21_kcv` zu schreiben. Anschließend wird mit dem Befehl cat der Inhalt der Datei `attr_21_kcv` abgerufen.

```
Command:  getAttribute -o 21 -a 371 -out attr_21_kcv
Attributes dumped into attr_21_kcv file

$  cat attr_21_kcv
OBJ_ATTR_KCV
0xc33cbd
```

Die Ausgabe zeigt, dass die KCV-Werte der beiden Schlüssel gleich sind, was beweist, dass die Schlüsselinformationen identisch sind.

Da die gleichen Schlüsselinformationen in den HSMs beider Cluster vorhanden sind, können Sie jetzt verschlüsselte Schlüssel für die Cluster freigeben, ohne den Klartextschlüssel verfügbar zu machen. Beispielsweise können Sie den Befehl `wrapKey` mit dem Schlüssel 14 zum Packen verwenden, um einen verschlüsselten Schlüssel aus Cluster 1 zu exportieren, und anschließend `unWrapKey` mit Schlüssel 21 zum Packen verwenden, um den verschlüsselten Schlüssel in Cluster 2 zu importieren.

Example : Importieren eines Sitzungsschlüssels

Dieser Befehl verwendet die Parameter `-sess` von imSymKey zum Importieren eines 192-Bit-3DES-Schlüssels, der nur für die aktuelle Sitzung gültig ist.

Der Befehl verwendet den Parameter `-f`, um die Datei mit dem zu importierenden Schlüssel anzugeben, den Parameter `-t` zur Angabe des Schlüsseltyps und den Parameter `-w`, um den Schlüssel zum Packen festzulegen. Er verwendet den Parameter `-l` zum Angeben einer Beschriftung, die den Schlüssel kategorisiert, und den Parameter `-id` zum Erstellen eines benutzerfreundlichen, aber eindeutigen Bezeichners für den Schlüssel. Mit dem Parameter `-attest` wird außerdem die Firmware, die den Schlüssel importiert, überprüft.

Die Ausgabe zeigt, dass der Schlüssel erfolgreich verpackt und entpackt und dann in das HSM importiert wurde, wo ihm das Schlüssel-Handle 37 zugewiesen wurde. Außerdem wurde die Bescheinigungsprüfung bestanden, was angibt, dass die Firmware nicht unbefugt geändert wurde.

```
Command:  imSymKey -f 3des192.key -w 6 -t 21 -l temp -id test01 -sess -attest

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 37

        Attestation Check : [PASS]

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Anschließend können Sie den Befehl getAttribute oder findKey verwenden, um die Attribute der neu importierten Schlüssel zu überprüfen. Der folgende Befehl verwendet findKey, um zu überprüfen, dass der Schlüssel 37 über die vom Befehl angegebenen Typ-, Beschriftungs- und ID-Werte verfügt, und dass es sich um einen Sitzungsschlüssel handelt. Wie in Zeile 5 der Ausgabe gezeigt, meldet findKey, dass der einzige Schlüssel, der allen Attributen entspricht, Schlüssel 37 ist.

```
Command:  findKey -t 21 -l temp -id test01 -sess 1
Total number of keys present 1

 number of keys matched from start index 0::0
37

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

-f

Gibt die Datei an, die den zu importierenden Schlüssel enthält.

Die Datei muss eine Klartextkopie eines AES- oder Triple-DES-Schlüssels der angegebenen Länge enthalten. RC4- und DES-Schlüssel sind in HSMs im FIPS-Modus-nicht gültig.

- AES: 16, 24 oder 32 Bytes
- Triple-DES (3DES): 24 Byte

Erforderlich: Ja

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-id

Gibt einen benutzerdefinierten Bezeichner für den Schlüssel an. Geben Sie eine Zeichenfolge ein, die im Cluster eindeutig ist. Der Standardwert ist eine leere Zeichenfolge.

Standard : Kein ID-Wert.

Erforderlich: Nein

-l

Gibt eine benutzerdefinierte Bezeichnung für den Schlüssel an. Geben Sie eine Zeichenfolge ein.

Sie können eine beliebige Phrase verwenden, die Ihnen bei der Identifizierung des Schlüssels hilft. Da die Bezeichnung nicht eindeutig sein muss, können Sie sie verwenden, um Schlüssel zu gruppieren und zu kategorisieren.

Erforderlich: Ja

-min_srv

Gibt die Mindestanzahl der HSMs an, in denen der Schlüssel synchronisiert wird, bevor der Wert des Parameters `-timeout` verfällt. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Zur Beschleunigung Ihres Prozesses legen Sie den Wert von `min_srv` auf weniger als die Anzahl

der HSMs im Cluster fest und stellen einen niedrigen Zeitüberschreitungswert ein. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie setAttribute.

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv`-Parameter angegebenen Anzahl von HSMs synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-t

Gibt den Typ des symmetrischen Schlüssels an. Geben Sie die Konstante ein, die den Schlüsseltyp darstellt. Zum Erstellen eines AES-Schlüssels geben Sie beispielsweise `-t 31` ein.

Zulässige Werte:

- 21: Triple DES (3DES).
- 31: AES

Erforderlich: Ja

-u

Gibt den zu importierenden Schlüssel für die angegebenen Benutzer frei. Dieser Parameter erteilt anderen HSM-Crypto-Benutzern die Berechtigung zur Verwendung dieses Schlüssels in kryptographischen Vorgängen.

Geben Sie eine ID oder eine Liste von HSM-Benutzer-IDs mit Kommas als Trennzeichen ein, z. B. `-u 5,6`. Fügen Sie die HSM-Benutzer-ID des aktuellen Benutzers nicht ein. Zum Ermitteln einer ID können Sie den Befehl listUsers im Befehlszeilen-Tool cloudhsm_mgmt_util oder den Befehl listUsers im Befehlszeilen-Tool key_mgmt_util verwenden.

Erforderlich: Nein

-w

Gibt das Schlüssel-Handle des Verpackungsschlüssels an. Dieser Parameter muss angegeben werden. Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Ein Schlüssel zum Packen ist ein Schlüssel im HSM, der zum Verschlüsseln („Verpacken“) und zum Entschlüsseln („Entpacken“) des Schlüssels während des Importvorgangs dient. Nur AES-Schlüssel können als Verpackungsschlüssel verwendet werden.

Sie können jeden AES-Schlüssel (in jeder beliebigen Größe) als Verpackungsschlüssel verwenden. Da der Verpackungsschlüssel den Zielschlüssel verpackt und danach sofort entpackt, können Sie einen nur für die aktuelle Sitzung gültigen AES-Schlüssel als Verpackungsschlüssel verwenden. Um zu bestimmen, ob ein Schlüssel als Schlüssel zum Packen verwendet werden kann, führen Sie getAttribute aus, um den Wert des `OBJ_ATTR_WRAP`-Attributs (262) abzurufen. Verwenden Sie genSymKey, um einen Umhüllungsschlüssel und einen AES-Schlüssel (Typ 31) zu erstellen.

Wenn Sie den Parameter `-wk` zum Angeben eines externen Schlüssels zum Packen verwenden, wird der `-w`-Schlüssel zum Packen zwar zum Entpacken, jedoch nicht zum Verpacken des Schlüssels während des Importvorgangs verwendet.

> **Note**
>
> Schlüssel 4 ist ein nicht unterstützter, interner Schlüssel. Wir empfehlen, dass Sie einen AES-Schlüssel als Umhüllungsschlüssel verwenden, den Sie erstellen und verwalten.

Erforderlich: Ja

-wk

Verwenden Sie den AES-Schlüssel in der angegebenen Datei, um den Schlüssel, der importiert wird, zu verpacken. Geben Sie den Pfad und den Namen einer Datei an, die einen Klartext-AES-Schlüssel enthält.

Wenn Sie diesen Parameter angeben, verwendet imSymKey den Schlüssel in der Datei `-wk`, um den zu importierenden Schlüssel zu verpacken, und den Schlüssel im HSM, der durch den `-w`-Parameter festgelegt ist, um ihn zu entpacken. Die Parameterwerte `-w` und `-wk` müssen durch denselben Klartextschlüssel aufgelöst werden.

Standard: Verwenden Sie den Verpackungsschlüssel auf dem HSM zum Entpacken.

Erforderlich: Nein

Verwandte Themen

- genSymKey
- exSymKey
- wrapKey
- unWrapKey
- exportPrivateKey
- exportPubKey

insertMaskedObject

Der insertMaskedObject-Befehl in key_mgmt_util fügt ein maskiertes Objekt aus einer Datei in ein bestimmtes HSM ein. Maskierte Objekte sind geklonte Objekte, die mit dem Befehl extractMaskedObject aus einem HSM extrahiert werden. Sie können nur verwendet werden, nachdem sie wieder in den ursprünglichen Cluster eingefügt wurden. Sie können ein maskiertes Objekt nur in den Cluster einfügen, aus dem es erstellt wurde (oder in eine geklonte Version dieses Clusters). Dies umfasst alle geklonten Versionen des ursprünglichen Clusters, die durch eine regionsübergreifende Sicherungskopie und die anschließende Verwendung dieser Sicherung zum Erstellen eines neuen Clusters generiert wurden.

Maskierte Objekte sind eine effiziente Möglichkeit zum Auslagern und Synchronisieren von Schlüsseln, auch von nicht extrahierbaren Schlüsseln (d. h. Schlüssel mit einem

OBJ_ATTR_EXTRACTABLE-Wert von 0). Auf diese Weise können Schlüssel zwischen Clustern verschiedener Regionen auf sichere Weise synchronisiert werden, ohne die AWS CloudHSM-Konfigurationsdatei aktualisieren zu müssen.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
insertMaskedObject -h

insertMaskedObject -f <filename>
                   [-min_srv <minimum-number-of-servers>]
                   [-timeout <number-of-seconds>]
```

Beispiele

In diesem Beispiel wird gezeigt, wie Sie mit insertMaskedObject ein maskiertes Objekt in ein HSM einfügen.

Example : Einfügen eines maskierten Objekts

Dieser Befehl fügt ein maskiertes Objekt aus einer Datei mit dem Namen `maskedObj` in ein HSM ein. Wenn der Befehl erfolgreich ausgeführt wurde, gibt insertMaskedObject ein Key-Handle für den aus dem maskierten Objekt entschlüsselten Schlüssel sowie eine Erfolgsmeldung zurück.

```
Command: insertMaskedObject -f maskedObj

Cfm3InsertMaskedObject returned: 0x00 : HSM Return: SUCCESS
        New Key Handle: 262433

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-h`

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

`-f`

Gibt den Dateinamen des einzufügenden maskierten Objekts an.

Erforderlich: Ja

`-min_srv`

Gibt die Mindestanzahl der Server an, auf denen das eingefügte maskierte Objekt synchronisiert wird, bevor der Wert des `-timeout`-Parameters verfällt. Falls das Objekt nicht in der vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert ist, wird es nicht eingefügt.

Standard: 1

Erforderlich: Nein

`-timeout`

Gibt an, wie viele Sekunden bei Einschluss des Parameters `min-serv` gewartet wird, bis der Schlüssel serverübergreifend synchronisiert ist. Wenn keine Anzahl angegeben ist, wird die Abfrage ohne zeitliche Einschränkung fortgesetzt.

Standard: keine Einschränkung

Erforderlich: Nein

Verwandte Themen

- extractMaskedObject
- syncKey
- Regionsübergreifendes Kopieren einer Sicherung
- Erstellen eines AWS CloudHSM-Clusters aus einer vorherigen Sicherung

IsValidKeyHandlefile

Der IsValidKeyHandlefile Befehl in key_mgmt_util wird verwendet, um herauszufinden, ob eine Schlüsseldatei einen echten privaten Schlüssel oder einen gefälschten RSA-PEM-Schlüssel enthält. Eine gefälschte PEM-Datei enthält nicht das Material des tatsächlichen privaten Schlüssels, sondern verweist auf den privaten Schlüssel im HSM. Eine solche Datei kann verwendet werden, um die

SSL-/TLS-Auslagerung von Ihrem Webserver nach AWS CloudHSM zu ermöglichen. Weitere Informationen erhalten Sie unter SSL-/TLS-Auslagerung unter Linux.

> **Note**
>
> IsValidKeyHandlefile funktioniert nur für RSA-Schlüssel.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
IsValidKeyHandlefile -h

IsValidKeyHandlefile -f <rsa-private-key-file>
```

Beispiele

Diese Beispiele verdeutlichen, wie mit IsValidKeyHandlefile ermittelt werden kann, ob eine bestimmte Schlüsseldatei das Material des echten Schlüssels oder Material eines gefälschten PEM-Schlüssels enthält.

Example : Validieren eines echten privaten Schlüssels

Mit diesem Befehl wird bestätigt, dass die Datei namens `privateKey.pem` Material des echten Schlüssels enthält.

```
Command: IsValidKeyHandlefile -f privateKey.pem

Input key file has real private key
```

Example : Entwerten eines gefälschten PEM-Schlüssels

Mit diesem Befehl wird bestätigt, dass die Datei namens `caviumKey.pem` Material eines gefälschten PEM-Schlüssels enthält, das aus dem Schlüssel-Handle 15 erstellt wurde.

```
Command: IsValidKeyHandlefile -f caviumKey.pem

Input file has invalid key handle: 15
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-h`

Zeigt die Befehlszeilenhilfe für den Befehl an.

Erforderlich: Ja

`-f`

Gibt die private RSA-Schlüsseldatei an, die auf gültiges Schlüsselmaterial überprüft werden soll.

Erforderlich: Ja

Verwandte Themen

- getCaviumPrivSchlüssel
- SSL/TLS-Auslagerung unter Linux

listAttributes

Der listAttributes-Befehl in key_mgmt_util listet die Attribute eines AWS CloudHSM-Schlüssels und die Konstanten, die sie repräsentieren, auf. Sie nutzen die Konstanten zum Identifizieren der Attribute in den Befehlen getAttribute und setAttribute. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

Dieser Befehl hat keine Parameter.

```
listAttributes
```

Beispiel

Mit diesem Befehl werden die Schlüsselattribute aufgelistet, die Sie mit key_mgmt_util abrufen und ändern können, sowie die Konstanten, die sie darstellen. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Um alle Attribute im Befehl getAttribute in key_mgmt_util darzustellen, verwenden Sie 512.

```
Command: listAttributes

Following are the possible attribute values for getAttributes:

      OBJ_ATTR_CLASS                = 0
      OBJ_ATTR_TOKEN                = 1
      OBJ_ATTR_PRIVATE              = 2
      OBJ_ATTR_LABEL                = 3
      OBJ_ATTR_KEY_TYPE             = 256
      OBJ_ATTR_ENCRYPT              = 260
      OBJ_ATTR_DECRYPT              = 261
      OBJ_ATTR_WRAP                 = 262
      OBJ_ATTR_UNWRAP               = 263
      OBJ_ATTR_SIGN                 = 264
      OBJ_ATTR_VERIFY               = 266
      OBJ_ATTR_LOCAL                = 355
      OBJ_ATTR_MODULUS              = 288
      OBJ_ATTR_MODULUS_BITS         = 289
      OBJ_ATTR_PUBLIC_EXPONENT      = 290
      OBJ_ATTR_VALUE_LEN            = 353
      OBJ_ATTR_EXTRACTABLE          = 354
      OBJ_ATTR_KCV                  = 371
```

Verwandte Themen

- listAttributes in cloudhsm_mgmt_util
- getAttribute
- setAttribute
- Schlüsselattributreferenz

listUsers

Der listUsers-Befehl in key_mgmt_util ruft die Benutzer in den HSMs ab, zusammen mit ihrem Benutzertyp und anderen Attributen.

In key_mgmt_util stellt die Ausgabe des Befehls „listUsers“ alle HSMs im Cluster dar, auch wenn diese nicht konsistent sind. Verwenden Sie zum Abrufen von Informationen über die Benutzer in jedem HSM den Befehl listUsers in cloudhsm_mgmt_util.

Die Benutzerbefehle in key_mgmt_util, listUsers und getKeyInfo sind schreibgeschützte Befehle, zu deren Ausführung Crypto-Benutzer (CUs) berechtigt sind. Die restlichen Benutzerverwaltungsbefehle sind Teil von cloudhsm_mgmt_util. Sie werden von Verschlüsselungsverantwortlichen (CO) mit Benutzerverwaltungsberechtigungen ausgeführt.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
listUsers

listUsers -h
```

Beispiel

Mit diesem Befehl werden die Benutzer von HSMs im Cluster sowie deren Attribute aufgeführt. Sie können das Attribut `User ID` zum Identifizieren von Benutzern in anderen Befehlen, z. B. findKey, getAttribute und getKeyInfo, verwenden.

```
Command:  listUsers

        Number Of Users found 4

        Index       User ID     User Type       User Name           MofnPubKey
 LoginFailureCnt         2FA
        1               1       PCO             admin                   NO
    0               NO
        2               2       AU              app_user                NO
    0               NO
        3               3       CU              alice                   YES
    0               NO
        4               4       CU              bob                     NO
    0               NO
        5               5       CU              trent                   YES
    0               NO

        Cfm3ListUsers returned: 0x00 : HSM Return: SUCCESS
```

Die Ausgabe umfasst die folgenden Benutzerattribute:

- Benutzer-ID: Identifiziert den Benutzer in den Befehlen key_mgmt_util und cloudhsm_mgmt_util.

- User type: Bestimmt die Operationen, die der Benutzer im HSM ausführen kann.
- User Name: Zeigt den benutzerdefinierten Anzeigenamen für den Benutzer an.
- MofnPubKey: Gibt an, ob der Benutzer ein Schlüsselpaar zum Signieren von Token für die Quorum-Authentifizierung registriert hat.
- LoginFailureCnt: Gibt an, wie oft der Benutzer sich nicht erfolgreich angemeldet hat.
- 2FA: Gibt an, dass der Benutzer Multifaktor-Authentifizierung aktiviert hat.

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

Verwandte Themen

- listUsers in cloudhsm_mgmt_util
- findKey
- getAttribute
- getKeyInfo

loginHSM und logoutHSM

Mit den Befehlen loginHSM und logoutHSM in key_mgmt_util können Sie sich bei den HSMs eines Clusters an- und abmelden. Sobald Sie bei den HSMs angemeldet sind, können Sie mit key_mgmt_util eine Vielzahl von Schlüsselverwaltungsoperationen durchführen, z. B. das Erstellen öffentlicher und privater Schlüssel, Synchronisierung und Verpackung.

Bevor Sie den Befehl key_mgmt_util ausführen, müssen Sie key_mgmt_util starten. Zum Verwalten von Schlüsseln mit key_mgmt_util müssen Sie bei den HSMs als Crypto-Benutzer (CU) angemeldet sein.

> **Note**
>
> Wenn Sie mehr als fünf falsche Anmeldeversuche tätigen, wird Ihr Konto gesperrt. Wenn Sie Ihren Cluster vor Februar 2018 erstellt haben, wird Ihr Konto nach 20 falschen

Anmeldeversuchen gesperrt. Ein Crypto Officer (CO) muss Ihr Passwort mit dem Befehl changePswd in cloudhsm_mgmt_util zurücksetzen, um das Konto zu entsperren.
Wenn Sie mehr als ein HSM in Ihrem Cluster haben, sind Ihnen möglicherweise weitere falsche Anmeldeversuche gestattet, bevor Ihr Konto gesperrt wird. Dies liegt daran, dass der CloudHSM-Client die Last auf verschiedene HSMs verteilt. Aus diesem Grund beginnt der Anmeldeversuch möglicherweise nicht jedes Mal auf demselben HSM. Wenn Sie diese Funktion testen, empfehlen wir Ihnen, dies auf einem Cluster mit nur einem aktiven HSM zu tun.

Syntax

```
loginHSM -h

loginHSM -u <user type>
         { -p | -hpswd } <password>
         -s <username>
```

Beispiel

In diesem Beispiel wird gezeigt, wie Sie sich bei den HSMs in einem Cluster mit den Befehlen `logoutHSM` und `loginHSM` an- und abmelden.

Example : Anmelden bei den HSMs

Über diesen Befehl melden Sie sich bei den HSMs als Crypto-Benutzer (CU) mit dem Benutzernamen `example_user` und dem Passwort `aws` an. Die Ausgabe zeigt, dass Sie sich bei allen HSMs im Cluster angemeldet haben.

```
Command:  loginHSM -u CU -s example_user -p aws

Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : Melden Sie sich mit einem versteckten Passwort an

Dieser Befehl entspricht dem obigen Beispiel, außer dass Sie dieses Mal angeben, dass das System das Passwort verbergen soll.

```
Command:  loginHSM -u CU -s example_user -hpswd
```

Sie werden vom System aufgefordert, Ihr Passwort einzugeben. Sie geben das Passwort ein, das System versteckt das Passwort und die Ausgabe zeigt, dass der Befehl erfolgreich war und dass Sie eine Verbindung zu den HSMs hergestellt haben.

```
Enter password:

Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

Command:
```

Example : Abmelden von den HSMs

Mit diesem Befehl melden Sie sich von den HSMs ab. Die Ausgabe zeigt, dass Sie sich bei allen HSMs im Cluster abgemeldet haben.

```
Command: logoutHSM

Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

-h

Zeigt Hilfe für diesen Befehl an.

-u

Gibt den Anmeldebenutzertyp an. Zum Verwenden von key_mgmt_util müssen Sie als CU angemeldet sein.

Erforderlich: Ja

-s

Gibt den Anmeldebenutzernamen an.

Erforderlich: Ja

{ -p | -hpswd }

Geben Sie das Login-Passwort mit `-p` an. Das Passwort wird in Klartext angezeigt, wenn Sie es eingeben. Um Ihr Passwort zu verbergen, verwenden Sie den optionalen `-hpswd`-Parameter anstelle von `-p` und folgen Sie der Aufforderung.

Erforderlich: Ja

Verwandte Themen

- exit

setAttribute

Mit dem setAttribute-Befehl in key_mgmt_util wird ein Schlüssel, der nur für die aktuelle Sitzung gültig ist, in einen persistenten Schlüssel konvertiert, der so lange existiert, bis Sie ihn löschen. Dazu wird der Wert des Token-Attributs des Schlüssels (`OBJ_ATTR_TOKEN`) von „false“ (`0`) in „true“ (`1`) geändert. Sie können nur die Attribute der Schlüssel ändern, deren Eigentümer Sie sind.

Sie können auch den setAttribute-Befehl in cloudhsm_mgmt_util verwenden, um die Attribute label, wrap, unwrap, encrypt und decrypt zu ändern.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
setAttribute -h

setAttribute -o <object handle>
```

```
            -a 1
```

Beispiel

In diesem Beispiel wird gezeigt, wie Sie einen Sitzungsschlüssel in einen persistenten Schlüssel konvertieren.

Der erste Befehl verwendet den Parameter `-sess` von genSymKey zum Erstellen eines 192-Bit-AES-Schlüssels, der nur für die aktuelle Sitzung gültig ist. Die Ausgabe zeigt, dass das Schlüssel-Handle des neuen Sitzungsschlüssels 262154 lautet.

```
Command: genSymKey -t 31 -s 24 -l tmpAES -sess

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 262154

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Dieser Befehl verwendet findKey, um die Sitzungsschlüssel in der aktuellen Sitzung zu suchen. Die Ausgabe verifiziert, dass der Schlüssel 262154 ein Sitzungsschlüssel ist.

```
Command:  findKey -sess 1

Total number of keys present 1

 number of keys matched from start index 0::0
262154

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Dieser Befehl verwendet setAttribute, um den Schlüssel 262154 von einem Sitzungsschlüssel in einen persistenten Schlüssel zu konvertieren. Dazu wird der Wert des Token-Attributs (`OBJ_ATTR_TOKEN`) des Schlüssels von `0` („falsch“) in `1` („wahr“) geändert. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Der Befehl verwendet den Parameter `-o` zum Angeben des Schlüssel-Handles (`262154`) und den Parameter `-a` zum Festlegen der Konstanten, die das Token-Attribut (1) darstellt. Wenn Sie den Befehl ausführen, werden Sie zur Eingabe eines Werts für das Token-Attribut aufgefordert. Der einzige gültige Wert ist 1 (true); der Wert für einen persistenten Schlüssel.

```
Command: setAttribute -o 262154 -a 1
         This attribute is defined as a boolean value.
          Enter the boolean attribute value (0 or 1):1

        Cfm3SetAttribute returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Um sich zu vergewissern, dass der Schlüssel `262154` nun persistent ist, verwendet dieser Befehl findKey für die Suche nach Sitzungsschlüsseln (`-sess 1`) und persistenten Schlüsseln (`-sess 0`). In diesem Fall findet der Befehl zwar keine Sitzungsschlüssel, gibt jedoch `262154` in der Liste der persistenten Schlüssel zurück.

```
Command: findKey -sess 1

Total number of keys present 0

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS



Command: findKey -sess 0

Total number of keys present 5

 number of keys matched from start index 0::4
6, 7, 524296, 9, 262154

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-o

Gibt das Schlüssel-Handle des Zielschlüssels an. Sie können nur jeweils einen Schlüssel in den einzelnen Befehlen angeben. Verwenden Sie zum Abrufen des Schlüssel-Handles eines Schlüssels findKey.

Erforderlich: Ja

-a

Gibt die Konstante an, die das Attribut darstellt, das Sie ändern möchten. Der einzige gültige Wert ist 1, der das Token-Attribut `OBJ_ATTR_TOKEN` darstellt.

Verwenden Sie listAttributes, um die Attribute und deren Ganzzahlwerte abzurufen.

Erforderlich: Ja

Verwandte Themen

- setAttribute in cloudhsm_mgmt_util
- getAttribute
- listAttributes
- Schlüsselattributreferenz

sign

Der sign-Befehl in key_mgmt_util verwendet einen ausgewählten privaten Schlüssel, um eine Signatur für eine Datei zu erzeugen.

Um sign verwenden zu können, müssen Sie zunächst über einen privaten Schlüssel in Ihrem HSM verfügen. Sie können einen privaten Schlüssel mit dem Befehl genSymKey, genRSAKeyPair

oder genECCKeyPair erstellen. Sie können einen privaten Schlüssel auch mit dem Befehl importPrivateKey importieren. Weitere Informationen finden Sie unter Generieren von Schlüsseln.

Der Befehl sign verwendet einen benutzerdefinierten Signierungsmechanismus (dargestellt durch eine Ganzzahl) zum Signieren einer Nachricht. Eine Liste mit möglichen Signierungsmechanismen finden Sie unter Parameter.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
sign -h

sign -f <file name>
     -k <private key handle>
     -m <signature mechanism>
     -out <signed file name>
```

Beispiel

In diesem Beispiel wird gezeigt, wie Sie den Befehl sign verwenden, um eine Datei zu signieren.

Example : Signieren einer Datei

Dieser Befehl signiert eine Datei mit dem Namen `messageFile` anhand eines privaten Schlüssels mit dem Handle `266309`. Er verwendet den `SHA256_RSA_PKCS` (1)-Signierungsmechanismus und speichert die signierte Datei als `signedFile`.

```
Command: sign -f messageFile -k 266309 -m 1 -out signedFile

Cfm3Sign returned: 0x00 : HSM Return: SUCCESS

signature is written to file signedFile

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-f`

Der Name der zu signierenden Datei.

Erforderlich: Ja

`-k`

Das Handle des privaten Schlüssels, der zum Signieren verwendet werden soll.

Erforderlich: Ja

`-m`

Eine Ganzzahl, die den zum Signieren verwendeten Signierungsmechanismus repräsentiert. Die möglichen Mechanismen entsprechen den folgenden Ganzzahlen:

Signierungsmechanismus	Entsprechende Ganzzahl
`SHA1_RSA_PKCS`	0
`SHA256_RSA_PKCS`	1
`SHA384_RSA_PKCS`	2
`SHA512_RSA_PKCS`	3
`SHA224_RSA_PKCS`	4
`SHA1_RSA_PKCS_PSS`	5
`SHA256_RSA_PKCS_PSS`	6
`SHA384_RSA_PKCS_PSS`	7
`SHA512_RSA_PKCS_PSS`	8
`SHA224_RSA_PKCS_PSS`	9
`ECDSA_SHA1`	15
`ECDSA_SHA224`	16
`ECDSA_SHA256`	17

Signierungsmechanismus	Entsprechende Ganzzahl
ECDSA_SHA384	18
ECDSA_SHA512	19

Erforderlich: Ja

-out

Gibt den Namen der Datei an, in die die signierte Datei gespeichert wird.

Erforderlich: Ja

Verwandte Themen

- verify
- importPrivateKey
- genRSAKeyPair
- genECCKeyPair
- genSymKey
- Generieren von Schlüsseln

unWrapKey

Der unWrapKey-Befehl im Tool key_mgmt_util importiert einen verschlüsselten symmetrischen oder privaten Schlüssel aus einer Datei in das HSM. Wurde entwickelt, um verschlüsselte Schlüssel zu importieren, die mit dem Befehl wrapKey in key_mgmt_util verpackt wurden. Kann aber auch verwendet werden, um Schlüssel zu entpacken, die mit anderen Tools verpackt wurden. In diesen Situationen empfehlen wir jedoch, die Softwarebibliotheken PKCS#11 oder JCE zu verwenden, um den Schlüssel zu entpacken.

Importierte Schlüssel funktionieren wie von AWS CloudHSM erzeugte Schlüssel. Der Wert ihres Attributs OBJ_ATTR_LOCAL ist jedoch Null, was bedeutet, dass sie nicht lokal generiert wurden.

Stellen Sie nach dem Importieren eines Schlüssels sicher, dass Sie die Schlüsseldatei markieren oder löschen. Dieser Befehl verhindert nicht das wiederholte Importieren der gleichen Schlüsselinformationen. Das Ergebnis – mehrere Schlüssel mit unterschiedlichen Schlüsselhandles

und demselben Schlüsselmaterial – macht es schwierig, die Verwendung des Schlüsselmaterials zu verfolgen und zu verhindern, dass die kryptografischen Grenzen überschritten werden.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
unWrapKey -h

unWrapKey -f <key-file-name>
          -w <wrapping-key-handle>
          [-sess]
          [-min_srv <minimum-number-of-HSMs>]
          [-timeout <number-of-seconds>]
          [-aad <additional authenticated data filename>]
          [-tag_size <tag size>]
          [-iv_file <IV file>]
          [-attest]
          [-m <wrapping-mechanism>]
          [-t <hash-type>]
          [-nex]
          [-u <user id list>]
          [-m_value <number of users needed for approval>]
          [-noheader]
          [-l <key-label>]
          [-id <key-id>]
          [-kt <key-type>]
          [-kc <key-class]
          [-i <unwrapping-IV>]
```

Beispiel

Diese Beispiele zeigen, wie man mit unWrapKey einen verpackten Schlüssel aus einer Datei in die HSMs importiert. Im ersten Beispiel entpacken wir einen Schlüssel, der mit dem Befehl wrapKey key_mgmt_util verpackt wurde und daher einen Header hat. Im zweiten Beispiel entpacken wir einen Schlüssel, der außerhalb von key_mgmt_util verpackt wurde und daher keinen Header hat.

Example : Einen Schlüssel entpacken (mit Header)

Dieser Befehl importiert eine verpackte Kopie eines symmetrischen 3DES-Schlüssels in ein HSM. Der Schlüssel wird mit einem AES-Schlüssel mit dem Bezeichner 6 entpackt, der mit demjenigen

kryptographisch identisch ist, der zum Verpacken des 3DES-Schlüssels verwendet wurde. Die Ausgabe zeigt, dass der Schlüssel in der Datei entpackt und importiert wurde und dass das Handle des importierten Schlüssels 29 ist.

```
        Command:  unWrapKey -f 3DES.key -w 6 -m 4

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Key Unwrapped.  Key Handle: 29

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : Einen Schlüssel entpacken (kein Header)

Dieser Befehl importiert eine verpackte Kopie eines symmetrischen 3DES-Schlüssels in ein HSM. Der Schlüssel wird mit einem AES-Schlüssel mit dem Bezeichner 6 entpackt, der mit demjenigen kryptographisch identisch ist, der zum Verpacken des 3DES-Schlüssels verwendet wurde. Da dieser 3DES-Schlüssel nicht mit key_mgmt_util verpackt wurde, wird der `noheader`-Parameter zusammen mit den erforderlichen Begleitparametern angegeben: ein Schlüssellabel (`unwrapped3DES`), eine Schlüsselklasse (4) und ein Schlüsseltyp (21). Die Ausgabe zeigt, dass der Schlüssel in der Datei entpackt und importiert wurde und dass das Handle des importierten Schlüssels 8 ist.

```
Command: unWrapKey -f 3DES.key -w 6 -noheader -l unwrapped3DES -kc 4 -kt 21 -m 4

      Cfm3CreateUnwrapTemplate2 returned: 0x00 : HSM Return: SUCCESS
       Cfm2UnWrapWithTemplate3 returned: 0x00 : HSM Return: SUCCESS

       Key Unwrapped. Key Handle: 8

       Cluster Error Status
       Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
       Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-f

Gibt den Pfad und Namen der Datei an, die den verpackten Schlüssel enthält.

Erforderlich: Ja

-w

Gibt den Wrapping-Schlüssel an. Geben Sie das Schlüssel-Handle eines AES-Schlüssels oder RSA-Schlüssels auf dem HSM ein. Dieser Parameter muss angegeben werden. Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Um einen Wrapping-Schlüssel zu erstellen, verwenden Sie GenSymKey, um einen AES-Schlüssel (Typ 31) zu generieren, oder genRSAKeyPair, um ein RSA-Schlüsselpaar zu generieren (Typ 0). Wenn Sie ein RSA-Schlüsselpaar verwenden, achten Sie darauf, den Schlüssel mit einem der Schlüssel zu umwickeln und ihn mit dem anderen zu entpacken. Verwenden Sie getAttribute, um zu überprüfen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann, und um den Wert des `OBJ_ATTR_WRAP`-Attributs abzurufen, der von der Konstanten `262` dargestellt wird.

Erforderlich: Ja

-sess

Erstellt einen Schlüssel, der nur in der aktuellen Sitzung existiert. Der Schlüssel kann nach Ende der Sitzung nicht wiederhergestellt werden.

Verwenden Sie diesen Parameter, wenn Sie einen Schlüssel nur für kurze Zeit benötigen, z. B. einen Schlüssel, der einen anderen Schlüssel verschlüsselt und dann schnell entschlüsselt. Verwenden Sie keinen Sitzungsschlüssel, um Daten zu verschlüsseln, die Sie nach dem Ende der Sitzung möglicherweise entschlüsseln müssen.

Um einen Sitzungsschlüssel in einen persistenten (Token-)Schlüssel zu ändern, verwenden Sie setAttribute.

Standard: Der Schlüssel ist persistent.

Erforderlich: Nein

-min_srv

Gibt die Mindestanzahl der HSMs an, in denen der Schlüssel synchronisiert wird, bevor der Wert des Parameters `-timeout` verfällt. Falls der Schlüssel nicht in der zulässigen vorgegebenen Zeit mit der angegebenen Anzahl von Servern synchronisiert wird, wird er nicht erstellt.

AWS CloudHSM synchronisiert automatisch jeden Schlüssel mit jedem HSM im Cluster. Zur Beschleunigung Ihres Prozesses legen Sie den Wert von `min_srv` auf weniger als die Anzahl der HSMs im Cluster fest und stellen einen niedrigen Zeitüberschreitungswert ein. Beachten Sie jedoch, dass einige Anfragen möglicherweise keinen Schlüssel generieren.

Standard: 1

Erforderlich: Nein

-timeout

Gibt an, wie lange (in Sekunden) der Befehl darauf wartet, dass ein Schlüssel mit der im `min_srv`-Parameter angegebenen Anzahl von HSMs synchronisiert wird.

Dieser Parameter ist nur gültig, wenn der `min_srv`-Parameter auch im Befehl verwendet wird.

Voreinstellung: Keine Zeitüberschreitung. Der Befehl wartet auf unbestimmte Zeit und kehrt erst zurück, wenn der Schlüssel mit der Mindestanzahl von Servern synchronisiert ist.

Erforderlich: Nein

-attest

Führt eine Integritätsprüfung durch, die sicherstellt, dass die Firmware, auf der der Cluster läuft, nicht manipuliert wurde.

Standard: Keine Bescheinigungsprüfung.

Erforderlich: Nein

-nex

Macht den Schlüssel nicht extrahierbar. Der generierte Schlüssel kann nicht aus dem HSM exportiert werden.

Standard: Der Schlüssel ist extrahierbar.

Erforderlich: Nein

-m

Der Wert, der den Verschlüsselungsmechanismus bezeichnet. CloudHSM unterstützt die folgenden Mechanismen:

Mechanismus	Wert
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (zur maximalen Datengröße lesen Sie den Hinweis weiter unten in diesem Abschnitt)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (Zur maximalen Datengröße lesen Sie den Hinweis weiter unten in diesem Abschnitt.) Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.	12

Erforderlich: Ja

> **Note**
>
> Bei Verwendung des RSA_OAEP-Verpackungsmechanismus wird die maximale verpackbare Schlüsselgröße durch den Modul-Wert des RSA-Schlüssels und die Länge des angegebenen Hash wie folgt bestimmt: Maximale Schlüsselgröße = modulusLengthInBytes-(2*hashLengthInBytes)-2.
> Wenn Sie den RSA_PKCS-Wrapping-Mechanismus verwenden, wird die maximale Schlüsselgröße, die Sie wrappen können, wie folgt durch den Modulus des RSA-Schlüssels bestimmt: Maximale Schlüsselgröße = (modulusLengthInBytes -11).

-t

Hash-Algorithmus	Wert
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (gültig für RSA_AES- und RSA_OAEP-Mechanismen)	6

Erforderlich: Nein

-noheader

Wenn Sie einen Schlüssel entpacken, der außerhalb von key_mgmt_util verpackt wurde, müssen Sie diesen Parameter und alle anderen zugehörigen Parameter angeben.

Erforderlich: Nein

> **Note**
>
> Wenn Sie diesen Parameter angeben, müssen Sie auch die folgenden `-noheader`-Parameter angeben:

- -l

  Gibt den Bezeichner an, der dem entpackten Schlüssel hinzugefügt werden soll.

  Erforderlich: Ja

- -kc

  Gibt die Klasse des zu entpackenden Schlüssels an. Die folgenden Werte sind zulässig:

  3 = privater Schlüssel aus einem öffentlich-privaten Schlüsselpaar

4 = geheimer Schlüssel (symetrisch)

Erforderlich: Ja

- -kt

  Gibt den Typ des zu entpackenden Schlüssels an. Die folgenden Werte sind zulässig:

  0 = RSA

  1 = DSA

  3 = ECC

  16 = GENERIC_SECRET

  21 = DES3

  31 = AES

  Erforderlich: Ja

Sie können auch optional die folgenden `-noheader`-Parameter angeben:

- -id

  Die ID, die dem entpackten Schlüssel hinzugefügt werden soll.

  Erforderlich: Nein

- -i

  Der zu verwendende Initialisierungsvektor (IV) für das Entpacken.

  Erforderlich: Nein

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Verwandte Themen

- wrapKey
- exSymKey

- imSymKey

verify

Mit dem verify-Befehl in key_mgmt_util überprüfen Sie, ob eine Datei von einem bestimmten Schlüssel signiert wurde. Hierzu vergleicht der Befehl verify eine signierte Datei mit einer Quelldatei und analysiert auf der Grundlage eines bestimmten öffentlichen Schlüssels und eines Signierungsmechanismus, ob in kryptografischer Hinsicht ein Zusammenhang zwischen ihnen besteht. Dateien können in AWS CloudHSM mit der Operation sign signiert werden.

Signierungsmechanismen werden durch die im Abschnitt Parameter aufgeführten Ganzzahlen dargestellt.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
verify -h

verify -f <message-file>
       -s <signature-file>
       -k <public-key-handle>
       -m <signature-mechanism>
```

Beispiel

Diese Beispiele zeigen, wie mit verify überprüft werden kann, ob ein bestimmter öffentlicher Schlüssel zum Signieren einer bestimmten Datei verwendet wurde.

Example : Überprüfen einer Dateisignatur

Mit diesem Befehl soll überprüft werden, ob eine Datei mit dem Namen `hardwarCert.crt` mit dem öffentlichen Schlüssel `262276` anhand des Signierungsmechanismus `SHA256_RSA_PKCS` signiert wurde, um die signierte Datei `hardwareCertSigned` zu erzeugen. Da es sich bei den angegebenen Parametern um ein echtes Signierungsverhältnis handelt, gibt der Befehl eine Erfolgsmeldung zurück.

```
Command: verify -f hardwareCert.crt -s hardwareCertSigned -k 262276 -m 1

Signature verification successful
```

```
Cfm3Verify returned: 0x00 : HSM Return: SUCCESS
```

Example : Nachweisen eines ungültigen Signierungsverhältnisses

Dieser Befehl überprüft, ob eine Datei mit dem Namen `hardwareCert.crt` mit dem öffentlichen Schlüssel `262276` anhand des Signierungsmechanismus `SHA256_RSA_PKCS` signiert wurde, um die signierte Datei `userCertSigned` zu erzeugen. Da die angegebenen Parameter kein echtes Signierungsverhältnis darstellen, gibt der Befehl eine Fehlermeldung zurück.

```
Command: verify -f hardwarecert.crt -s usercertsigned -k 262276 -m 1
Cfm3Verify returned: 0x1b

CSP Error: ERR_BAD_PKCS_DATA
```

Parameter

Dieser Befehl erfordert die folgenden Parameter.

`-f`

Der Name der ursprünglichen Nachrichtendatei.

Erforderlich: Ja

`-s`

Der Name der signierten Datei.

Pflichtfeld: Ja

`-k`

Das Handle des öffentlichen Schlüssels, der vermutlich zum Signieren der Datei verwendet wird.

Erforderlich: Ja

`-m`

Eine Ganzzahl, die den vorgeschlagenen und zum Signieren der Datei verwendeten Signierungsmechanismus repräsentiert. Die möglichen Mechanismen entsprechen den folgenden Ganzzahlen:

Signierungsmechanismus	Entsprechende Ganzzahl
SHA1_RSA_PKCS	0
SHA256_RSA_PKCS	1
SHA384_RSA_PKCS	2
SHA512_RSA_PKCS	3
SHA224_RSA_PKCS	4
SHA1_RSA_PKCS_PSS	5
SHA256_RSA_PKCS_PSS	6
SHA384_RSA_PKCS_PSS	7
SHA512_RSA_PKCS_PSS	8
SHA224_RSA_PKCS_PSS	9
ECDSA_SHA1	15
ECDSA_SHA224	16
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

Erforderlich: Ja

Verwandte Themen

- sign
- getCert
- Generieren von Schlüsseln

wrapKey

Der wrapKey-Befehl in key_mgmt_util exportiert eine verschlüsselte Kopie eines symmetrischen oder privaten Schlüssels aus dem HSM in eine Datei. Wenn Sie wrapKey ausführen, geben Sie den zu exportierenden Schlüssel, einen Schlüssel in dem HSM zum Verschlüsseln (Wrap) des zu exportierenden Schlüssels und die Ausgabedatei an.

Der Befehl `wrapKey` schreibt den verschlüsselten Schlüssel in eine von Ihnen angegebene Datei, entfernt den Schlüssel jedoch nicht aus dem HSM oder verhindert, dass Sie ihn in kryptographischen Operationen verwenden. Sie können den gleichen Schlüssel mehrmals exportieren.

Nur der Eigentümer eines Schlüssels, d.h. der Crypto-Benutzer (CU), der den Schlüssel angelegt hat, kann ihn exportieren. Benutzer, für die der Schlüssel freigegeben ist, können ihn in kryptografischen Vorgängen verwenden, können ihn aber nicht exportieren.

Um den verschlüsselten Schlüssel wieder in das HSM zu importieren, verwenden Sie unWrapKey. Um einen Klartextschlüssel aus einem HSM zu exportieren, verwenden Sie exSymKey oder exportPrivateKey. Der Befehl aesWrapUnwrap kann keine Schlüssel entschlüsseln (entpacken), die mit wrapKey verschlüsselt werden.

Bevor Sie einen key_mgmt_util-Befehl ausführen, müssen Sie key_mgmt_util starten und sich am HSM als Crypto-Benutzer (CU) anmelden.

Syntax

```
wrapKey -h

wrapKey -k <exported-key-handle>
        -w <wrapping-key-handle>
        -out <output-file>
        [-m <wrapping-mechanism>]
        [-aad <additional authenticated data filename>]
        [-t <hash-type>]
        [-noheader]
        [-i <wrapping IV>]
        [-iv_file <IV file>]
        [-tag_size <num_tag_bytes>>]
```

Beispiel

Example

Mit diesem Befehl wird ein symmetrischer 192-Bit-3DES-Schlüssel exportiert (Schlüssel-Handle 7). Der Befehl verwendet einen 256-Bit-AES-Schlüssel im HSM (Schlüssel-Handle 14), um den Schlüssel 7 zu verpacken. Anschließend wird der verschlüsselte 3DES-Schlüssel in die Datei `3DES-encrypted.key` geschrieben.

Die Ausgabe zeigt, dass der Schlüssel 7 (der 3DES-Schlüssel) erfolgreich verpackt und in die angegebene Datei geschrieben wurde. Der verschlüsselte Schlüssel hat eine Länge von 307 Byte.

```
        Command:  wrapKey -k 7 -w 14 -out 3DES-encrypted.key -m 4

        Key Wrapped.

        Wrapped Key written to file "3DES-encrypted.key length 307

        Cfm2WrapKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

-h

Zeigt Hilfe für den Befehl an.

Erforderlich: Ja

-k

Der Schlüssel-Handle des Schlüssels, den Sie exportieren möchten. Geben Sie das Schlüssel-Handle eines symmetrischen oder privaten Schlüssels ein, der Ihnen gehört. Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Verwenden Sie den Befehl getAttribute, um sicherzustellen, dass ein Schlüssel exportiert werden kann, und um den Wert des Attributs `OBJ_ATTR_EXTRACTABLE` abzurufen, der von der Konstanten `354` dargestellt wird. Hilfe zur Interpretation der Schlüsselattribute finden Sie unter Schlüsselattributreferenz.

Zudem können Sie nur Schlüssel exportieren, deren Eigentümer Sie sind. Um den Eigentümer eines Schlüssels zu finden, verwenden Sie den Befehl getKeyInfo.

Erforderlich: Ja

-w

Gibt den Wrapping-Schlüssel an. Geben Sie das Schlüssel-Handle eines AES-Schlüssels oder RSA-Schlüssels auf dem HSM ein. Dieser Parameter muss angegeben werden. Verwenden Sie den findKey-Befehl, um Schlüssel-Handles zu finden.

Um einen Wrapping-Schlüssel zu erstellen, verwenden Sie GenSymKey, um einen AES-Schlüssel (Typ 31) zu generieren, oder genRSAKeyPair, um ein RSA-Schlüsselpaar zu generieren (Typ 0). Wenn Sie ein RSA-Schlüsselpaar verwenden, achten Sie darauf, den Schlüssel mit einem der Schlüssel zu umwickeln und ihn mit dem anderen zu entpacken. Verwenden Sie getAttribute, um zu überprüfen, ob ein Schlüssel als Verpackungsschlüssel verwendet werden kann, und um den Wert des `OBJ_ATTR_WRAP`-Attributs abzurufen, der von der Konstanten `262` dargestellt wird.

Erforderlich: Ja

-out

Pfad und Name der Ausgabedatei. Wenn der Befehl erfolgreich ist, enthält diese Datei eine verschlüsselte Kopie des exportierten Schlüssels. Wenn die Datei bereits vorhanden ist, überschreibt der Befehl sie ohne Warnung.

Erforderlich: Ja

-m

Der Wert, der den Verschlüsselungsmechanismus bezeichnet. CloudHSM unterstützt die folgenden Mechanismen:

Mechanismus	Wert
`AES_KEY_WRAP_PAD_PKCS5`	4
`NIST_AES_WRAP_NO_PAD`	5
`NIST_AES_WRAP_PAD`	6
`RSA_AES`	7
`RSA_OAEP` (zur maximalen Datengröße lesen Sie den Hinweis weiter unten in diesem Abschnitt)	8

Mechanismus	Wert
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (Zur maximalen Datengröße lesen Sie den Hinweis weiter unten in diesem Abschnitt.) Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.	12

Erforderlich: Ja

> **Note**
>
> Bei Verwendung des RSA_0AEP-Verpackungsmechanismus wird die maximale verpackbare Schlüsselgröße durch den Modul-Wert des RSA-Schlüssels und die Länge des angegebenen Hash wie folgt bestimmt: Maximale Schlüsselgröße = (modulusLengthInBytes-2*hashLengthInBytes-2).
> Wenn Sie den RSA_PKCS-Wrapping-Mechanismus verwenden, wird die maximale Schlüsselgröße, die Sie wrappen können, wie folgt durch den Modulus des RSA-Schlüssels bestimmt: Maximale Schlüsselgröße = (modulusLengthInBytes -11).

-t

Der Wert, der den Hash-Algorithmus darstellt. CloudHSM unterstützt die folgenden Algorithmen:

Hash-Algorithmus	Wert
SHA1	2
SHA256	3
SHA384	4
SHA512	5

Hash-Algorithmus	Wert
SHA224 (gültig für RSA_AES- und RSA_OAEP-Mechanismen)	6

Erforderlich: Nein

AAD

Der Dateiname mit AAD.

> **Note**
>
> Gültig nur für AES_GCM- und CLOUDHSM_AES_GCM-Mechanismen.

Erforderlich: Nein

-noheader

Lässt den Header weg, der CloudHSM-spezifische Schlüsselattribute angibt. Verwenden Sie diesen Parameter nur, wenn Sie planen, den Schlüssel mit Tools außerhalb von key_mgmt_util zu entpacken.

Erforderlich: Nein

-i

Der Initialisierungsvektor (IV) (Hex-Wert).

> **Note**
>
> Gültig nur, wenn mit dem -noheader-Parameter für CLOUDHSM_AES_KEY_WRAP- und NIST_AES_WRAP-Mechanismen übergeben.

Erforderlich: Nein

-iv_file

Die Datei, in die Sie den als Antwort erhaltenen IV-Wert schreiben möchten.

> **Note**
>
> Gültig nur, wenn mit dem `-noheader`-Parameter für den `AES_GCM`-Mechanismus übergeben.

Erforderlich: Nein

-tag_size

Die Größe des Tags, das zusammen mit verschlüsseltem Blob gespeichert werden soll.

> **Note**
>
> Gültig nur, wenn mit dem `-noheader`-Parameter für `AES_GCM`- und `CLOUDHSM_AES_GCM`-Mechanismen übergeben. Die Mindestgröße der Tags ist acht.

Erforderlich: Nein

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Verwandte Themen

- exSymKey
- imSymKey
- unWrapKey

Schlüsselattributreferenz

Die key_mgmt_util-Befehle verwenden Konstanten, um die Attribute von Schlüsseln in einer HSM darzustellen. Dieses Thema kann Ihnen helfen, die Attribute zu identifizieren, die Konstanten zu finden, die sie in Befehlen darstellen, und deren Werte zu verstehen.

Sie legen die Attribute eines Schlüssels fest, wenn Sie ihn erstellen. Um das Token-Attribut zu ändern, das angibt, ob ein Schlüssel persistent ist oder nur in der Sitzung existiert, verwenden Sie den Befehl setAttribute in key_mgmt_util. Um die Attribute label, wrap, unwrap, encrypt oder decrypt zu ändern, verwenden Sie den `setAttribute`-Befehl in cloudhsm_mgmt_util.

Verwenden Sie listAttributes, um eine Liste der Attribute und deren Konstanten abzurufen. Um die Attributwerte für einen Schlüssel abzurufen, verwenden Sie getAttribute.

In der folgenden Tabelle finden Sie die Schlüsselattribute, ihre Konstanten und ihre gültige Werte.

Attribut	Konstante	Werte
OBJ_ATTR_ALL	512	Repräsentiert alle Attribute.
OBJ_ATTR_ALWAYS_SENSITIVE	357	0: False. 1: True.
OBJ_ATTR_CLASS	0	2: Öffentlicher Schlüssel in einem öffentlich-privaten Schlüsselpaar. 3: Privater Schlüssel in einem öffentlich-privaten Schlüssel paar. 4: Geheimer (symmetrischer) Schlüssel.
OBJ_ATTR_DECRYPT	261	0: False. 1: True. Der Schlüssel kann zum Entschlüsseln von Daten verwendet werden.
OBJ_ATTR_DERIVE	268	0: False. 1: True. Die Funktion leitet den Schlüssel ab.
OBJ_ATTR_DESTROYABLE	370	0: False. 1: True.
OBJ_ATTR_ENCRYPT	260	0: False.

Attribut	Konstante	Werte
		1: True. Der Schlüssel kann zum Verschlüsseln von Daten verwendet werden.
OBJ_ATTR_EXTRACTABLE	354	0: False. 1: True. Der Schlüssel kann aus den HSMs exportiert werden.
OBJ_ATTR_ID	258	Benutzerdefinierte Zeichenfolge. Muss im Cluster eindeutig sein. Der Standardwert ist eine leere Zeichenfolge.
OBJ_ATTR_KCV	371	Schlüsselprüfwert des Schlüssels. Weitere Informationen finden Sie unter Weitere Details.
OBJ_ATTR_KEY_TYPE	256	0: RSA. 1: DSA. 3: EC. 16: Allgemeiner geheimer Schlüssel. 18: RC4. 21: Triple DES (3DES). 31: AES.
OBJ_ATTR_LABEL	3	Benutzerdefinierte Zeichenfolge. Muss im Cluster nicht eindeutig sein.

Attribut	Konstante	Werte
OBJ_ATTR_LOCAL	355	0. Falsch. Der Schlüssel wurde in die HSMs importiert. 1: True.
OBJ_ATTR_MODULUS	288	Der Modulus, der zum Generieren eines RSA-Schlüsselpaares verwendet wurde. Bei EC-Schlüsseln steht dieser Wert für die DER-Kodierung des ANSI X9.62 ECPoint-Wertes „Q“ in einem hexadezimalen Format. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.
OBJ_ATTR_MODULUS_BITS	289	Die Länge des Modulus, der zum Generieren eines RSA-Schlüsselpaares verwendet wurde. Bei EC-Schlüsseln steht dies für die ID der elliptischen Kurve, die zur Generierung des Schlüssels verwendet wurde. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.
OBJ_ATTR_NEVER_EXTRACTABLE	356	0: False. 1: True. Der Schlüssel kann nicht aus den HSMs exportiert werden.

Attribut	Konstante	Werte
OBJ_ATTR_PUBLIC_EXPONENT	290	Der öffentliche Exponent, der zum Generieren eines RSA-Schlüsselpaares verwendet wurde. Für andere Schlüsseltypen ist dieses Attribut nicht vorhanden.
OBJ_ATTR_PRIVATE	2	0: False. 1: True. Dieses Attribut gibt an, ob nicht authentifizierte Benutzer die Attribute des Schlüssels auflisten können. Da der CloudHSM PKCS # 11-Provider derzeit keine öffentlichen Sitzungen unterstützt, ist bei allen Schlüsseln (einschließlich öffentlicher Schlüssel in einem öffentlich-privaten Schlüsselpaar) dieses Attribut auf 1 festgelegt.
OBJ_ATTR_SENSITIVE	259	0: False. Öffentlicher Schlüssel in einem öffentlich-privaten Schlüsselpaar. 1: True.
OBJ_ATTR_SIGN	264	0: False. 1: True. Der Schlüssel kann zum Signieren verwendet werden (private Schlüssel).

Attribut	Konstante	Werte
OBJ_ATTR_TOKEN	1	0: False. Sitzungsschlüssel. 1: True. Persistenter Schlüssel.
OBJ_ATTR_TRUSTED	134	0: False. 1: True.
OBJ_ATTR_UNWRAP	263	0: False. 1: True. Der Schlüssel kann zum Entschlüsseln von Schlüsseln verwendet werden.
OBJ_ATTR_UNWRAP_TEMPLATE	1073742354	Für die Werte sollte die Attributvorlage verwendet werden, die auf jeden Schlüssel angewendet wird, der mit diesem Schlüssel zum Packen entpackt wird.
OBJ_ATTR_VALUE_LEN	353	Schlüssellänge in Bytes.
OBJ_ATTR_VERIFY	266	0: False. 1: True. Der Schlüssel für die Verifizierung verwendet werden (öffentliche Schlüssel).
OBJ_ATTR_WRAP	262	0: False. 1: True. Der Schlüssel kann zum Verschlüsseln von Schlüsseln verwendet werden.

Attribut	Konstante	Werte
OBJ_ATTR_WRAP_TEMPLATE	1073742353	Für die Werte sollte die Attributvorlage verwendet werden, die dem Schlüssel entspricht, der mit diesem Schlüssel zum Packen gepackt wurde.
OBJ_ATTR_WRAP_WITH_TRUSTED	528	0: False. 1: True.

Weitere Details

Schlüsselprüfwert (Key Check Value, KCV)

Der Schlüsselprüfwert (KCV) ist ein 3-Byte-Hash oder eine Prüfsumme eines Schlüssels, der generiert wird, wenn das HSM einen Schlüssel importiert oder generiert. Sie können einen KCV auch außerhalb des HSM berechnen, z. B. nachdem Sie einen Schlüssel exportiert haben. Anschließend können Sie die KCVs vergleichen, um die Identität und Integrität des Schlüssels zu bestätigen. Um den KCV eines Schlüssels abzurufen, verwenden Sie getAttribute.

AWS CloudHSM verwendet die folgende Standardmethode, um einen Schlüsselprüfwert zu generieren:

- Symmetrische Schlüssel: Die ersten 3 Byte des Ergebnisses der Verschlüsselung eines Nullblocks mit dem Schlüssel.
- Asymmetrische Schlüsselpaare: Die ersten 3 Byte des SHA-1-Hashs des öffentlichen Schlüssels.
- HMAC-Schlüssel: KCV für HMAC-Schlüssel wird derzeit nicht unterstützt.

AWS CloudHSM Kunden-SDKs

Verwenden Sie ein Client-SDK, um kryptografische Operationen von plattform- oder sprachbasierten Anwendungen auf Hardware-Sicherheitsmodule (HSMs) auszulagern.

AWS CloudHSM bietet zwei Hauptversionen, und Client SDK 5 ist die neueste Version. Es bietet eine Reihe von Vorteilen gegenüber Client-SDK 3 (der vorherigen Serie). Weitere Informationen finden Sie unter Vorteile von Client-SDK 5. Informationen zur Plattformunterstützung finden Sie unter Client-SDK 5 unterstützte Plattformen.

Informationen zur Verwendung von Client-SDK 3 finden Sie unter Vorheriges Client-SDK (Client-SDK 3).

the section called “PKCS #11-Bibliothek”

PKCS #11 ist ein Standard für die Ausführung von kryptografischen Vorgängen auf Hardwaresicherheitsmodulen (HSMs). AWS CloudHSM bietet Implementierungen der PKCS #11-Bibliothek, die mit PKCS #11 Version 2.40 kompatibel sind.

the section called “OpenSSL Dynamic Engine”

Mit der AWS CloudHSM OpenSSL Dynamic Engine können Sie kryptografische Operationen über die OpenSSL-API auf Ihren CloudHSM-Cluster auslagern.

the section called “JCE-Anbieter”

Der AWS CloudHSM JCE-Anbieter ist mit der Java Cryptographic Architecture (JCA) kompatibel. Der Anbieter ermöglicht es Ihnen, kryptografische Operationen auf dem HSM durchzuführen.

the section called “KSP- und CNG-Anbieter”

Der AWS CloudHSM Client für Windows umfasst CNG- und KSP-Anbieter. Derzeit unterstützt nur Client-SDK 3 CNG- und KSP-Anbieter.

Client-SDK 5 unterstützte Plattformen

Die AWS CloudHSM Basisunterstützung ist für jede Version des Client-SDK unterschiedlich. Die Plattformunterstützung für Komponenten in einem SDK entspricht in der Regel der Basisunterstützung, jedoch nicht immer. Um die Plattformunterstützung für eine bestimmte Komponente zu ermitteln, stellen Sie zunächst sicher, dass die gewünschte Plattform im Basisbereich

des SDK angezeigt wird, und suchen Sie dann im Komponentenabschnitt nach Ausnahmen oder anderen relevanten Informationen.

AWS CloudHSM unterstützt nur 64-Bit-Betriebssysteme.

Die Plattformunterstützung ändert sich im Laufe der Zeit. Frühere Versionen des CloudHSM-Client SDK unterstützen möglicherweise nicht alle hier aufgeführten Betriebssysteme. Ermitteln Sie anhand der Versionshinweise die Betriebssystemunterstützung für frühere Versionen des CloudHSM-Client SDK. Weitere Informationen finden Sie unter Downloads für das AWS CloudHSM Client-SDK.

Informationen zu den unterstützten Plattformen für das vorherige Client-SDK finden Sie unter Unterstützte Plattformen vom Client-SDK 3.

Für das Client-SDK 5 ist kein Client-Daemon erforderlich.

Linux-Unterstützung für Client-SDK 5

Unterstützte Plattformen	X86_64-Architektur	ARM-Architektur
Amazon Linux 2	Ja	Ja
Amazon Linux 2023	Ja	Ja
CentOS 7 (7,8+)	Ja	Nein
RedHat Enterprise Linux 7 (7.8+)	Ja	Nein
RedHat Enterprise Linux 8 (8,3 und höher)	Ja	Nein
RedHat Enterprise Linux 9 (9,2 und höher)	Ja	Ja
Ubuntu 20.04 LTS	Ja	Nein
Ubuntu 22.04 LTS	Ja	Ja

Hinweis: SDK 5.4.2 war die letzte Version, die CentOS 8-Plattformunterstützung bot. Weitere Informationen finden Sie auf der CentOS-Website.

Windows-Unterstützung für Client-SDK 5

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Serverless-Unterstützung für Client-SDK 5

- AWS Lambda
- Docker/ECS

Unterstützung für Komponenten

PKCS-#11-Bibliothek

Die PKCS-#11-Bibliothek ist eine plattformübergreifende Komponente, die der Basisunterstützung von Linux und Windows Client-SDK 5 entspricht. Weitere Informationen finden Sie unter the section called “Linux-Unterstützung für Client-SDK 5” und the section called “Windows-Unterstützung für Client-SDK 5”.

OpenSSL Dynamic Engine

Die OpenSSL Dynamic Engine ist eine reine Linux-Komponente, die OpenSSL 1.0.2, 1.1.1 oder 3.x benötigt.

JCE-Anbieter

Der JCE-Anbieter ist ein Java-SDK, das auf allen unterstützten Plattformen mit OpenJDK 8, OpenJDK 11, OpenJDK 17 und OpenJDK 21 kompatibel ist.

Vorteile von Client-SDK 5

Im Vergleich zu Client-SDK 3 ist Client-SDK 5 einfacher zu verwalten, bietet eine bessere Konfigurierbarkeit und eine höhere Zuverlässigkeit. Client-SDK 5 bietet außerdem einige zusätzliche wichtige Vorteile gegenüber Client-SDK 3.

Konzipiert für Serverless-Architekturen

Das Client-SDK 5 benötigt keinen Client-Daemon, sodass Sie keinen Hintergrunddienst mehr verwalten müssen. Dies hilft Benutzern in einigen wichtigen Punkten:

- Vereinfacht den Startvorgang der Anwendung. Um mit CloudHSM zu beginnen, müssen Sie lediglich das SDK konfigurieren, bevor Sie Ihre Anwendung ausführen.
- Sie benötigen keinen ständig laufenden Prozess, was die Integration mit Serverless-Komponenten wie Lambda und Elastic Container Service (ECS) erleichtert.

Bessere Integrationen von Drittanbietern und einfachere Portabilität

Das Client-SDK 5 folgt genau der JCE-Spezifikation und bietet eine einfachere Portabilität zwischen verschiedenen JCE-Anbietern sowie bessere Integrationen von Drittanbietern

Verbesserte Benutzererfahrung und Konfigurierbarkeit

Das Client-SDK 5 verbessert die Lesbarkeit von Protokollnachrichten und bietet klarere Ausnahmen und Mechanismen zur Fehlerbehandlung, was die Self-Service-Suche für Benutzer erheblich erleichtert. SDK 5 bietet auch eine Vielzahl von Konfigurationen, die auf der Seite Configure Tool aufgeführt sind.

Breitere Plattformunterstützung

Client-SDK 5 bietet mehr Unterstützung für moderne Betriebsplattformen. Dies beinhaltet Unterstützung für ARM-Technologien und eine bessere Unterstützung für JCE, PKCS #11 und OpenSSL. Weitere Informationen finden Sie unter Unterstützte Plattformen.

Zusätzliche Funktionen und Mechanismen

Client-SDK 5 enthält zusätzliche Funktionen und Mechanismen, die in Client-SDK 3 nicht verfügbar sind, und Client-SDK 5 wird in Zukunft weitere Mechanismen hinzufügen.

Migration von Client-SDK 3 zu Client-SDK 5

Ausführliche Anweisungen zur Migration von Client-SDK 3 zu Client-SDK 5 finden Sie in den Migrationsanweisungen für jedes einzelne Client-SDK:

- Migrieren Sie Ihre PKCS #11-Bibliothek von Client-SDK 3 zu Client-SDK 5
- Migrieren Sie Ihre OpenSSL Dynamic Engine von Client SDK 3 auf Client SDK 5
- Migrieren Sie Ihren JCE-Anbieter von Client SDK 3 auf Client SDK 5
- Migrieren von Client-SDK 3 CMU und KMU zu Client-SDK 5 CloudHSM CLI

Für Funktionen oder Anwendungsfälle, die von der CloudHSM-CLI nicht unterstützt werden, wenden Sie sich bitte an den Support von .

> **Note**
>
> Die PKCS #11-Bibliothek für Client-SDK 5 wird jetzt auf Windows-Plattformen unterstützt. Es kann die meisten Anwendungsfälle bewältigen, die CNG- und KSP-Anbieter als Ersatz verwenden können und sollten. KSP ist derzeit nur in Client-SDK 3 verfügbar.

PKCS #11-Bibliothek

PKCS #11 ist ein Standard für die Ausführung von kryptografischen Vorgängen auf Hardwaresicherheitsmodulen (HSMs). AWS CloudHSM bietet Implementierungen der PKCS #11-Bibliothek, die mit PKCS #11 Version 2.40 kompatibel sind.

Informationen zu Bootstrapping finden Sie unter Verbinden mit dem Cluster. Informationen zur Fehlerbehebung finden Sie unter Bekannte Probleme für den PKCS#11-Bibliothek.

Informationen zur Verwendung von Client-SDK 3 finden Sie unter Vorheriges Client-SDK (Client-SDK 3).

Themen

- Installieren Sie die Bibliothek Client-SDK 5 für PKCS #11
- PKCS #11-Bibliothek
- Unterstützte Schlüsseltypen
- Unterstützte Mechanismen
- Unterstützte API-Operationen
- Unterstützte Schlüsselattribute
- Codebeispiele für die PKCS #11-Bibliothek
- Migrieren Sie Ihre PKCS #11-Bibliothek von Client-SDK 3 zu Client-SDK 5

- Erweiterte Konfigurationen für PKCS #11

Installieren Sie die Bibliothek Client-SDK 5 für PKCS #11

Dieses Thema enthält Anweisungen zur Installation der neuesten Version der PKCS #11-Bibliothek aus der Client-SDK 5-Versionsserie. Weitere Informationen zum Client-SDK oder zur PKCS #11-Bibliothek finden Sie unter Verwenden des Client-SDK und der PKCS #11-Bibliothek.

Installation

Mit dem Client-SDK 5 müssen Sie keinen Client-Daemon installieren oder ausführen.

Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung `disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter Schlüsselsynchronisierung und Client-SDK-5-Configure-Tool.

Weitere Informationen über die PKCS #11-Bibliothek in Client-SDK 5 finden Sie unter PKCS #11-Bibliothek.

> **Note**
>
> Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung `disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter Schlüsselsynchronisierung und Client-SDK-5-Configure-Tool.

Installieren und Konfigurieren der PKCS #11-Bibliothek

1. Verwenden Sie die folgenden Befehle zum Herunterladen und Installieren der PKCS #11-Bibliothek.

   Amazon Linux 2

   Installieren Sie die PKCS #11-Bibliothek für Amazon Linux 2 auf der X86_64-Architektur:

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.x86_64.rpm
   ```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

Installieren Sie die PKCS #11-Bibliothek für Amazon Linux 2 auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

Amazon Linux 2023

Installieren Sie die PKCS #11 -Bibliothek für Amazon Linux 2023 auf der X86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-pkcs11-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.amzn2023.x86_64.rpm
```

Installieren Sie die PKCS #11 -Bibliothek für Amazon Linux 2023 auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-pkcs11-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

Installieren Sie die PKCS #11-Bibliothek für CentOS 7.8+ auf der X86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

Installieren Sie die PKCS #11 -Bibliothek für RHEL 7 auf der X86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

Installieren Sie die PKCS #11 -Bibliothek für RHEL 8 auf der X86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

Installieren Sie die PKCS #11 -Bibliothek für RHEL 9 auf der X86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-pkcs11-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el9.x86_64.rpm
```

Installieren Sie die PKCS #11 -Bibliothek für RHEL 9 auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-pkcs11-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

Installieren Sie die PKCS #11-Bibliothek für Ubuntu 20.04 LTS auf der X86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/
cloudhsm-pkcs11_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

Installieren Sie die PKCS #11-Bibliothek für Ubuntu 22.04 LTS auf der X86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-pkcs11_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u22.04_amd64.deb
```

Installieren Sie die PKCS #11 -Bibliothek für Ubuntu 22.04 LTS auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-pkcs11_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u22.04_arm64.deb
```

Windows Server 2016

Installieren Sie die PKCS #11-Bibliothek für Windows Server 2016 auf der X86_64-Architektur:

1. Laden Sie die PKCS #11-Bibliothek für Client-SDK 5 herunter.
2. Führen Sie das PKCS #11 -Bibliotheksinstallationsprogramm (AWSCloudHSMPKCS11-latest.msi) mit Windows-Administratorrechten aus.

Windows Server 2019

Installieren Sie die PKCS #11-Bibliothek für Windows Server 2019 auf der X86_64-Architektur:

1. Laden Sie die PKCS #11-Bibliothek für Client-SDK 5 herunter.

2. Führen Sie das PKCS #11 -Bibliotheksinstallationsprogramm (AWSCloudHSMPKCS11-latest.msi) mit Windows-Administratorrechten aus.

2. Verwenden Sie das Configure-Tool, um den Speicherort des ausstellenden Zertifikats anzugeben. Anweisungen finden Sie unter Geben Sie den Speicherort des ausstellenden Zertifikats an.
3. Um eine Verbindung zu Ihrem Cluster herzustellen, siehe Bootstrap für das Client-SDK.
4. Die PKCS #11-Bibliotheksdateien finden Sie an den folgenden Speicherorten:

   - Linux-Binärdateien, Konfigurationsskripten und Protokolldateien:

     ```
     /opt/cloudhsm
     ```

     Windows-Binärdateien:

     ```
     C:\ProgramFiles\Amazon\CloudHSM
     ```

     Windows-Konfigurationsskripten und Protokolldateien:

     ```
     C:\ProgramData\Amazon\CloudHSM
     ```

PKCS #11-Bibliothek

Wenn Sie die PKCS #11-Bibliothek verwenden, läuft Ihre Anwendung als ein bestimmter Crypto-Benutzer (CU) in Ihren HSMs. Ihre Anwendung kann nur die Schlüssel anzeigen und verwalten, die der CU besitzt und freigibt. Sie können einen vorhandenen CU in HSMs verwenden oder für die Anwendung einen neuen CU erstellen. Weitere Informationen zum Verwalten von CUs finden Sie unter HSM-Benutzer mit CloudHSM-CLI verwalten und HSM-Benutzer mit CloudHSM Management Utility (CMU) verwalten.

Um den CU für die PKCS #11-Bibliothek anzugeben, verwenden Sie den Pin-Parameter der PKCS #11-Funktion C_Login. Für AWS CloudHSM hat der Pin-Parameter das folgende Format:

```
<CU_user_name>:<password>
```

Mit dem folgenden Befehl wird beispielsweise dem PKCS #11-Bibliothek-Pin der CU mit dem Benutzernamen `CryptoUser` und dem Passwort `CUPassword123!` zugewiesen.

```
CryptoUser:CUPassword123!
```

Unterstützte Schlüsseltypen

Die PKCS #11-Bibliothek unterstützt die folgenden Schlüsseltypen.

Schlüsseltyp	Beschreibung
AES	Generieren der 128-, 192- und 256-Bit-AES-Schlüssel.
Dreifaches DES (3DES, DESede)	Generieren von 192-Bit-Triple-DES-Schlüsseln. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.
EC	Generieren Sie Schlüssel mit den Kurven secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384) und secp521r1 (P-521).
GENERIC_SECRET	Generieren Sie generische Geheimnisse mit 1 bis 800 Byte.
RSA	Generieren Sie RSA-Schlüssel mit 2048 bis 4096 Bit, in Schritten von 256 Bit.

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Unterstützte Mechanismen

Die PKCS #11-Bibliothek entspricht Version 2.40 der PKCS #11-Spezifikation. Um eine kryptographische Funktion mit PKCS#11 aufzurufen, rufen Sie eine Funktion mit einem bestimmten Mechanismus auf. In den folgenden Abschnitten werden die Kombinationen von Funktionen und Mechanismen zusammengefasst, die von AWS CloudHSM unterstützt werden.

Die PKCS #11-Bibliothek unterstützt die folgenden Algorithmen:

- Verschlüsselung und Entschlüsselung – AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP und RSA-PKCS
- Signieren und Verifizieren – RSA, HMAC und ECDSA; mit und ohne Hashing
- Hash/Digest – SHA1, SHA224, SHA256, SHA384 und SHA512
- Key Wrap – AES Key Wrap[1], AES-GCM, RSA-AES und RSA-OAEP

Themen

- Schlüssel- und Schlüsselpaarfunktionen generieren
- Funktionen zum Signieren und Überprüfen
- Funktionen zur Signierung, Wiederherstellung und Überprüfung der Wiederherstellung
- Digest-Funktionen
- Funktionen zum Verschlüsseln und Entschlüsseln
- Schlüsselfunktionen ableiten
- Funktionen „Wrap“ und „Unwrap“
- Maximale Datengröße für jeden Mechanismus
- Anmerkungen zum Mechanismus

Schlüssel- und Schlüsselpaarfunktionen generieren

Mit der AWS CloudHSM-Softwarebibliothek für die PKCS #11-Bibliothek können Sie die folgenden Mechanismen für die Funktionen Schlüssel generieren und Schlüsselpaar verwenden.

- `CKM_RSA_PKCS_KEY_PAIR_GEN`
- `CKM_RSA_X9_31_KEY_PAIR_GEN` – Dieser Mechanismus ist funktionell identisch mit dem `CKM_RSA_PKCS_KEY_PAIR_GEN`-Mechanismus, aber bietet stärkere Garantien für die Generierung von `p` und `q`.
- `CKM_EC_KEY_PAIR_GEN`
- `CKM_GENERIC_SECRET_KEY_GEN`
- `CKM_AES_KEY_GEN`
- `CKM_DES3_KEY_GEN` – bevorstehende Änderung in der Fußnote 5 aufgeführt.

Funktionen zum Signieren und Überprüfen

Mit der AWS CloudHSM-Softwarebibliothek für die PKCS #11-Bibliothek können Sie die folgenden Mechanismen für die Funktionen Signieren und Überprüfen verwenden. Mit Client-SDK 5 werden die Daten lokal in der Software gehasht. Das bedeutet, dass die Größe der Daten, die vom SDK gehasht werden können, unbegrenzt ist.

Mit Client-SDK 5 erfolgt das RSA- und ECDSA-Hashing lokal, sodass es kein Datenlimit gibt. Bei HMAC gibt es ein Datenlimit. Weitere Informationen finden Sie in der Fußnote 2.

RSA

- `CKM_RSA_X_509`
- `CKM_RSA_PKCS` – Nur Single-Part-Operationen.
- `CKM_RSA_PKCS_PSS` – Nur Single-Part-Operationen.
- `CKM_SHA1_RSA_PKCS`
- `CKM_SHA224_RSA_PKCS`
- `CKM_SHA256_RSA_PKCS`
- `CKM_SHA384_RSA_PKCS`
- `CKM_SHA512_RSA_PKCS`
- `CKM_SHA512_RSA_PKCS`
- `CKM_SHA1_RSA_PKCS_PSS`
- `CKM_SHA224_RSA_PKCS_PSS`
- `CKM_SHA256_RSA_PKCS_PSS`
- `CKM_SHA384_RSA_PKCS_PSS`
- `CKM_SHA512_RSA_PKCS_PSS`

ECDSA

- `CKM_ECDSA` – Nur Single-Part-Operationen.
- `CKM_ECDSA_SHA1`
- `CKM_ECDSA_SHA224`
- `CKM_ECDSA_SHA256`
- `CKM_ECDSA_SHA384`

- CKM_ECDSA_SHA512

HMAC

- CKM_SHA_1_HMAC[2]
- CKM_SHA224_HMAC[2]
- CKM_SHA256_HMAC[2]
- CKM_SHA384_HMAC[2]
- CKM_SHA512_HMAC[2]

CMAC

- CKM_AES_CMAC

Funktionen zur Signierung, Wiederherstellung und Überprüfung der Wiederherstellung

Das Client-SDK 5 unterstützt die Funktionen Sign Recover und Verify Recover nicht.

Digest-Funktionen

Die AWS CloudHSM-Softwarebibliothek für die PKCS #11-Bibliothek ermöglicht es Ihnen, die folgenden Mechanismen für Digest-Funktionen zu verwenden. Mit Client-SDK 5 werden die Daten lokal in der Software gehasht. Das bedeutet, dass die Größe der Daten, die vom SDK gehasht werden können, unbegrenzt ist.

- CKM_SHA_1
- CKM_SHA224
- CKM_SHA256
- CKM_SHA384
- CKM_SHA512

Funktionen zum Verschlüsseln und Entschlüsseln

Die AWS CloudHSM-Softwarebibliothek für die PKCS #11-Bibliothek ermöglicht es Ihnen, die folgenden Mechanismen für Verschlüsselungs- und Entschlüsselungsfunktionen zu verwenden.

- CKM_RSA_X_509
- CKM_RSA_PKCS – Nur Single-Part-Operationen. Bevorstehende Änderung in der Fußnote 5 aufgeführt.
- CKM_RSA_PKCS_OAEP – Nur Single-Part-Operationen.
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_DES3_CBC – bevorstehende Änderung in der Fußnote 5 aufgeführt.
- CKM_DES3_ECB – bevorstehende Änderung in der Fußnote 5 aufgeführt.
- CKM_DES3_CBC_PAD – bevorstehende Änderung in der Fußnote 5 aufgeführt.
- CKM_AES_GCM [1, 2]
- CKM_CLOUDHSM_AES_GCM[3]

Schlüsselfunktionen ableiten

Die AWS CloudHSM-Softwarebibliothek für die PKCS #11-Bibliothek ermöglicht es Ihnen, die folgenden Mechanismen für Ableitungsfunktionen zu verwenden.

- CKM_SP800_108_COUNTER_KDF

Funktionen „Wrap“ und „Unwrap“

Die AWS CloudHSM-Softwarebibliothek für die PKCS #11-Bibliothek ermöglicht es Ihnen, die folgenden Mechanismen für die Funktionen Wrap und Unwrap zu verwenden.

Weitere Informationen zum AES-Schlüssel-Wrapping finden Sie unter AES Key Wrapping.

- CKM_RSA_PKCS – Nur Single-Part-Operationen. Bevorstehende Änderung in der Fußnote 5 aufgeführt.
- CKM_RSA_PKCS_OAEP[4]
- CKM_AES_GCM[1, 3]
- CKM_CLOUDHSM_AES_GCM[3]

- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD[3]
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD[3]
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD[3]

Maximale Datengröße für jeden Mechanismus

Die folgende Tabelle listet die maximale Datengröße für jeden Mechanismus auf:.

Maximale Datensatzgröße

Mechanismus	Maximale Datengröße in Bytes
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288
CKM_SHA384_HMAC	16224
CKM_SHA512_HMAC	16224
CKM_AES_CBC	16272
CKM_AES_GCM	16224
CKM_CLOUDHSM_AES_GCM	16224
CKM_DES3_CBC	16280

Anmerkungen zum Mechanismus

- [1] Bei AES-GCM-Verschlüsselungen akzeptiert HSM keine Initialisierungsvektor (IV)-Daten von der Anwendung. Sie müssen einen erzeugten IV verwenden. Das vom HSM bereitgestellte 12 Byte IV wird in den referenzierten Speicherbereich geschrieben, auf den das pIV-Element der von Ihnen bereitgestellten CK_GCM_PARAMS-Parameterstruktur zeigt. Um Missverständnissen

vorzubeugen: Das PKCS#11-SDK in Version 1.1.1 und höher erzwingt, dass pIV auf einen auf Null zurückgesetzten Puffer verweist, wenn die AES-GCM-Verschlüsselung initialisiert wird.

- [2] Wenn der Datenpuffer die maximale Datengröße überschreitet, führt die Operation bei der Bearbeitung von Daten mittels eines der folgenden Mechanismen zu einem Fehler. Für diese Mechanismen muss die gesamte Datenverarbeitung innerhalb des HSM erfolgen. Informationen zu maximalen Datengrößensätzen für jeden Mechanismus finden Sie unter Maximale Datengröße für jeden Mechanismus.
- [3] Anbieterdefinierter Mechanismus. Um die von anbieterdefinierten CloudHSM-Mechanismen zu verwenden, müssen PKCS #11 -Anwendungen während der Kompilierung `/opt/cloudhsm/include/pkcs11t.h` enthalten.

  `CKM_CLOUDHSM_AES_GCM`: Dieser proprietäre Mechanismus ist eine programmatisch sicherere Alternative zum Standard `CKM_AES_GCM`. Er stellt die vom HSM generierte IV dem Chiffretext voran, anstatt sie zurück in die `CK_GCM_PARAMS`- Struktur zu schreiben, die während der Chiffrierinitialisierung bereitgestellt wird. Sie können diesen Mechanismus mit `C_Encrypt`-, `C_WrapKey`-, `C_Decrypt`- und `C_UnwrapKey`-Funktionen verwenden. Bei Verwendung dieses Mechanismus muss die pIV-Variable in der `CK_GCM_PARAMS`-Struktur auf `NULL` gesetzt werden. Wenn Sie diesen Mechanismus mit `C_Decrypt` und `C_UnwrapKey` verwenden, wird erwartet, dass der IV dem Verschlüsselungstext vorangestellt wird, der entpackt werden soll.

  `CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD`: AES-Verschlüsselung mit PKCS #5 Padding.

  `CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD`: AES-Verschlüsselung mit Zero Padding.
- [4] Die folgenden `CK_MECHANISM_TYPE` und `CK_RSA_PKCS_MGF_TYPE` werden als `CK_RSA_PKCS_OAEP_PARAMS` für `CKM_RSA_PKCS_OAEP` unterstützt:
  - `CKM_SHA_1` mit `CKG_MGF1_SHA1`
  - `CKM_SHA224` mit `CKG_MGF1_SHA224`
  - `CKM_SHA256` mit `CKG_MGF1_SHA256`
  - `CKM_SHA384` mit `CKM_MGF1_SHA384`
  - `CKM_SHA512` mit `CKM_MGF1_SHA512`
- [5] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Unterstützte API-Operationen

Die PKCS #11-Bibliothek unterstützt die folgenden PKCS #11 API Operationen.

- `C_CloseAllSessions`
- `C_CloseSession`
- `C_CreateObject`
- `C_Decrypt`
- `C_DecryptFinal`
- `C_DecryptInit`
- `C_DecryptUpdate`
- `C_DeriveKey`
- `C_DestroyObject`
- `C_Digest`
- `C_DigestFinal`
- `C_DigestInit`
- `C_DigestUpdate`
- `C_Encrypt`
- `C_EncryptFinal`
- `C_EncryptInit`
- `C_EncryptUpdate`
- `C_Finalize`
- `C_FindObjects`
- `C_FindObjectsFinal`
- `C_FindObjectsInit`
- `C_GenerateKey`
- `C_GenerateKeyPair`
- `C_GenerateRandom`
- `C_GetAttributeValue`
- `C_GetFunctionList`

- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_OpenSession
- C_Sign
- C_SignFinal
- C_SignInit
- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyUpdate
- C_WrapKey

Unterstützte Schlüsselattribute

Bei einem Schlüsselobjekt kann es sich um einen öffentlichen, privaten oder geheimen Schlüssel handeln. Für ein Schlüsselobjekt genehmigte Aktionen werden durch Attribute angegeben. Attribute werden bei der Erstellung des Schlüsselobjekts bestimmt. Wenn Sie die PKCS #11-Bibliothek verwenden, weisen wir Standardwerte zu, wie sie im PKCS #11-Standard festgelegt sind.

AWS CloudHSM unterstützt nicht alle in der PKCS #11-Spezifikation genannten Attribute. Wir erfüllen die Spezifikation für alle von uns unterstützten Attribute. Diese Attribute sind in den entsprechenden Tabellen aufgeführt.

Kryptografische Funktionen wie `C_CreateObject`, `C_GenerateKey`, `C_GenerateKeyPair`, `C_UnwrapKey` und `C_DeriveKey` zum Erstellen, Ändern oder Kopieren von Objekten benötigen eine Attributvorlage für einen ihrer Parameter. Weitere Informationen zum Übertragen von Attributvorlagen während der Erstellung von Objekten finden Sie in den Beispielen unter Generieren von Schlüsseln über die PKCS #11-Bibliothek.

Interpretation der PKCS #11-Bibliotheksattribut-Tabelle

Die PKCS #11-Bibliothek-Tabelle enthält eine Liste von Attributen, die je nach Schlüsseltyp unterschiedlich sind. Sie gibt an, ob ein vorhandenes Attribut für einen bestimmten Schlüsseltyp bei Verwendung einer spezifischen kryptografischen Funktion mit AWS CloudHSM unterstützt wird.

Legende:

- ✓ gibt an, dass CloudHSM das Attribut für den spezifischen Schlüsseltyp unterstützt.
- ✖ gibt an, dass CloudHSM das Attribut für den spezifischen Schlüsseltyp nicht unterstützt.
- R gibt an, dass der Attributwert für den spezifischen Schlüsseltyp als schreibgeschützt festgelegt ist.
- S zeigt an, dass das Attribut von `GetAttributeValue` nicht gelesen werden kann, da hierbei Groß- und Kleinschreibung beachtet werden muss.
- Eine leere Zelle in der Spalte „Standardwert“ gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

GenerateKeyPair

Attribut	Schlüsseltyp					Default Value (Standard wert)
		EC privat	EC öffentlich	RSA privat	RSA öffentlich	
`CKA_CLASS`		✓	✓	✓	✓	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	True
CKA_TOKEN	✓	✓	✓	✓	Falsch
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✖	✓	✖	✓	False
CKA_DECRYPT	✓	✖	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓	✓	✓	✓	Wahr
CKA_SIGN	✓	✖	✓	✖	False

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_SIGN_RECOVER	✖	✖	✖	✖	
CKA_VERIFY	✖	✓	✖	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	
CKA_WRAP	✖	✓	✖	✓	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	
CKA_TRUSTED	✖	✓	✖	✓	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	False
CKA_UNWRAP	✓	✖	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	
CKA_SENSITIVE	✓[1]	✖	✓[1]	✖	True

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓[2]	
CKA_PRIME_1	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_PRIVATE_EXPONENT		✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT		✖	✖	✖	✓[2]	
CKA_EC_PARAMS		✖	✓[2]	✖	✖	
CKA_EC_POINT		✖	✖	✖	✖	
CKA_VALUE		✖	✖	✖	✖	
CKA_VALUE_LEN		✖	✖	✖	✖	
CKA_CHECK_VALUE		R	R	R	R	

GenerateKey

Attribut	Schlüsseltyp				Default Value (Standard wert)
		AES	DES3	Allgemeiner geheimer Schlüssel	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	Falsch
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓	✓	✓	Wahr
CKA_SIGN	✓	✓	✓	Wahr
CKA_SIGN_RECOVER	✖	✖	✖	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_VERIFY	✓	✓	✓	Wahr
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✓	✓	✖	
CKA_TRUSTED	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	False
CKA_UNWRAP	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✓	✖	
CKA_SENSITIVE	✓	✓	✓	Wahr
CKA_ALWAYS_SENSITIVE	✖	✖	✖	
CKA_EXTRACTABLE	✓	✓	✓	True

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

CreateObject

Attribut	Schlüsseltyp							Default Value (Standard wert)
	EC privat	EC öffentlich	RSA privat	RSA öffentlich	AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	

Attribut	Schlüsseltyp							Default Value (Standardwert)
CKA_LOCAL	R	R	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖	False
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	Wahr
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	✖	✖	False
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	False

Attribut	Schlüsseltyp							Default Value (Standardwert)
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	False
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	R	R	R	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓	True

Attribut	Schlüsseltyp							Default Value (Standardwert)
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	R	R	R	
CKA_MODULUS	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✓[2]	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✓[2]	✓[2]	✖	✖	✖	

Attribut	Schlüsseltyp							Default Value (Standard wert)
CKA_EC_PARAMS	✓[2]	✓[2]	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✓[2]	✖	✖	✖	✖	✖	
CKA_VALUE	✓[2]	✖	✖	✖	✓[2]	✓[2]	✓[2]	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

Attribut	Schlüsseltyp						Default Value (Standard wert)
	EC privat	RSA privat	AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	

Attribut	Schlüsseltyp					Default Value (Standard wert)
CKA_LABEL	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✖	✖	✓	✓	✖	False
CKA_DECRYPT	✖	✓	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓	✓	✓	✓	✓	Wahr
CKA_SIGN	✓	✓	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	False

Attribut	Schlüsseltyp					Default Value (Standardwert)
CKA_VERIFY	✖	✖	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	False
CKA_UNWRAP	✖	✓	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	✓	✓	Wahr
CKA_EXTRACTABLE	✓	✓	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	
CKA_MODULUS	✖	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	

Attribut	Schlüsseltyp					Default Value (Standard wert)
CKA_PRIME_2	✖	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	

DeriveKey

Attribut	Schlüsseltyp			Default Value (Standard wert)
	AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	Falsch
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	Wahr

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_SIGN	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	R	R	R	True
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

Attribut	Schlüsseltyp						
	EC privat	EC öffentlich	RSA privat	RSA öffentlich	AES	DES3	Allgemeiner geheimer Schlüssel
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]

Attribut	Schlüsseltyp						
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓

Attribut	Schlüsseltyp						
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_NEVER_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R
CKA_MODULUS	✖	✖	✓	✓	✖	✖	✖
CKA_MODULUS_BITS	✖	✖	✖	✓	✖	✖	✖
CKA_PRIME_1	✖	✖	S	✖	✖	✖	✖
CKA_PRIME_2	✖	✖	S	✖	✖	✖	✖
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖

Attribut	Schlüsseltyp						
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓	✓	✓
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

Anmerkungen zu Attributen

- [1] Dieses Attribut wird teilweise von der Firmware unterstützt und muss explizit nur auf den Standardwert festgelegt werden.
- [2] Obligatorisches Attribut.

Ändern von Attributen

Einige Attribute eines Objekts können geändert werden, nachdem das Objekt erstellt wurde, andere dagegen nicht. Um Attribute zu ändern, verwenden Sie den Befehl setAttribute aus cloudhsm_mgmt_util. Sie können auch eine Liste der Attribute und der Konstanten, die sie repräsentieren, ableiten, indem Sie den Befehl listAttribute aus cloudhsm_mgmt_util verwenden.

Die folgende Liste zeigt Attribute, die nach der Erstellung eines Objekts geändert werden dürfen.

- `CKA_LABEL`
- `CKA_TOKEN`

> **Note**
>
> Änderungen sind nur zum Umwandeln eines Sitzungsschlüssels in einen Token-Schlüssel zulässig. Verwenden Sie den Befehl setAttribute aus key_mgmt_util, um den Wert des Attributs zu ändern.

- `CKA_ENCRYPT`
- `CKA_DECRYPT`
- `CKA_SIGN`
- `CKA_VERIFY`
- `CKA_WRAP`
- `CKA_UNWRAP`
- `CKA_LABEL`
- `CKA_SENSITIVE`
- `CKA_DERIVE`

> **Note**
>
> Dieses Attribut unterstützt die Schlüsselableitung. Es muss für alle öffentlichen Schlüssel `False` lauten und kann nicht auf `True` festgelegt werden. Für geheime und private EC-Schlüssel kann es auf `True` oder `False` einstellt werden.

- `CKA_TRUSTED`

> **Note**
>
> Dieses Attribut kann nur vom Verschlüsselungsverantwortlichen (CO) auf `True` oder `False` festgelegt werden.

- `CKA_WRAP_WITH_TRUSTED`

> **Note**
>
> Wenden Sie dieses Attribut auf einen exportierbaren Datenschlüssel an, um anzugeben, dass Sie diesen Schlüssel nur mit Schlüsseln umschließen können, die als `CKA_TRUSTED` markiert sind. Sobald Sie `CKA_WRAP_WITH_TRUSTED` auf wahr setzen, wird das Attribut schreibgeschützt und Sie können das Attribut nicht mehr ändern oder entfernen.

Interpretieren von Fehlercodes

Das Angeben eines nicht von einem spezifischen Schlüssel unterstützten Attributs in der Vorlage führt zu einem Fehler. Die folgende Tabelle enthält die Fehlercodes, die generiert werden, wenn Sie gegen die Spezifikationen verstoßen:

Fehlercode	Beschreibung
`CKR_TEMPLATE_INCONSISTENT`	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage ein Attribut angeben, das zwar der PKCS #11-Spezifikation entspricht, jedoch nicht von CloudHSM unterstützt wird.
`CKR_ATTRIBUTE_TYPE_INVALID`	Diese Fehlermeldung erhalten Sie, wenn Sie den Wert für ein Attribut abrufen, das zwar der PKCS #11-Spezifikation entspricht, jedoch nicht von CloudHSM unterstützt wird.
`CKR_ATTRIBUTE_INCOMPLETE`	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage das obligatorische Attribut nicht angeben.

Fehlercode	Beschreibung
`CKR_ATTRIBUTE_READ_ONLY`	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage ein schreibgeschütztes Attribut angeben.

Codebeispiele für die PKCS #11-Bibliothek

Die Codebeispiele auf GitHub zeigen Ihnen, wie Sie grundlegende Aufgaben mit der PKCS #11-Bibliothek ausführen.

Voraussetzungen

Bevor Sie die Beispiele ausführen, führen Sie die folgenden Schritte aus, um Ihre Umgebung einzurichten:

- Installieren und konfigurieren Sie die PKCS #11-Bibliothek für das Client-SDK 5.
- Richten Sie einen Crypto-Benutzer (CU) ein. Ihre Anwendung verwendet dieses HSM-Konto, um die Codebeispiele auf dem HSM auszuführen.

Codebeispiele

Codebeispiele für die AWS CloudHSM Softwarebibliothek für PKCS#11 sind auf verfügbarGitHub. Dieses Repository enthält Beispiele für allgemeine Vorgänge mit PKCS #11, einschließlich Verschlüsselung, Entschlüsselung, Signieren und Verifizieren.

- Schlüssel generieren (AES, RSA, EC)
- Schlüsselattribute auflisten
- Verschlüsseln und Entschlüsseln von Daten mit AES GCM
- Verschlüsseln und Entschlüsseln von Daten mit AES_CTR
- Verschlüsseln und Entschlüsseln von Daten mit 3DES
- Signieren und Verifizieren von Daten mit RSA
- Ableiten von Schlüsseln mit HMAC KDF
- Verpacken und Entpacken von Schlüsseln mit AES und PKCS #5 Padding
- Verpacken und Entpacken von Schlüsseln mit AES ohne Padding

- Verpacken und Entpacken von Schlüsseln mit AES und Zero Padding
- Packen und Entpacken von Schlüsseln mit AES-GCM
- Schlüssel mit RSA ver- und entpacken

Migrieren Sie Ihre PKCS #11-Bibliothek von Client-SDK 3 zu Client-SDK 5

Verwenden Sie dieses Thema, um Ihre PKCS #11-Bibliothek von Client-SDK 3 zu Client-SDK 5 zu migrieren. Vorteile der Migration finden Sie unter Vorteile von Client-SDK 5.

In führen AWS CloudHSMKundenanwendungen kryptografische Operationen mit dem AWS CloudHSM Client Software Development Kit (SDK) aus. Client-SDK 5 ist das primäre SDK, dem weiterhin neue Funktionen und Plattformunterstützung hinzugefügt werden.

Informationen zur Überprüfung der Migrationsanweisungen für alle Anbieter finden Sie unter Migration von Client-SDK 3 zu Client-SDK 5.

Vorbereiten, indem Sie grundlegende Änderungen angehen

Überprüfen Sie diese grundlegenden Änderungen und aktualisieren Sie Ihre Anwendung in Ihrer Entwicklungsumgebung entsprechend.

Wrap-Mechanismen haben sich geändert

Client-SDK-3-Mechanismus	Gleichwertiger Client-SDK-5-Mechanismus
`CKM_AES_KEY_WRAP`	`CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD`
`CKM_AES_KEY_WRAP_PAD`	`CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD`
`CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD`	`CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD`
`CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD`	`CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD`
`CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD`	`CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD`

ECDH

In Client-SDK 3 können Sie ECDH verwenden und eine KDF angeben. Diese Funktionalität ist derzeit in Client-SDK 5 nicht verfügbar. Wenn Ihre Anwendung diese Funktionalität benötigt, wenden Sie sich bitte an den Support von .

Schlüssel-Handles sind jetzt sitzungsspezifisch

Um Schlüsselnamen in Client-SDK 5 erfolgreich verwenden zu können, müssen Sie bei jeder Ausführung einer Anwendung die Schlüsselnamen abrufen. Wenn Sie bereits Anwendungen haben, die dieselben Schlüssel-Handles für verschiedene Sitzungen verwenden, müssen Sie Ihren Code so ändern, dass er bei jeder Ausführung der Anwendung den Schlüssel-Handle erhält. Informationen zum Abrufen von Schlüsselhandles finden Sie in diesem AWS CloudHSM PKCS #11-Beispiel. Diese Änderung entspricht der PKCS #11 2.40-Spezifikation.

Migrieren zu Client-SDK 5

Folgen Sie den Anweisungen in diesem Abschnitt, um von Client-SDK 3 zu Client-SDK 5 zu migrieren.

> **Note**
>
> Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 und RHEL 6 werden derzeit mit Client-SDK 5 nicht unterstützt. Wenn Sie derzeit eine dieser Plattformen mit Client-SDK 3 verwenden, müssen Sie bei der Migration zu Client-SDK 5 eine andere Plattform wählen.

1. Deinstallieren Sie die PKCS #11-Bibliothek für Client-SDK 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-pkcs11
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-pkcs11
   ```

RHEL 7

```
$ sudo yum remove cloudhsm-pkcs11
```

RHEL 8

```
$ sudo yum remove cloudhsm-pkcs11
```

2. Deinstallieren Sie den Client-Daemon für Client-SDK 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-client
   ```

   > **Note**
   >
   > Benutzerdefinierte Konfigurationen müssen erneut aktiviert werden.

3. Installieren Sie die PKCS #11-Bibliothek des Client-SDK, indem Sie die Schritte unter befolgenInstallieren Sie die Bibliothek Client-SDK 5 für PKCS #11.
4. Client-SDK 5 führt ein neues Konfigurationsdateiformat und ein Befehlszeilen-Bootstrapping-Tool ein. Befolgen Sie zum Bootstrappen Ihrer PKCS #11-Bibliothek für Client-SDK 5 die Anweisungen im Benutzerhandbuch unter Bootstrap für das Client-SDK.

5. Testen Sie Ihre Anwendung in Ihrer Entwicklungsumgebung. Nehmen Sie Aktualisierungen an Ihrem vorhandenen Code vor, um Ihre grundlegenden Änderungen vor der letzten Migration zu beheben.

Verwandte Themen

- Bewährte Methoden für AWS CloudHSM

Erweiterte Konfigurationen für PKCS #11

Der AWS CloudHSM PKCS #11-Anbieter umfasst die folgende erweiterte Konfiguration, die nicht Teil der allgemeinen Konfigurationen ist, die die meisten Kunden verwenden. Diese Konfigurationen bieten zusätzliche Funktionen.

- Verbindung zu mehreren Steckplätzen mit PKCS #11
- Versuchen Sie erneut, die Konfiguration für PKCS #11 zu konfigurieren

Verbindung zu mehreren Steckplätzen mit PKCS #11

Ein einzelner Steckplatz in der PKCS #11-Bibliothek des Client-SDK 5 steht für eine einzelne Verbindung zu einem Cluster in AWS CloudHSM. Mit Client-SDK 5 können Sie Ihre PKCS11-Bibliothek so konfigurieren, dass mehrere Steckplätze Benutzer über eine einzige PKCS #11-Anwendung mit mehreren CloudHSM-Clustern verbinden können.

Verwenden Sie die Anweisungen in diesem Thema, um dafür zu sorgen, dass Ihre Anwendung Multi-Slot-Funktionen verwendet, um eine Verbindung mit mehreren Clustern herzustellen.

Themen

- Voraussetzungen für mehrere Steckplätze
- Konfigurieren Sie die PKCS #11-Bibliothek für die Multi-Steckplatz-Funktionalität
- configure-pkcs11 add-cluster
- configure-pkcs11 remove-cluster

Voraussetzungen für mehrere Steckplätze

- Zwei oder mehr AWS CloudHSM Cluster, zu denen Sie eine Verbindung herstellen möchten, zusammen mit ihren Cluster-Zertifikaten.
- Eine EC2-Instance mit Sicherheitsgruppen, die korrekt konfiguriert sind, um eine Verbindung zu allen oben genannten Clustern herzustellen. Weitere Informationen zum Einrichten eines Clusters und der Client-Instanz finden Sie unter Erste Schritte mit AWS CloudHSM.
- Um die Multi-Steckplatz-Funktionalität einzurichten, müssen Sie die PKCS #11-Bibliothek bereits heruntergeladen und installiert haben. Wenn Sie dies noch nicht getan haben, lesen Sie die Anweisungen unter ???.

Konfigurieren Sie die PKCS #11-Bibliothek für die Multi-Steckplatz-Funktionalität

Gehen Sie wie folgt vor, um Ihre PKCS #11-Bibliothek für die Multi-Steckplatz-Funktionalität zu konfigurieren:

1. Identifizieren Sie die Cluster, zu denen Sie mithilfe der Multi-Steckplatz-Funktionalität eine Verbindung herstellen möchten.
2. Fügen Sie diese Cluster zu Ihrer PKCS #11-Konfiguration hinzu, indem Sie den Anweisungen unter ??? folgen.
3. Wenn Ihre PKCS #11-Anwendung das nächste Mal ausgeführt wird, wird sie über Multi-Steckplatz-Funktionen verfügen.

configure-pkcs11 add-cluster

Wenn Sie mit PKCS #11 eine Verbindung zu mehreren Steckplätzen herstellen, verwenden Sie den configure-pkcs11 add-cluster-Befehl, um Ihrer Konfiguration einen Cluster hinzuzufügen.

Syntax

```
configure-pkcs11 add-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [--region <REGION>]
        [--endpoint <ENDPOINT>]
        [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
        [--server-client-cert-file <CLIENT CERTIFICATE FILE>]
        [--server-client-key-file <CLIENT KEY FILE>]
        [-h, --help]
```

Beispiele

Fügen Sie mithilfe des **`cluster-id`**-Parameters einen Cluster hinzu

Example

Verwenden Sie den configure-pkcs11 add-cluster zusammen mit dem `cluster-id`-Parameter, um Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) hinzuzufügen.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-pkcs11.exe add-cluster --cluster-
id cluster-1234567
```

> **Tip**
>
> Wenn die Verwendung von configure-pkcs11 add-cluster mit dem `cluster-id`-Parameter nicht dazu führt, dass der Cluster hinzugefügt wird, finden Sie im folgenden Beispiel eine längere Version dieses Befehls, die auch die Parameter `--region` und `--endpoint` zur Identifizierung des hinzugefügten Clusters erfordert. Wenn zum Beispiel die Region des Clusters eine andere ist als die, die als Standard für Ihre AWS CLI konfiguriert ist, sollten Sie den `--region`-Parameter verwenden, um die richtige Region zu verwenden. Darüber hinaus haben Sie die Möglichkeit, den AWS CloudHSM API-Endpunkt anzugeben, der für den Anruf verwendet werden soll. Dies kann für verschiedene Netzwerkkonfigurationen erforderlich sein, z. B. für die Verwendung von VPC-Schnittstellenendpunkten, für die nicht den Standard-DNS-Hostnamen verwendet wird. AWS CloudHSM

Fügen Sie einen Cluster mit den Parametern **`cluster-id`**, **`endpoint`**, und **`region`** hinzu

Example

Verwenden Sie configure-pkcs11 add-cluster zusammen mit den Parametern `cluster-id`, `endpoint` und `region`, um Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) hinzuzufügen.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-pkcs11.exe add-cluster --
cluster-id cluster-1234567--region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

Weitere Hinweise zu den Parametern `--cluster-id`, `--region` und `--endpoint` finden Sie unter the section called “Parameter”.

Parameter

--cluster-id ***<Cluster ID>***

Führt einen `DescribeClusters`-Aufruf aus, um alle IP-Adressen der HSM-Elastic-Network-Schnittstelle (ENI) im Cluster mit der Cluster-ID zu finden. Das System fügt die ENI-IP-Adressen zu den Konfigurationsdateien hinzu. AWS CloudHSM

> **Note**
>
> Wenn Sie den `--cluster-id` Parameter von einer EC2-Instance innerhalb einer VPC verwenden, die keinen Zugriff auf das öffentliche Internet hat, müssen Sie einen VPC-Schnittstellen-Endpunkt erstellen, mit dem Sie eine Verbindung herstellen können. AWS CloudHSM Weitere Informationen über VPC-Endpunkte finden Sie unter ???.

Erforderlich: Ja

--endpoint ***<Endpoint>***

Geben Sie den AWS CloudHSM API-Endpunkt an, der für den Aufruf verwendet wird. `DescribeClusters` Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Erforderlich: Nein

-- hsm-ca-cert <HsmCA Certificate Filepath>

Gibt den Dateipfad zum HSM-CA-Zertifikat an.

Erforderlich: Nein

--region ***<Region>***

Geben Sie die Region Ihres Clusters an. Sie müssen diese Option in Kombination mit `--cluster-id` festlegen.

Wenn Sie den `--region`-Parameter nicht angeben, wählt das System die Region aus, indem es versucht, die Umgebungsvariablen `AWS_DEFAULT_REGION` oder `AWS_REGION` zu lesen. Wenn diese Variablen nicht festgelegt sind, überprüft das System die Region, die Ihrem Profil in Ihrer AWS-Config-Datei zugeordnet ist (normalerweise `~/.aws/config`), sofern Sie in der `AWS_CONFIG_FILE`-Umgebungsvariable keine andere Datei angegeben haben. Wenn keine der oben genannten Optionen festgelegt ist, verwendet das System standardmäßig die `us-east-1`-Region.

Erforderlich: Nein

-- server-client-cert-file <Client Certificate Filepath>

Pfad zum Client-Zertifikat, das für die gegenseitige TLS-Client-Server-Authentifizierung verwendet wird.

Verwenden Sie diese Option nur, wenn Sie nicht den Standardschlüssel und das SSL/TLS-Zertifikat verwenden möchten, die im Client-SDK 5 enthalten sind. Sie müssen diese Option in Kombination mit `--server-client-key-file` festlegen.

Erforderlich: Nein

-- server-client-key-file <Client Key Filepath>

Pfad zum Client-Schlüssel, der für die gegenseitige TLS-Client-Server-Authentifizierung verwendet wird.

Verwenden Sie diese Option nur, wenn Sie nicht den Standardschlüssel und das SSL/TLS-Zertifikat verwenden möchten, die im Client-SDK 5 enthalten sind. Sie müssen diese Option in Kombination mit `--server-client-cert-file` festlegen.

Erforderlich: Nein

configure-pkcs11 remove-cluster

Wenn Sie mit PKCS #11 eine Verbindung zu mehreren Steckplätzen herstellen, verwenden Sie den configure-pkcs11 remove-cluster-Befehl, um einen Cluster aus den verfügbaren PKCS #11-Steckplätzen zu entfernen.

Syntax

```
configure-pkcs11 remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

Beispiele

Entfernen Sie mithilfe des **`cluster-id`**-Parameters einen Cluster

Example

Verwenden Sie den configure-pkcs11 remove-cluster zusammen mit dem `cluster-id`-Parameter, um aus Ihrer Konfiguration einen Cluster (mit der ID von `cluster-1234567`) zu entfernen.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 remove-cluster --cluster-
id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-pkcs11.exe remove-cluster --cluster-
id cluster-1234567
```

Weitere Informationen zum Parameter `--cluster-id` erhalten Sie unter the section called “Parameter”.

Parameter

--cluster-id ***<Cluster ID>***

Die ID des Clusters, der aus der Konfiguration entfernt werden soll

Erforderlich: Ja

Wiederholungsbefehle für PKCS #11

Das Client-SDK 5.8.0 und höher verfügen über eine integrierte automatische Wiederholungsstrategie, mit der HSM-gedrosselte Operationen von der Clientseite aus wiederholt werden. Wenn ein HSM Operationen drosselt, weil es zu sehr mit der Ausführung früherer Operationen beschäftigt ist und keine weiteren Anfragen annehmen kann, versuchen Client-SDKs, gedrosselte Operationen bis zu dreimal zu wiederholen, während es exponentiell abbricht. Diese automatische Wiederholungsstrategie kann auf einen von zwei Modi eingestellt werden: aus und Standard.

- aus: Das Client-SDK führt bei gedrosselten Vorgängen durch das HSM keine Wiederholungsstrategie durch.
- Standard: Dies ist der Standardmodus für das Client-SDK 5.8.0 und höher. In diesem Modus wiederholen die Client-SDKs automatisch gedrosselte Operationen, indem sie sich exponentiell zurückziehen.

Weitere Informationen finden Sie unter HSM-Drosselung.

Stellen Sie den Modus für Wiederholungsbefehle auf Aus

Linux

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Linux

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --default-retry-mode off
```

Windows

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Windows

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-pkcs11.exe --default-retry-mode off
```

OpenSSL Dynamic Engine

Mit der AWS CloudHSM OpenSSL Dynamic Engine können Sie kryptografische Operationen über die OpenSSL-API auf Ihren CloudHSM-Cluster auslagern.

AWS CloudHSM stellt eine OpenSSL Dynamic Engine bereit, über die Sie in SSL/TLS-Auslagerung unter Linux nachlesen können. Ein Beispiel zur Verwendung AWS CloudHSM mit OpenSSL finden Sie in diesem AWS-Sicherheitsblog. Informationen zur Plattformunterstützung für SDKs finden Sie unter the section called “Unterstützte Plattformen”. Informationen zur Fehlerbehebung finden Sie unterBekannte Probleme für die OpenSSL Dynamic Engine.

Informationen zur Verwendung von Client-SDK 3 finden Sie unter Vorheriges Client-SDK (Client-SDK 3).

Weitere Informationen finden Sie in den folgenden Themen.

Themen

- Installieren von OpenSSL Dynamic Engine
- OpenSSL Dynamic Engine-Schlüsseltypen
- Mechanismen der OpenSSL Dynamic Engine
- Migrieren Sie Ihre OpenSSL Dynamic Engine von Client SDK 3 auf Client SDK 5
- Erweiterte Konfigurationen für OpenSSL

Installieren von OpenSSL Dynamic Engine

> **Note**
>
> Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung `disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter Schlüsselsynchronisierung und Client-SDK-5-Configure-Tool.

Zum Installieren und Konfigurieren von OpenSSL Dynamic Engine

1. Verwenden Sie die folgenden Befehle zum Herunterladen und Installieren der OpenSSL-Engine.

   Amazon Linux 2

   Installieren Sie die OpenSSL Dynamic Engine für Amazon Linux 2 auf einer x86_64-Architektur:

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-dyn-latest.el7.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-dyn-latest.el7.x86_64.rpm
   ```

   Installieren Sie die OpenSSL Dynamic Engine für Amazon Linux 2 auf der ARM64-Architektur:

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-dyn-latest.el7.aarch64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-dyn-latest.el7.aarch64.rpm
   ```

   Amazon Linux 2023

   Installieren Sie die OpenSSL Dynamic Engine für Amazon Linux 2023 auf einer x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-dyn-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.amzn2023.x86_64.rpm
```

Installieren Sie die OpenSSL Dynamic Engine für Amazon Linux 2023 auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-dyn-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

Installieren Sie die OpenSSL Dynamic Engine für CentOS 7 auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

Installieren Sie die OpenSSL Dynamic Engine für RHEL 7 auf einer x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

Installieren Sie die OpenSSL Dynamic Engine für RHEL 8 auf einer x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-dyn-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

Installieren Sie die OpenSSL Dynamic Engine für RHEL 9 auf einer x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-dyn-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el9.x86_64.rpm
```

Installieren Sie die OpenSSL Dynamic Engine für RHEL 9 auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-dyn-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

Installieren Sie die OpenSSL Dynamic Engine für Ubuntu 20.04 LTS auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-dyn_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

Installieren Sie die OpenSSL Dynamic Engine für Ubuntu 22.04 LTS auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-dyn_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u22.04_amd64.deb
```

Installieren Sie die OpenSSL Dynamic Engine für Ubuntu 22.04 LTS auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-dyn_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u22.04_arm64.deb
```

Sie haben die gemeinsam genutzte Bibliothek für die Dynamic Engine unter `/opt/cloudhsm/lib/libcloudhsm_openssl_engine.so` installiert.

2. Bootstrap für Client-SDK 5 Weitere Informationen zu Bootstrapping finden Sie unter Bootstrap für das Client-SDK.
3. Legen Sie eine Umgebungsvariable mit den Anmeldeinformationen eines Crypto-Benutzers (CU) fest. Weitere Informationen zum Erstellen von CUs finden Sie unter Verwenden von CMU zur Verwaltung von Benutzern.

```
$ export CLOUDHSM_PIN=<HSM user name>:<password>
```

> **Note**
>
> Das Client-SDK 5 führt die `CLOUDHSM_PIN`-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein. Im Client-SDK 3 speichern Sie die CU-Anmeldeinformationen in der `n3fips_password`-Umgebungsvariable. Client-SDK 5 unterstützt beide Umgebungsvariablen, wir empfehlen jedoch, `CLOUDHSM_PIN` zu verwenden.

4. Verbinden Sie Ihre Installation von OpenSSL Dynamic Engine mit dem Cluster. Weitere Informationen finden Sie unter Verbinden mit dem Cluster.

5. Bootstrap für das Client-SDK 5. Weitere Informationen finden Sie unter the section called “Bootstrap für das Client-SDK”.

Überprüfen Sie die OpenSSL Dynamic Engine für Client-SDK 5

Verwenden Sie den folgenden Befehl, um Ihre Installation der OpenSSL Dynamic Engine zu überprüfen.

```
$ openssl engine -t cloudhsm
```

Die folgende Ausgabe verifiziert Ihre Konfiguration:

```
(cloudhsm) CloudHSM OpenSSL Engine
     [ available ]
```

OpenSSL Dynamic Engine-Schlüsseltypen

Die AWS CloudHSM OpenSSL Dynamic Engine unterstützt die folgenden Schlüsseltypen.

Schlüsseltyp	Beschreibung
EC	ECDSA signiert/verifiziert die Schlüsseltypen P-256, P-384 und secp256k1. Informationen zum Generieren von EC-Schlüsseln, die mit der OpenSSL-Engine kompatibel sind, finden Sie unter key generate-file.
RSA	Generierung von RSA-Schlüsseln für 2048-, 3072- und 4096-Bit-Schlüssel. RSA signieren /verifizieren. Die Überprüfung wird auf die OpenSSL-Software übertragen.

Mechanismen der OpenSSL Dynamic Engine

Erfahren Sie, wie Sie die Mechanismen der AWS CloudHSM OpenSSL Dynamic Engine verwenden.

Funktionen zum Signieren und Überprüfen

Die AWS CloudHSM OpenSSL Dynamic Engine ermöglicht es Ihnen, die folgenden Mechanismen für die Funktionen Signieren und Überprüfen zu verwenden.

Mit Client-SDK 5 werden die Daten lokal in der Software gehasht. Das bedeutet, dass die Größe der Daten, die gehasht werden können, unbegrenzt ist.

RSA-Signaturtypen

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

ECDSA-Signaturtypen

- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Migrieren Sie Ihre OpenSSL Dynamic Engine von Client SDK 3 auf Client SDK 5

Verwenden Sie dieses Thema, um Ihre OpenSSL Dynamic Engine von Client SDK 3 auf Client SDK 5 zu migrieren. Informationen zu den Vorteilen der Migration finden Sie unter. Vorteile von Client-SDK 5

In AWS CloudHSM führen Kundenanwendungen mithilfe des AWS CloudHSM Client Software Development Kit (SDK) kryptografische Operationen durch. Das Client SDK 5 ist das primäre SDK, das weiterhin um neue Funktionen und Plattformunterstützung erweitert wird.

> **Note**
>
> Die Generierung von Zufallszahlen wird derzeit in Client SDK 5 mit OpenSSL Dynamic Engine nicht unterstützt.

Die Migrationsanweisungen für alle Anbieter finden Sie unterMigration von Client-SDK 3 zu Client-SDK 5.

Migrieren Sie zum Client SDK 5

Folgen Sie den Anweisungen in diesem Abschnitt, um von Client SDK 3 auf Client SDK 5 zu migrieren.

> **Note**
>
> Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 und RHEL 6 werden derzeit nicht mit Client SDK 5 unterstützt. Wenn Sie derzeit eine dieser Plattformen mit Client SDK 3 verwenden, müssen Sie bei der Migration zu Client SDK 5 eine andere Plattform wählen.

1. Deinstallieren Sie die OpenSSL Dynamic Engine für Client SDK 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

RHEL 8

```
$ sudo yum remove cloudhsm-dyn
```

2. Deinstallieren Sie den Client Daemon für Client SDK 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-client
   ```

   > **Note**
   >
   > Benutzerdefinierte Konfigurationen müssen erneut aktiviert werden.

3. Installieren Sie das Client SDK OpenSSL Dynamic Engine, indem Sie die Schritte unter befolgen. Installieren von OpenSSL Dynamic Engine
4. Das Client SDK 5 führt ein neues Konfigurationsdateiformat und ein Befehlszeilen-Bootstrapping-Tool ein. Um Ihr Client SDK 5 OpenSSL Dynamic Engine zu booten, folgen Sie den Anweisungen im Benutzerhandbuch unter. Bootstrap für das Client-SDK
5. Testen Sie Ihre Anwendung in Ihrer Entwicklungsumgebung. Nehmen Sie vor der endgültigen Migration Aktualisierungen an Ihrem vorhandenen Code vor, um Ihre wichtigsten Änderungen zu beheben.

Verwandte Themen

- Bewährte Methoden für AWS CloudHSM

Erweiterte Konfigurationen für OpenSSL

Der AWS CloudHSM OpenSSL-Anbieter beinhaltet die folgende erweiterte Konfiguration, die nicht Teil der allgemeinen Konfigurationen ist, die die meisten Kunden verwenden. Diese Konfigurationen bieten zusätzliche Funktionen.

- Wiederholungsbefehle für OpenSSL

Wiederholungsbefehle für OpenSSL

Die Client-SDK-Versionen 5.8.0 und höher verfügen über eine integrierte automatische Wiederholungsstrategie, mit der HSM-gedrosselte Operationen von der Clientseite aus wiederholt werden. Wenn ein HSM Operationen drosselt, weil es zu sehr mit der Ausführung früherer Operationen beschäftigt ist und keine weiteren Anfragen annehmen kann, versuchen Client-SDKs, gedrosselte Operationen bis zu dreimal zu wiederholen, während es exponentiell abbricht. Diese automatische Wiederholungsstrategie kann auf einen von zwei Modi eingestellt werden: aus und Standard.

- aus: Das Client-SDK führt bei gedrosselten Vorgängen durch das HSM keine Wiederholungsstrategie durch.
- Standard: Dies ist der Standardmodus für das Client-SDK 5.8.0 und höher. In diesem Modus wiederholen die Client-SDKs automatisch gedrosselte Operationen, indem sie sich exponentiell zurückziehen.

Weitere Informationen finden Sie unter HSM-Drosselung.

Stellen Sie den Modus für Wiederholungsbefehle auf Aus

Linux

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Linux

- Sie können den folgenden Befehl verwenden, um Wiederholungsbefehle in den off-Modus zu setzen:

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

Windows

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Windows

- Sie können den folgenden Befehl verwenden, um Wiederholungsbefehle in den off-Modus zu setzen:

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-dyn.exe --default-retry-mode
 off
```

JCE-Anbieter

Der AWS CloudHSM-JCE-Anbieter ist eine Anbieter-Implementierung, die auf dem Java Cryptographic Extension (JCE)-Anbieter-Framework basiert. JCE ermöglicht es Ihnen, kryptografische Operationen mit dem Java Development Kits (JDK) durchzuführen. In diesem Handbuch wird der AWS CloudHSM-JCE-Anbieter manchmal als JCE-Anbieter bezeichnet. Verwenden Sie den JCE-Anbieter und das JDK, um kryptografische Operationen auf das HSM auszulagern. Informationen zur Fehlerbehebung finden Sie unter Bekannte Probleme für das JCE-SDK.

Informationen zur Verwendung von Client-SDK 3 finden Sie unter Vorheriges Client-SDK (Client-SDK 3).

Themen

- Installieren und verwenden Sie den AWS CloudHSM JCE-Anbieter für Client SDK 5
- Unterstützte Schlüsseltypen
- Unterstützte Mechanismen
- Unterstützte Java-Schlüsselattribute
- Codebeispiele für die AWS CloudHSM-Softwarebibliothek für Java
- AWS CloudHSM-JCE-Anbieter Javadocs
- AWS CloudHSM KeyStore Java-Klasse verwenden
- Migrieren Sie Ihren JCE-Anbieter von Client SDK 3 auf Client SDK 5

- Erweiterte Konfigurationen für JCE

Installieren und verwenden Sie den AWS CloudHSM JCE-Anbieter für Client SDK 5

Der JCE-Anbieter ist mit OpenJDK 8, OpenJDK 11, OpenJDK 17 und OpenJDK 21 kompatibel. Sie können beide von der OpenJDK-Website herunterladen.

> **Note**
>
> Um einen einzelnen HSM-Cluster mit Client-SDK 5 auszuführen, müssen Sie zunächst die Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten, indem Sie die Einstellung `disable_key_availability_check` auf `True` festlegen. Weitere Informationen finden Sie unter Schlüsselsynchronisierung und Client-SDK-5-Configure-Tool.

Themen

- Den JCE-Anbieter installieren
- Geben Sie Anmeldeinformationen für den JCE-Anbieter ein
- Grundlagen der Schlüsselverwaltung im JCE-Anbieter

Den JCE-Anbieter installieren

1. Verwenden Sie die folgenden Befehle, um den JCE-Anbieter herunterzuladen und zu installieren.

   Amazon Linux 2

   Installieren Sie den JCE-Anbieter für Amazon Linux 2 auf der x86_64-Architektur:

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-jce-latest.el7.x86_64.rpm
   ```

   Installieren Sie den JCE-Anbieter für Amazon Linux 2 auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.aarch64.rpm
```

Amazon Linux 2023

Installieren Sie den JCE-Anbieter für Amazon Linux 2023 auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-jce-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.amzn2023.x86_64.rpm
```

Installieren Sie den JCE-Anbieter für Amazon Linux 2023 auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-jce-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

Installieren Sie den JCE-Anbieter für CentOS 7 auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

Installieren Sie den JCE-Anbieter für RHEL 7 auf einer x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

Installieren Sie den JCE-Anbieter für RHEL 8 auf einer x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

Installieren Sie den JCE-Anbieter für RHEL 9 (9.2+) auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-jce-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el9.x86_64.rpm
```

Installieren Sie den JCE-Anbieter für RHEL 9 (9.2+) auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-jce-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

Installieren Sie den JCE-Anbieter für Ubuntu 20.04 LTS auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-jce_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

Installieren Sie den JCE-Anbieter für Ubuntu 22.04 LTS auf der x86_64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-jce_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u22.04_amd64.deb
```

Installieren Sie den JCE-Anbieter für Ubuntu 22.04 LTS auf der ARM64-Architektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-jce_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u22.04_arm64.deb
```

Windows Server 2016

Installieren Sie den JCE-Anbieter für Windows Server 2016 auf einer x86_64-Architektur, öffnen Sie ihn PowerShell als Administrator und führen Sie den folgenden Befehl aus:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/
Windows/AWSCloudHSMJCE-latest.msi -Outfile C:\AWSCloudHSMJCE-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMJCE-
latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

Installieren Sie den JCE-Anbieter für Windows Server 2019 auf der x86_64-Architektur, öffnen Sie ihn PowerShell als Administrator und führen Sie den folgenden Befehl aus:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/
Windows/AWSCloudHSMJCE-latest.msi -Outfile C:\AWSCloudHSMJCE-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMJCE-
latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

2. Bootstrap für Client-SDK 5 Weitere Informationen zu Bootstrapping finden Sie unter Bootstrap für das Client-SDK.
3. Suchen Sie die folgenden JCE-Anbieter-Dateien:

   Linux

   - `/opt/cloudhsm/java/cloudhsm-version.jar`
   - `/opt/cloudhsm/bin/configure-jce`
   - `/opt/cloudhsm/bin/jce-info`

   Windows

   - `C:\Program Files\Amazon\CloudHSM\java\cloudhsm-version.jar>`
   - `C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe`
   - `C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe`

Geben Sie Anmeldeinformationen für den JCE-Anbieter ein

Bevor Ihre Java-Anwendung ein HSM verwenden kann, muss das HSM die Anwendung zunächst authentifizieren. HSMs authentifizieren sich entweder mit einer expliziten oder einer impliziten Anmeldemethode.

Explizite Anmeldemethode – Mit dieser Methode können Sie AWS CloudHSM -Anmeldeinformationen direkt in der Anwendung bereitstellen. Es verwendet die Methode aus dem `AuthProvider`, bei der Sie einen CU-Benutzernamen und ein Passwort im Pin-Muster übergeben. Weitere Informationen finden Sie unter Codebeispiel für Anmeldung bei einem HSM.

Implizite Anmeldemethode – Mit dieser Methode können Sie AWS CloudHSM -Anmeldeinformationen entweder in einer neuen Property-Datei, über Systemeigenschaften oder als Umgebungsvariablen festlegen.

- Systemeigenschaften – Anmeldeinformationen über Systemeigenschaften fest, wenn Sie Ihre Anwendung ausführen. Die folgenden Beispiele zeigen zwei verschiedene Möglichkeiten:

  Linux

  ```
  $ java -DHSM_USER=<HSM user name> -DHSM_PASSWORD=<password>
  ```

```
System.setProperty("HSM_USER","<HSM user name>");
System.setProperty("HSM_PASSWORD","<password>");
```

Windows

```
PS C:\> java -DHSM_USER=<HSM user name> -DHSM_PASSWORD=<password>
```

```
System.setProperty("HSM_USER","<HSM user name>");
System.setProperty("HSM_PASSWORD","<password>");
```

- Umgebungsvariablen – Anmeldeinformationen als Umgebungsvariablen festlegen.

  Linux

```
$ export HSM_USER=<HSM user name>
$ export HSM_PASSWORD=<password>
```

  Windows

```
PS C:\> $Env:HSM_USER="<HSM user name>"
PS C:\> $Env:HSM_PASSWORD="<password>"
```

Anmeldeinformationen sind möglicherweise nicht verfügbar, wenn die Anwendung sie nicht bereitstellt oder wenn Sie eine Operation ausführen, bevor der HSM die Sitzung authentifiziert. In diesen Fällen sucht die CloudHSM-Softwarebibliothek für Java in der folgenden Reihenfolge nach den Anmeldeinformationen:

1. Systemeigenschaften
2. Umgebungsvariablen

Grundlagen der Schlüsselverwaltung im JCE-Anbieter

Die Grundlagen der Schlüsselverwaltung im JCE-Anbieter umfassen den Import von Schlüsseln, den Export von Schlüsseln, das Laden von Schlüsseln per Handle oder das Löschen von Schlüsseln. Weitere Informationen zur Schlüsselverwaltung finden Sie im Codebeispiel Schlüssel verwalten.

Weitere Codebeispiele für JCE-Anbieter finden Sie auch in Codebeispiele.

Unterstützte Schlüsseltypen

Mit der AWS CloudHSM-Softwarebibliothek für Java können Sie die folgenden Schlüsseltypen generieren.

Schlüsseltyp	Beschreibung
AES	Generieren der 128-, 192- und 256-Bit-AES-Schlüssel.
Dreifaches DES (3DES, DESede)	Generieren Sie einen 192-Bit-Triple-DES-Schlüssel. Eine bevorstehende Änderung finden Sie 1 in der Fußnote.
EC	Generieren Sie EC-Schlüsselpaare – die NIST-Kurven secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384) und secp521r1 (P-521).
GENERIC_SECRET	Generieren Sie generische Geheimnisse mit 1 bis 800 Byte.
HMAC	Hash-Unterstützung für SHA1, SHA224, SHA256, SHA384, SHA512.
RSA	Generieren Sie RSA-Schlüssel mit 2048 bis 4096 Bit, in Schritten von 256 Bit.

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Unterstützte Mechanismen

Weitere Informationen über die von AWS CloudHSM unterstützten Java Cryptography Architecture (JCA)-Schnittstellen und Engine-Klassen finden Sie in den folgenden Themen.

Themen

- Schlüssel- und Schlüsselpaarfunktionen generieren

Schlüssel- und Schlüsselpaarfunktionen generieren

Die AWS CloudHSM-Softwarebibliothek für Java ermöglicht es Ihnen, die folgenden Operationen zum Generieren von Schlüssel- und Schlüsselpaarfunktionen zu verwenden.

- `RSA`
- `EC`
- `AES`
- `DESede (Triple DES)`[siehe Hinweis 1]
- `GenericSecret`

Cipher-Funktionen

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden Kombinationen aus Algorithmen, Modi und Paddings.

Algorithmus	Mode	Padding	Hinweise

AES	CBC	`AES/CBC/NoPadding`
		
		`AES/CBC/PKCS5Padding`
			Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE` .
			
			Implementiert `Cipher.UNWRAP_MODE`

Algorithmus	Mode	Padding	Hinweise
			for AES/CBC NoPadding
AES	ECB	AES/ECB/PKCS5Padding AES/ECB/NoPadding	Implementiert Cipher.ENCRYPT_MODE und Cipher.DECRYPT_MODE .
AES	CTR	AES/CTR/NoPadding	Implementiert Cipher.ENCRYPT_MODE und Cipher.DECRYPT_MODE .

Algorithmus	Mode	Padding	Hinweise
AES	GCM	`AES/GCM/NoPadding`	Implementiert `Cipher.WRAP_MODE`, `Cipher.UNWRAP_MODE`, `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE`. HSM ignoriert den Initialisierungsvektor (IV) der Anforderung während der AES-GCM-Datenverschlüsselung und verwendet stattdessen einen selbst generierten IV. Nach Abschluss der Operation müssen Sie `Cipher.getIV()` abrufen, um den IV zu erhalten.

AESWrap	ECB	`AESWrap/ECB/NoPadding`
		
		`AESWrap/ECB/PKCS5Padding`
		
		`AESWrap/ECB/ZeroPadding`
			Implementiert `Cipher.WRAP_MODE` und `Cipher.UNWRAP_MODE`.

Algorithmus	Mode	Padding	Hinweise

DESede (Triple DES)	CBC	`DESede/CBC/PKCS5Padding`
		
		`DESede/CBC/NoPadding`
			Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE`. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

DESede (Triple DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE`. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

Algorithmus	Mode	Padding	Hinweise

RSA	ECB	`RSA/ECB/PKCS1Padding` siehe Hinweis 1
		
		`RSA/ECB/OAEPPadding`
		
		`RSA/ECB/OAEPWithSHA-1ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-224ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-256ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-384ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-512ANDMGF1Padding`
			Implementiert `Cipher.WRAP_MODE`, `Cipher.UNWRAP_MODE`, `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE`.

Algorithmus	Mode	Padding	Hinweise
RSA	ECB	`RSA/ECB/NoPadding`	Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE` .

RSAAESWrap	ECB	`RSAAESWrap/ECB/OAEPPadding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding`
			Implementiert `Cipher.WRAP_MODE` und `Cipher.UNWRAP_MODE` .

Funktionen zum Signieren und Überprüfen

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden Signatur- und Verifizierungstypen. Mit Client-SDK 5 und Signaturalgorithmen mit Hashing werden die Daten lokal in der Software gehasht, bevor sie zur Signatur/Überprüfung an das HSM gesendet werden. Das bedeutet, dass die Größe der Daten, die vom SDK gehasht werden können, unbegrenzt ist.

RSA-Signaturtypen

- `NONEwithRSA`
- `RSASSA-PSS`
- `SHA1withRSA`
- `SHA1withRSA/PSS`
- `SHA1withRSAandMGF1`
- `SHA224withRSA`
- `SHA224withRSAandMGF1`
- `SHA224withRSA/PSS`
- `SHA256withRSA`
- `SHA256withRSAandMGF1`
- `SHA256withRSA/PSS`
- `SHA384withRSA`
- `SHA384withRSAandMGF1`
- `SHA384withRSA/PSS`
- `SHA512withRSA`
- `SHA512withRSAandMGF1`
- `SHA512withRSA/PSS`

ECDSA-Signaturtypen

- `NONEwithECDSA`
- `SHA1withECDSA`
- `SHA224withECDSA`
- `SHA256withECDSA`

- SHA384withECDSA
- SHA512withECDSA

Digest-Funktionen

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden Mitteilungs-Digests. Mit Client-SDK 5 werden die Daten lokal in der Software gehasht. Das bedeutet, dass die Größe der Daten, die vom SDK gehasht werden können, unbegrenzt ist.

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

Funktionen des Hash-basierten Nachrichtenauthentifizierungscodes (HMAC)

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden HMAC-Algorithmen.

- HmacSHA1 (Maximale Datengröße in Byte: 16288)
- HmacSHA224 (Maximale Datengröße in Byte: 16256)
- HmacSHA256 (Maximale Datengröße in Byte: 16288)
- HmacSHA384 (Maximale Datengröße in Byte: 16224)
- HmacSHA512 (Maximale Datengröße in Byte: 16224)

Funktionen des verschlüsselten Nachrichtenauthentifizierungscodes (CMAC)

CMACs (Cipher-based Message Authentication Codes [verschlüsselter Nachrichtenauthentifizierungscode]) erstellen Nachrichtenauthentifizierungscodes (MACs) mithilfe einer Blockchiffre und eines geheimen Schlüssels. Sie unterscheiden sich von HMACs dadurch, dass sie für die MACs eher eine blocksymmetrische Schlüsselmethode als eine Hashing-Methode verwenden.

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden CMAC-Algorithmen.

- AESCMAC

Mithilfe von Schlüsselfabriken können Schlüssel in Schlüsselspezifikationen umgewandelt werden

Sie können Schlüsselfabriken verwenden, um Schlüssel in Schlüsselspezifikationen umzuwandeln. AWS CloudHSM hat zwei Arten von Schlüsselfabriken für JCE:

SecretKeyFactory: Wird verwendet, um symmetrische Schlüssel zu importieren oder abzuleiten. Mit können Sie einen unterstützten Schlüssel oder einen unterstützten übergeben SecretKeyFactory, KeySpec um symmetrische Schlüssel in zu importieren oder abzuleitenAWS CloudHSM. Im Folgenden sind die unterstützten Spezifikationen für aufgeführt KeyFactory:

- Für SecretKeyFactorydie -`generateSecret`Methode von werden die folgenden KeySpec Klassen unterstützt:
  - KeyAttributesMapkann verwendet werden, um Schlüsselbytes mit zusätzlichen Attributen als CloudHSM-Schlüssel zu importieren. Ein Beispiel finden Sie hier.
  - SecretKeySpeckann verwendet werden, um eine symmetrische Schlüsselspezifikation als CloudHSM-Schlüssel zu importieren.
  - AesCmacKdfParameterSpeckann verwendet werden, um symmetrische Schlüssel mit einem anderen CloudHSM-AES-Schlüssel abzuleiten.

> **Note**
>
> SecretKeyFactoryDie Methode von `translateKey` verwendet jeden Schlüssel, der die Schlüsselschnittstelle implementiert.

KeyFactory: Wird zum Importieren asymmetrischer Schlüssel verwendet. Mit können Sie einen KeyFactoryunterstützten Schlüssel übergeben oder unterstützte KeySpec Schlüssel in importierenAWS CloudHSM. Weitere Informationen finden Sie in folgenden verwandten Ressourcen:

- Für KeyFactorydie -`generatePublic`Methode von werden die folgenden KeySpec Klassen unterstützt:
- CloudHSM KeyAttributesMap sowohl für RSA als auch für EC KeyTypes, einschließlich:
  - CloudHSM KeyAttributesMap sowohl für RSA als auch für EC Public KeyTypes. Ein Beispiel finden Sie hier.
  - X509EncodedKeySpec sowohl für RSA als auch für EC Public Key

  - Öffentlicher Schlüssel von RSAPublicKeySpec für RSA
  - Öffentlicher EC-PublicKeySpecfür-EC-Schlüssel
- Für KeyFactorydie -`generatePrivate`Methode von werden die folgenden KeySpec Klassen unterstützt:
- CloudHSM KeyAttributesMap sowohl für RSA als auch für EC KeyTypes, einschließlich:
  - CloudHSM KeyAttributesMap sowohl für RSA als auch für EC Public KeyTypes. Ein Beispiel finden Sie hier.
  - PKCS8EncodedKeySpec für privaten EC- und RSA-Schlüssel
  - RSAPrivateCrtKeySpec für privaten RSA-Schlüssel
  - ECPrivateKeySpec für privaten EC-Schlüssel

Für KeyFactorydie -`translateKey`Methode nimmt sie jeden Schlüssel auf, der die Schlüsselschnittstelle implementiert.

Anmerkungen zum Mechanismus

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Unterstützte Java-Schlüsselattribute

Dieses Thema beschreibt, wie Sie eine proprietäre Erweiterung für den JCE-Anbieter verwenden, um Schlüsselattribute festzulegen. Verwenden Sie diese Erweiterung, um unterstützte Schlüsselattribute und ihre Werte während der folgenden Vorgänge festzulegen:

- Schlüsselgenerierung
- Schlüsselimport

Beispiele für die Verwendung von Schlüsselattributen finden Sie unter the section called “Codebeispiele”.

Themen

- Grundlegendes zu Attributen
- Unterstützte Attribute
- Festlegen von Attributen für einen Schlüssel

Grundlegendes zu Attributen

Mithilfe von Schlüsselattributen legen Sie fest, welche Aktionen für Schlüsselobjekte zulässig sind, einschließlich öffentlicher, privater oder geheimer Schlüssel. Schlüsselattribute und -werte definieren Sie bei der Erstellung von Schlüsselobjekten.

Die Java Cryptography Extension (JCE) gibt nicht an, wie Sie Werte für Schlüsselattribute festlegen sollten. Daher sind die meisten Aktionen standardmäßig zulässig. Im Gegensatz dazu definiert der PKCS# 11-Standard einen umfassenden Satz von Attributen mit restriktiveren Standardeinstellungen. Ab dem JCE-Anbieter 3.1 gibt es eine proprietäre Erweiterung von AWS CloudHSM, mit der Sie restriktivere Werte für häufig verwendete Attribute festlegen können.

Unterstützte Attribute

Sie können Werte für die Attribute festlegen, die in der folgenden Tabelle aufgeführt sind. Als bewährte Methode legen Sie nur Werte für Attribute fest, die Sie einschränken möchten. Wenn Sie keinen Wert angeben, verwendet AWS CloudHSM den in der folgenden Tabelle angegebenen Standardwert. Eine leere Zelle in der Spalte „Standardwert“ gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
DECRYPT	TRUE		TRUE	„True“ gibt an, dass Sie den Schlüssel zur Entschlüsselung eines beliebigen Puffers verwenden können. Sie legen dies im Allgemeinen auf „FALSE“ fest für einen Schlüssel

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
				, dessen WRAP auf wahr festgelegt ist.
DERIVE				Ermöglicht die Verwendung eines Schlüssels zur Ableitung anderer Schlüssel.
ENCRYPT	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel zur Verschlüsselung eines beliebigen Puffers verwenden können.
EXTRACTABLE	TRUE		TRUE	„True“ gibt an, dass Sie diesen Schlüssel aus dem HSM exportieren können.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
`ID`				Ein benutzerdefinierter Wert, der zur Identifizierung des Schlüssels verwendet wird.
`KEY_TYPE`				Wird verwendet, um den Schlüsseltyp zu identifizieren (AES, DeSede, generisches Geheimnis, EC oder RSA).

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
LABEL				Eine benutzerdefinierte Zeichenfolge, mit der Sie Schlüssel auf Ihrem HSM bequem identifizieren können. Um den bewährten Methoden zu folgen, verwenden Sie für jeden Schlüssel eine eindeutige Bezeichnung, damit er später leichter zu finden ist.
LOCAL				Weist auf einen vom HSM generierten Schlüssel hin.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
OBJECT_CLASS				Wird verwendet, um die Objektklasse eines Schlüssels (SecretKey, PublicKey oder) zu identifizieren PrivateKey.
PRIVATE	TRUE	TRUE	TRUE	„True“ gibt an, dass ein Benutzer erst auf den Schlüssel zugreifen darf, wenn der Benutzer authentifiziert ist. Zur Verdeutlichung: Benutzer können erst dann auf Schlüssel in AWS CloudHSM zugreifen, wenn sie authentifiziert sind, selbst wenn dieses Attribut auf „FALSCH“ gesetzt ist.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
`SIGN`	`TRUE`		`TRUE`	„True“ gibt an, dass Sie den Schlüssel verwenden können, um einen Hashwert zu signieren. Für öffentliche und private Schlüssel, die Sie archiviert haben, ist dies im Allgemeinen auf „FALSE“ festgelegt.
`SIZE`				Ein Attribut, das die Größe eines Schlüssels definiert. Weitere Informationen zu den unterstützten Schlüsselgrößen finden Sie unter Unterstützte Mechanismen für das Client-SDK 5.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
TOKEN	FALSE	FALSE	FALSE	Ein permanenter Schlüssel, der über alle HSMs im Cluster repliziert und in Backups enthalten ist. TOKEN = FALSCH impliziert einen flüchtigen Schlüssel, der automatisch gelöscht wird, wenn die Verbindung zum HSM unterbrochen oder abgemeldet wird.
UNWRAP	TRUE		TRUE	„True“ gibt an, dass Sie mit dem Schlüssel einen anderen Schlüssel entpacken (importieren) können.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
VERIFY	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel verwenden können, um eine Signatur zu überprüfen. Für private Schlüssel ist dies im Allgemeinen auf „FALSE“ festgelegt.
WRAP	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel zum Packen eines anderen Schlüssels verwenden können. Für private Schlüssel legen Sie dies in der Regel auf „FALSE“ fest.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
`WRAP_WITH_TRUSTED`	`FALSE`		`FALSE`	Wahr bedeutet, dass ein Schlüssel nur mit Schlüsseln, deren `TRUSTED`-Attribut auf wahr gesetzt ist, ein- und ausgepackt werden kann. Sobald bei einem Schlüssel `WRAP_WITH_TRUSTED` auf wahr gesetzt wurde, ist dieses Attribut schreibgeschützt und kann nicht auf Falsch gesetzt werden. Weitere Informationen zum Trust-Wrapping finden Sie unter Verwenden vertrauenswürdiger Schlüssel zur Steuerung

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
				von Schlüssel-Unwraps.

> **Note**
>
> Sie erhalten eine breitere Unterstützung für Attribute in der PKCS #11-Bibliothek. Weitere Informationen finden Sie unter Unterstützte PKCS #11-Attribute.

Festlegen von Attributen für einen Schlüssel

`KeyAttributesMap` ist ein Java-Map-ähnliches Objekt, mit dem Sie Attributwerte für Schlüsselobjekte festlegen können. Die Methoden für `KeyAttributesMap` funktionieren ähnlich den Methoden, die für die Java-Map-Manipulation verwendet werden.

Sie haben zwei Optionen, um benutzerdefinierte Werte für Attribute festzulegen:

- Verwenden Sie die in der folgenden Tabelle aufgeführten Methoden
- Verwenden Sie Builder-Muster, die später in diesem Dokument gezeigt werden

Attributzuordnungsobjekte unterstützen die folgenden Methoden zum Festlegen von Attributen:

Operation	Rückgabewert	**`KeyAttributesMap`** - Methode
Abrufen des Werts eines Schlüsselattributs für einen vorhandenen Schlüssel	Objekt (das den Wert enthält) oder null	get(keyAttribute)
Eingeben des Werts eines Schlüsselattributs	Der vorherige Wert, der dem Schlüsselattribut zugeordne	put(keyAttribute, Wert)

Operation	Rückgabewert	**`KeyAttributesMap`** - Methode
	t ist, oder null, wenn keine Zuordnung für ein Schlüssel attribut vorhanden ist	
Eingeben von Werten für mehrere Schlüsselattribute	N/A	putAll (keyAttributesMap)
Entfernen eines Schlüssel-Wert-Paares aus der Attributz uordnung	Der vorherige Wert, der dem Schlüsselattribut zugeordne t ist, oder null, wenn keine Zuordnung für ein Schlüssel attribut vorhanden ist	remove(keyAttribute)

> **Note**
>
> Alle Attribute, die Sie nicht explizit bestimmen, werden auf die Standardwerte festgelegt, die in der vorherigen Tabelle in the section called “Unterstützte Attribute” aufgeführt sind.

Festlegen von Attributen für ein Schlüsselpaar

Verwenden Sie die Java-Klasse `KeyPairAttributesMap`, um Schlüsselattribute für ein Schlüsselpaar zu verarbeiten. `KeyPairAttributesMap` fasst zwei `KeyAttributesMap`-Objekte zusammen, eines für einen öffentlichen Schlüssel und eines für einen privaten Schlüssel.

Um einzelne Attribute für den öffentlichen und den privaten Schlüssel separat festzulegen, können Sie die `put()`-Methode für das entsprechende `KeyAttributes`-Zuordnungsobjekt für diesen Schlüssel verwenden. Verwenden Sie die `getPublic()`-Methode, um die Attributzuordnung für den öffentlichen Schlüssel abzurufen, und die `getPrivate()`-Methode, um die Attributzuordnung für den privaten Schlüssel abzurufen. Geben Sie den Wert mehrerer Schlüsselattribute für öffentliche und private Schlüsselpaare zusammen ein, indem Sie die `putAll()`-Methode für eine Attributzuordnung von Schlüsselpaaren als Argument verwenden.

Codebeispiele für die AWS CloudHSM-Softwarebibliothek für Java

Voraussetzungen

Bevor Sie die Beispiele ausführen, müssen Sie die Umgebung einrichten:

- Installieren und konfigurieren Sie den Java Cryptographic Extension (JCE)-Anbieter.
- Legen Sie einen gültigen HSM-Benutzernahmen und ein Passwort fest. Crypto-Benutzer (CU)-Berechtigungen sind ausreichend für diese Aufgaben. Ihre Anwendung verwendet diese Anmeldeinformationen, um sich für die einzelnen Beispiele beim HSM anzumelden.
- Entscheiden Sie, wie Anmeldeinformationen für den JCE-Anbieter bereitgestellt werden sollen.

Codebeispiele

Die folgenden Codebeispiele zeigen Ihnen, wie Sie den AWS CloudHSMJCE-Anbieter verwenden, um grundlegende Aufgaben auszuführen. Weitere Beispiele finden Sie auf GitHub.

- Anmeldung bei einem HSM
- Verwalten von Schlüsseln
- Generieren von symmetrischen Schlüsseln
- Generieren von asymmetrischen Schlüsseln
- Verschlüsseln und Entschlüsseln mit AES-GCM
- Verschlüsseln und Entschlüsseln mit AES-CTR
- Mit deSede-ECB verschlüsseln und entschlüsseln siehe Hinweis 1
- Signieren und Verifizieren mit RSA-Schlüsseln
- Signieren und Verifizieren mit EC-Schlüsseln
- Unterstützte Schlüsselattribute verwenden
- Verwenden des CloudHSM-Schlüsselspeichers

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

AWS CloudHSM-JCE-Anbieter Javadocs

Verwenden Sie den JCE-Anbieter Javadocs, um Nutzungsinformationen zu Java-Typen und -Methoden abzurufen, die im AWS CloudHSM JCE SDK definiert sind. Informationen zum Herunterladen der neuesten Javadocs für AWS CloudHSM finden Sie im Abschnitt Neuste Version auf der Download-Seite.

Sie können Javadocs in eine integrierte Entwicklungsumgebung (IDE) importieren oder sie in einem Webbrowser anzeigen.

AWS CloudHSM KeyStore Java-Klasse verwenden

Die AWS CloudHSM `KeyStore` Klasse stellt einen speziellen PKCS12-Schlüsselspeicher bereit. Dieser Schlüsselspeicher kann Zertifikate zusammen mit Ihren Schlüsseldaten speichern und sie mit den in AWS CloudHSM gespeicherten Schlüsseldaten korrelieren. Die AWS CloudHSM `KeyStore` Klasse implementiert das `KeyStore` Service Provider Interface (SPI) der Java Cryptography Extension (JCE). Weitere Informationen zur Verwendung `KeyStore` finden Sie unter Class. KeyStore

> **Note**
>
> Da es sich bei Zertifikaten um öffentliche Informationen handelt und um die Speicherkapazität für kryptografische Schlüssel zu maximieren, AWS CloudHSM wird das Speichern von Zertifikaten auf HSMs nicht unterstützt.

Auswählen des passenden Schlüsselspeichers

Der Anbieter AWS CloudHSM Java Cryptographic Extension (JCE) bietet ein spezielles AWS CloudHSM an. KeyStore Die AWS CloudHSM `KeyStore` Klasse unterstützt das Auslagern wichtiger Operationen in das HSM, die lokale Speicherung von Zertifikaten und zertifikatsbasierte Operationen.

Laden Sie das spezielle CloudHSM KeyStore wie folgt:

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Initialisieren AWS CloudHSM KeyStore

Melden Sie sich auf AWS CloudHSM KeyStore die gleiche Weise an, wie Sie sich beim JCE-Anbieter anmelden. Sie können entweder Umgebungsvariablen oder die Systemeigenschaftendatei verwenden, und Sie sollten sich anmelden, bevor Sie CloudHSM KeyStore verwenden. Ein Beispiel für die Anmeldung bei einem HSM mit dem JCE-Anbieter finden Sie unter Anmeldung an HSM.

Falls gewünscht, können Sie ein Passwort angeben, um die lokale PKCS12-Datei zu verschlüsseln, die Schlüsselspeicherdaten enthält. Wenn Sie den AWS CloudHSM Keystore erstellen, legen Sie das Passwort fest und geben es an, wenn Sie die Methoden load, set und get verwenden.

Instanziieren Sie ein neues CloudHSM-Objekt KeyStore wie folgt:

```
ks.load(null, null);
```

Schreiben Sie Schlüsselspeicherdaten mit der `store`-Methode in eine Datei. Von da an können Sie den vorhandenen Schlüsselspeicher mit der `load`-Methode mit der Quelldatei und dem Passwort wie folgt laden:

```
ks.load(inputStream, password);
```

Verwenden AWS CloudHSM KeyStore

AWS CloudHSM KeyStore entspricht der KeyStoreJCE-Klassenspezifikation und bietet die folgenden Funktionen.

- `load`

  Lädt den Schlüsselspeicher aus dem angegebenen Eingabe-Stream. Wenn beim Speichern des Schlüsselspeichers ein Passwort festgelegt wurde, muss dasselbe Passwort angegeben werden, damit das Laden erfolgreich ist. Setzen Sie beide Parameter auf null, um einen neuen leeren Schlüsselspeicher zu initialisieren.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- `aliases`

  Gibt eine Aufzählung der Aliasnamen aller Einträge in der angegebenen Schlüsselspeicherinstanz zurück. Die Ergebnisse umfassen Objekte, die lokal in der PKCS12-Datei gespeichert sind, sowie Objekte, die sich auf dem HSM befinden.

  Beispiel-Code:

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {
      String label = entry.nextElement();
  ```

```
    System.out.println(label);
}
```

- `containsalias`

  Gibt „true“ zurück, wenn der Schlüsselspeicher Zugriff auf mindestens ein Objekt mit dem angegebenen Alias hat. Der Schlüsselspeicher prüft lokal in der PKCS12-Datei gespeicherte Objekte und Objekte auf dem HSM.

- `deleteEntry`

  Löscht einen Zertifikateintrag aus der lokalen PKCS12-Datei. Das Löschen von Schlüsseldaten, die in einem HSM gespeichert sind, wird mit dem nicht unterstützt. AWS CloudHSM KeyStore Sie können Schlüssel mit der `destroy`-Methode der Destroyable-Schnittstelle löschen.

  ```
  ((Destroyable) key).destroy();
  ```

- `getCertificate`

  Gibt das Zertifikat zurück, das einem Alias zugeordnet ist, falls verfügbar. Wenn der Alias nicht existiert oder auf ein Objekt verweist, das kein Zertifikat ist, gibt die Funktion NULL zurück.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  Certificate cert = ks.getCertificate(alias);
  ```

- `getCertificateAlias`

  Gibt den Namen (Alias) des ersten Schlüsselspeichereintrags zurück, dessen Daten mit dem angegebenen Zertifikat übereinstimmen.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  String alias = ks.getCertificateAlias(cert);
  ```

- `getCertificateChain`

  Gibt die Zertifikatkette zurück, die dem angegebenen Alias zugeordnet ist. Wenn der Alias nicht existiert oder auf ein Objekt verweist, das kein Zertifikat ist, gibt die Funktion NULL zurück.

- `getCreationDate`

  Gibt das Erstellungsdatum des durch den angegebenen Alias identifizierten Eintrags zurück. Wenn kein Erstellungsdatum verfügbar ist, gibt die Funktion das Datum zurück, an dem das Zertifikat gültig wurde.

- `getKey`

  GetKey wird an das HSM übergeben und gibt ein Schlüsselobjekt zurück, das dem angegebenen Label entspricht. Da es das HSM `getKey` direkt abfragt, kann es für jeden Schlüssel auf dem HSM verwendet werden, unabhängig davon, ob er von der generiert wurde. KeyStore

  ```
  Key key = ks.getKey(keyLabel, null);
  ```

- `isCertificateEntry`

  Überprüft, ob der Eintrag mit dem angegebenen Alias einen Zertifikateintrag darstellt.

- `isKeyEntry`

  Überprüft, ob der Eintrag mit dem angegebenen Alias einen Schlüsseleintrag darstellt. Die Aktion durchsucht sowohl die PKCS12-Datei, als auch das HSM nach dem Alias.

- `setCertificateEntry`

  Weist dem angegebenen Alias das angegebene Zertifikat zu. Wenn der angegebene Alias bereits verwendet wird, um einen Schlüssel oder ein Zertifikat zu identifizieren, wird ein `KeyStoreException` ausgelöst. Sie können JCE-Code verwenden, um das Schlüsselobjekt abzurufen, und dann die KeyStore `SetKeyEntry` Methode verwenden, um das Zertifikat dem Schlüssel zuzuordnen.

- `setKeyEntry` mit `byte[]`-Schlüssel

  Diese API wird derzeit vom Client-SDK 5 nicht unterstützt.

- `setKeyEntry` mit `Key`-Objekt

  Weist den angegebenen Schlüssel dem angegebenen Alias zu und speichert ihn im HSM. Wenn der Schlüssel noch nicht im HSM vorhanden ist, wird er als extrahierbarer Sitzungsschlüssel in das HSM importiert.

  Wenn das `Key`-Objekt vom Typ `PrivateKey` ist, muss es von einer entsprechenden Zertifikatkette begleitet werden.

  Wenn der Alias bereits vorhanden ist, wird ein `SetKeyEntry`- Aufruf ausgelöst, und `KeyStoreException` verhindert, dass der Schlüssel überschrieben wird. Wenn der Schlüssel überschrieben werden muss, verwenden Sie hierfür KMU oder JCE.

- `engineSize`

Gibt die Anzahl der Einträge im Schlüsselspeicher zurück.

- `store`

  Speichert den Schlüsselspeicher im angegebenen Ausgabe-Stream als PKCS12-Datei und sichert ihn mit dem angegebenen Passwort. Darüber hinaus bleiben alle geladenen Schlüssel (die über `setKey`-Aufrufe gesetzt werden) bestehen.

Migrieren Sie Ihren JCE-Anbieter von Client SDK 3 auf Client SDK 5

Verwenden Sie dieses Thema, um Ihren JCE-Anbieter von Client SDK 3 auf Client SDK 5 zu migrieren. Informationen zu den Vorteilen der Migration finden Sie unter. Vorteile von Client-SDK 5

In AWS CloudHSM führen Kundenanwendungen mithilfe des AWS CloudHSM Client Software Development Kit (SDK) kryptografische Operationen durch. Das Client SDK 5 ist das primäre SDK, das weiterhin um neue Funktionen und Plattformunterstützung erweitert wird.

Der Client SDK 3 JCE-Anbieter verwendet benutzerdefinierte Klassen und APIs, die nicht Teil der Standard-JCE-Spezifikation sind. Das Client-SDK 5 für den JCE-Anbieter entspricht der JCE-Spezifikation und ist in bestimmten Bereichen abwärtsinkompatibel mit dem Client SDK 3. Kundenanwendungen müssen im Rahmen der Migration zum Client-SDK 5 möglicherweise geändert werden. In diesem Abschnitt werden die Änderungen beschrieben, die für eine erfolgreiche Migration erforderlich sind.

Die Migrationsanweisungen für alle Anbieter finden Sie unterMigration von Client-SDK 3 zu Client-SDK 5.

Themen

- Bereiten Sie sich darauf vor, indem Sie wichtige Änderungen berücksichtigen
- Migrieren Sie zum Client SDK 5
- Verwandte Themen

Bereiten Sie sich darauf vor, indem Sie wichtige Änderungen berücksichtigen

Überprüfen Sie diese grundlegenden Änderungen und aktualisieren Sie Ihre Anwendung in Ihrer Entwicklungsumgebung entsprechend.

Die Provider-Klasse und der Name haben sich geändert

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Klasse und Name des Anbieters	Die JCE-Anbieterklasse im Client SDK 3 wird aufgerufen `CaviumProvider` und hat den Anbieternamen `Cavium`.	In Client SDK 5 wird die Provider-Klasse aufgerufen `CloudHsmProvider` und hat den Provider-Namen `CloudHSM`.	Ein Beispiel für die Initialisierung des `CloudHsmProvider` Objekts ist im AWS CloudHSM GitHub Beispiel-Repository verfügbar.

Die explizite Anmeldung hat sich geändert, die implizite nicht

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Explizite Anmeldung	Das Client-SDK 3 verwendet die `LoginManager` Klasse für die explizite Anmeldung [1].	Im Client-SDK 5 implementiert der `CloudHSM` Anbieter `AuthProvider` die explizite Anmeldung. `AuthProvider` ist eine Standard-Java-Klasse und folgt der idiomatischen Methode von Java, sich bei einem Provider anzumelden. Dank der verbesserten Verwaltung des Anmeldestatus im Client SDK 5 müssen Anwendungen die Anmeldung bei	Ein Beispiel für die Verwendung der expliziten Anmeldung mit Client SDK 5 finden Sie im LoginRunner Beispiel im AWS CloudHSM GitHub CloudHSM-Beispiel-Repository.

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
		erneuten [2]Verbindungen nicht mehr überwachen und durchführen.	
Implizite Anmeldung	Für die implizite Anmeldung sind keine Änderungen erforderlich. Dieselbe Eigenschaftendatei und alle Umgebungsvariablen funktionieren weiterhin für die implizite Anmeldung, wenn von Client SDK 3 auf Client SDK 5 migriert wird.		Ein Beispiel für die Verwendung der impliziten Anmeldung mit dem Client SDK 5 finden Sie im LoginRunner Beispiel im AWS CloudHSM GitHub Beispiel-Repository.

- [1] Codeausschnitt für das Client-SDK 3:

```
LoginManager lm = LoginManager.getInstance();

lm.login(partition, user, pass);
```

- [2] Codeausschnitt für das Client-SDK 5:

```
// Construct or get the existing provider object
AuthProvider provider = new CloudHsmProvider();

// Call login method on the CloudHsmProvider object
// Here loginHandler is a CallbackHandler
provider.login(null, loginHandler);
```

Ein Beispiel für die Verwendung der expliziten Anmeldung mit dem Client SDK 5 finden Sie im LoginRunner Beispiel im AWS CloudHSM GitHub Beispiel-Repository.

Die Schlüsselgenerierung hat sich geändert

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Schlüsselgenerierung	Wird im Client-SDK 3 zur Angabe von Schlüsselgenerierungsparametern verwendet. `Cavium[Key-type]AlgorithmParameterSpec` Einen Codeausschnitt finden Sie in der Fußnote. 1	Wird im Client SDK 5 zur Angabe von `KeyAttributesMap` Schlüsselgenerierungsattributen verwendet. Einen Codeausschnitt finden Sie in der Fußnote. 2	Ein Beispiel `KeyAttributesMap` zur Generierung eines symmetrischen Schlüssels finden Sie im Beispiel im AWS CloudHSM Symmetric Keys Github-Beispiel-Repository.
Generierung von Schlüsselpaaren	Wird im Client SDK 3 verwendet, `Cavium[Key-type]Algorithmpa rameterSpec` um Parameter für die Schlüsselpaargenerierung anzugeben. Einen Codeausschnitt finden Sie in der Fußnote. 3	Wird im Client SDK 5 verwendet, `KeyPairAttributesMap` um diese Parameter anzugeben. Einen Codeausschnitt finden Sie in der Fußnote. 4	Ein Beispiel `KeyAttributesMap` zur Generierung eines asymmetrischen Schlüssels finden Sie im AsymmetricKeys Beispiel im Beispiel-Repository. AWS CloudHSM GitHub

- [1] Codeausschnitt zur Schlüsselgenerierung des Client-SDK 3:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec aesSpec = new CaviumAESKeyGenParameterSpec(
keySizeInBits,
keyLabel,
```

```
isExtractable,
isPersistent);
keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [2] Codeausschnitt zur Schlüsselgenerierung des Client-SDK 5:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES",
CloudHsmProvider.PROVIDER_NAME);

final KeyAttributesMap aesSpec = new KeyAttributesMap();
aesSpec.put(KeyAttribute.LABEL, keyLabel);
aesSpec.put(KeyAttribute.SIZE, keySizeInBits);
aesSpec.put(KeyAttribute.EXTRACTABLE, isExtractable);
aesSpec.put(KeyAttribute.TOKEN, isPersistent);

keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [3] Codeausschnitt zur Generierung von key pair für das Client-SDK 3:

```
KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("rsa", "Cavium");
CaviumRSAKeyGenParameterSpec spec = new CaviumRSAKeyGenParameterSpec(
keySizeInBits,
new BigInteger("65537"),
label + ":public",
label + ":private",
isExtractable,
isPersistent);

keyPairGen.initialize(spec);

keyPairGen.generateKeyPair();
```

- [4] Codeausschnitt zur Generierung von key pair für das Client-SDK 5:

```
KeyPairGenerator keyPairGen =
KeyPairGenerator.getInstance("RSA", providerName);

// Set attributes for RSA public key
final KeyAttributesMap publicKeyAttrsMap = new KeyAttributesMap();
publicKeyAttrsMap.putAll(additionalPublicKeyAttributes);
publicKeyAttrsMap.put(KeyAttribute.LABEL, label + ":Public");
```

```
publicKeyAttrsMap.put(KeyAttribute.MODULUS_BITS, keySizeInBits);
publicKeyAttrsMap.put(KeyAttribute.PUBLIC_EXPONENT,
new BigInteger("65537").toByteArray());

// Set attributes for RSA private key
final KeyAttributesMap privateKeyAttrsMap = new KeyAttributesMap();
privateKeyAttrsMap.putAll(additionalPrivateKeyAttributes);
privateKeyAttrsMap.put(KeyAttribute.LABEL, label + ":Private");

// Create KeyPairAttributesMap and use that to initialize the
// keyPair generator
KeyPairAttributesMap keyPairSpec =
new KeyPairAttributesMapBuilder()
.withPublic(publicKeyAttrsMap)
.withPrivate(privateKeyAttrsMap)
.build();

keyPairGen.initialize(keyPairSpec);
keyPairGen.generateKeyPair();
```

Das Suchen, Löschen und Referenzieren von Schlüsseln hat sich geändert

Um einen bereits generierten Schlüssel mit zu finden, AWS CloudHSM müssen Sie den KeyStore verwenden. Das Client SDK 3 hat zwei KeyStore Typen: `Cavium` und`CloudHSM`. Das Client-SDK 5 hat nur einen KeyStore Typ:`CloudHSM`.

Der Wechsel von A `Cavium` KeyStore nach `CloudHSM` KeyStore erfordert eine Änderung des KeyStore Typs. Darüber hinaus verwendet das Client-SDK 3 Schlüsselnamen, um auf Schlüssel zu verweisen, während das Client-SDK 5 Tastenbezeichnungen verwendet. Die daraus resultierenden Verhaltensänderungen sind unten aufgeführt.

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Wichtige Referenzen	Mit Client SDK 3 verwenden Anwendungen entweder Tastenbezeichnungen oder Schlüsselhandles,	Im Client SDK 5 können Anwendungen die verwenden, AWS CloudHSM KeyStore Java-Klasse verwenden um	

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
	um auf Schlüssel im HSM zu verweisen . Sie verwenden Labels mit KeyStore , um einen Schlüssel zu finden, oder sie verwenden Griffe und erstellen `CaviumKey` Objekte.	Schlüssel anhand von Labels zu finden. Verwenden Sie with, um Schlüssel anhand des Handles AWS CloudHSM `KeyStoreWithAttributes` zu finden AWS CloudHSM `KeyRefereneSpec` .	

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Suche nach mehreren Einträgen	Bei der Suche nach einem Schlüssel mithilfe von `getEntry`, `getKey`, oder `getCertificate` in Szenarien, in denen mehrere Elemente mit denselben Kriterien existieren `Cavium` KeyStore, wird nur der erste gefundene Eintrag zurückgegeben.	Mit dem AWS CloudHSM `KeyStore` und führt dasselbe Szenario dazu`KeyStoreWithAttributes`, dass eine Ausnahme ausgelöst wird. Um dieses Problem zu beheben, wird empfohlen, mithilfe des key set-attribute Befehls in der CloudHSM-CLI eindeutige Bezeichnungen für Schlüssel festzulegen. Oder verwenden Sie diese `KeyStoreWithAttributes#getKeys` Option, um alle Schlüssel zurückzugeben, die den Kriterien entsprechen.	

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Finde alle Schlüssel	Im Client SDK 3 ist es möglich, alle Schlüssel im HSM mithilfe von `Util.findAllKeys()` zu finden.	Das Client SDK 5 macht das Auffinden von Schlüsseln mithilfe der `KeyStoreWithAttributes` Klasse einfacher und effizienter. Wenn möglich, speichern Sie Ihre Schlüssel im Cache, um die Latenz zu minimieren. Weitere Informationen finden Sie unter Verwalten Sie Schlüssel in Ihrer Anwendung effektiv. Wenn Sie alle Schlüssel aus dem HSM abrufen müssen, verwenden Sie es `KeyStoreWithAttributes#getKeys` mit einem leeren `KeyAttributesMap` Schlüssel.	Ein Beispiel, das die `KeyStoreWithAttributes` Klasse verwendet, um einen Schlüssel zu finden, ist im AWS CloudHSM Github-Beispiel-Repository verfügbar. Ein Codeausschnitt ist unter zu finden. 1

Was hat sich geändert	Was es in Client SDK 3 war	Was ist es in Client SDK 5	Beispiel
Löschen von Schlüsseln	Client SDK 3 verwendet `Util.deleteKey()` , um einen Schlüssel zu löschen.	Das `Key` Objekt in Client SDK 5 implementiert die `Destroyable` Schnittstelle, die das Löschen von Schlüsseln mithilfe der `destroy()` Methode dieser Schnittstelle ermöglicht.	Ein Beispielcode, der die Funktion zum Löschen von Schlüsseln zeigt, finden Sie im CloudHSM Github-Beispiel-Repository. Ein Beispielausschnitt für jedes SDK finden Sie unter. 2

- [1] Ein Ausschnitt ist unten dargestellt:

```
KeyAttributesMap findSpec = new KeyAttributesMap();
findSpec.put(KeyAttribute.LABEL, label);
findSpec.put(KeyAttribute.KEY_TYPE, keyType);
KeyStoreWithAttributes keyStore = KeyStoreWithAttributes.getInstance("CloudHSM");

keyStore.load(null, null);
keyStore.getKey(findSpec);
```

- [2] Löschen eines Schlüssels im Client SDK 3:

```
Util.deleteKey(key);
```

  Löschen eines Schlüssels im Client-SDK 5:

```
((Destroyable) key).destroy();
```

Operationen zum Entpacken von Chiffren haben sich geändert, andere Verschlüsselungsoperationen nicht

> **Note**
>
> Für Verschlüsselungs-, Entschlüsselungs- und Umschließungsvorgänge mit Chiffrierung sind keine Änderungen erforderlich.

Bei Unwrap-Vorgängen muss die `CaviumUnwrapParameterSpec` Klasse Client SDK 3 durch eine der folgenden Klassen ersetzt werden, die für die aufgeführten kryptografischen Operationen spezifisch sind.

- `GCMUnwrapKeySpec`zum Auspacken `AES/GCM/NoPadding`
- `IvUnwrapKeySpec`für und `AESWrap unwrap` `AES/CBC/NoPadding unwrap`
- `OAEPUnwrapKeySpec` für `RSA OAEP unwrap`

Beispielausschnitt für: `OAEPUnwrapkeySpec`

```
OAEPParameterSpec oaepParameterSpec =
new OAEPParameterSpec(
        "SHA-256",
        "MGF1",
        MGF1ParameterSpec.SHA256,
        PSpecified.DEFAULT);

KeyAttributesMap keyAttributesMap =
        new KeyAttributesMap(KeyAttributePermissiveProfile.KEY_CREATION);
keyAttributesMap.put(KeyAttribute.TOKEN, true);
keyAttributesMap.put(KeyAttribute.EXTRACTABLE, false);

OAEPUnwrapKeySpec spec = new OAEPUnwrapKeySpec(oaepParameterSpec,
        keyAttributesMap);

Cipher hsmCipher =
        Cipher.getInstance(
                "RSA/ECB/OAEPPadding",
                CloudHsmProvider.PROVIDER_NAME);
hsmCipher.init(Cipher.UNWRAP_MODE, key, spec);
```

Die Signaturoperationen haben sich nicht geändert

Für Signaturoperationen sind keine Änderungen erforderlich.

Migrieren Sie zum Client SDK 5

Folgen Sie den Anweisungen in diesem Abschnitt, um von Client SDK 3 auf Client SDK 5 zu migrieren.

> **Note**
>
> Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 und RHEL 6 werden derzeit nicht mit Client SDK 5 unterstützt. Wenn Sie derzeit eine dieser Plattformen mit Client SDK 3 verwenden, müssen Sie bei der Migration zu Client SDK 5 eine andere Plattform wählen.

1. Deinstallieren Sie den JCE-Anbieter für Client SDK 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-jce
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-jce
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-jce
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-jce
   ```

2. Deinstallieren Sie den Client Daemon für Client SDK 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client
   ```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

> **Note**
>
> Benutzerdefinierte Konfigurationen müssen erneut aktiviert werden.

3. Installieren Sie den Client SDK JCE Provider, indem Sie die Schritte unter befolgen. Installieren und verwenden Sie den AWS CloudHSM JCE-Anbieter für Client SDK 5
4. Das Client SDK 5 führt ein neues Konfigurationsdateiformat und ein Befehlszeilen-Bootstrapping-Tool ein. Folgen Sie den Anweisungen im Benutzerhandbuch unter, um Ihren Client SDK 5 JCE-Anbieter zu booten. Bootstrap für das Client-SDK
5. Testen Sie Ihre Anwendung in Ihrer Entwicklungsumgebung. Nehmen Sie vor der endgültigen Migration Aktualisierungen an Ihrem vorhandenen Code vor, um Ihre wichtigsten Änderungen zu beheben.

Verwandte Themen

- Bewährte Methoden für AWS CloudHSM

Erweiterte Konfigurationen für JCE

Der AWS CloudHSM JCE-Anbieter umfasst die folgenden erweiterten Konfigurationen, die nicht Teil der allgemeinen Konfigurationen sind, die die meisten Kunden verwenden.

- Verbindung zu mehreren Clustern herstellen

- Schlüsselextraktion mit JCE

- Versuchen Sie erneut, die Konfiguration für JCE zu konfigurieren

Verbindung zu mehreren Clustern mit dem JCE-Anbieter herstellen

Diese Konfiguration ermöglicht es einer einzelnen Client-Instance, mit mehreren Clustern zu kommunizieren Im Vergleich dazu, dass eine einzelne Instance nur mit einem einzigen Cluster kommuniziert, kann dies in einigen Anwendungsfällen zu Kosteneinsparungen führen. Die `CloudHsmProvider`-Klasse ist die Implementierung von AWS CloudHSM der Anbieter-Klasse von Java Security. Jede Instance dieser Klasse stellt eine Verbindung zu Ihrem gesamten AWS CloudHSM-Cluster dar. Sie instanziieren diese Klasse und fügen sie der Liste der Java-Security-Anbieter hinzu, so dass Sie mit ihr über Standard-JCE-Klassen interagieren können.

Das folgende Beispiel instanziiert diese Klasse und fügt sie der Liste der Java-Security-Anbieter hinzu:

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {
    Security.addProvider(new CloudHsmProvider());
}
```

`CloudHsmProvider`-Konfiguration

`CloudHsmProvider` kann auf zwei Arten konfiguriert werden:

1. Mit Datei konfigurieren (Standardkonfiguration)
2. Mit Code konfigurieren

Mit Datei konfigurieren (Standardkonfiguration)

Wenn Sie `CloudHsmProvider` mit dem Standardkonstruktor instanziieren, sucht er standardmäßig nach einer Konfigurationsdatei im `/opt/cloudhsm/etc/cloudhsm-jce.cfg`-Pfad unter Linux. Diese Konfigurationsdatei kann mit der Datei `configure-jce` konfiguriert werden.

Ein mit dem Standardkonstruktor erstelltes Objekt verwendet den standardmäßigen CloudHSM-Anbieternamen `CloudHSM`. Der Anbietername ist nützlich, um mit JCE zu interagieren und ihm mitzuteilen, welcher Anbieter für verschiedene Operationen verwendet werden soll. Ein Beispiel für die Verwendung des CloudHSM-Anbieternamens für den Chiffriervorgang lautet wie folgt:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHSM");
```

Mit Code konfigurieren

Ab Version 5.8.0 des Client-SDK können Sie den `CloudHsmProvider` auch mithilfe von Java-Code konfigurieren. Dazu verwenden Sie ein Objekt der `CloudHsmProviderConfig`-Klasse. Sie können `CloudHsmProviderConfigBuilder` verwenden, um dieses Objekt zu entwickeln.

`CloudHsmProvider` hat einen anderen Konstruktor, der das `CloudHsmProviderConfig`-Objekt aufnimmt, wie das folgende Beispiel zeigt.

Example

```
CloudHsmProviderConfig config = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath)

 .withClusterUniqueIdentifier("CloudHsmCluster1")
        .withServer(CloudHsmServer.builder().withHostIP(hostName).build())
                        .build())
        .build();
CloudHsmProvider provider = new CloudHsmProvider(config);
```

In diesem Beispiel lautet der Name des JCE-Anbieters `CloudHsmCluster1`. Dies ist der Name, den die Anwendung dann für die Interaktion mit JCE verwenden kann:

Example

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHsmCluster1");
```

Alternativ können Anwendungen auch das oben erstellte Anbieter-Objekt verwenden, um JCE mitzuteilen, dass es diesen Anbieter für den Vorgang verwenden soll:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider);
```

Wenn mit der `withClusterUniqueIdentifier`-Methode kein eindeutiger Bezeichner angegeben wird, wird ein zufällig generierter Anbietername für Sie erstellt. Um diesen zufällig generierten Bezeichner zu erhalten, können Anwendungen `provider.getName()` aufrufen, um den Bezeichner abzurufen.

Verbindung zu mehreren Clustern herstellen

Wie oben angegeben, steht jeder `CloudHsmProvider` für eine Verbindung zu Ihrem CloudHSM-Cluster. Wenn Sie mit einem anderen Cluster aus derselben Anwendung kommunizieren möchten, können Sie ein anderes Objekt von `CloudHsmProvider` mit Konfigurationen für Ihren anderen Cluster erstellen und mit diesem anderen Cluster entweder über das Anbieterobjekt oder über den Anbieternamen interagieren, wie im folgenden Beispiel gezeigt.

Example

```
CloudHsmProviderConfig config = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath)

 .withClusterUniqueIdentifier("CloudHsmCluster1")
        .withServer(CloudHsmServer.builder().withHostIP(hostName).build())
                        .build())
        .build();
CloudHsmProvider provider1 = new CloudHsmProvider(config);

if (Security.getProvider(provider1.getName()) == null) {
    Security.addProvider(provider1);
}

CloudHsmProviderConfig config2 = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath2)

 .withClusterUniqueIdentifier("CloudHsmCluster2")
        .withServer(CloudHsmServer.builder().withHostIP(hostName2).build())
                        .build())
        .build();
CloudHsmProvider provider2 = new CloudHsmProvider(config2);

if (Security.getProvider(provider2.getName()) == null) {
    Security.addProvider(provider2);
}
```

Nachdem Sie beide Anbieter (beide Cluster) oben konfiguriert haben, können Sie entweder über das Anbieterobjekt oder über den Anbieternamen mit ihnen interagieren.

Als Ergänzung zu diesem Beispiel, das zeigt, wie man mit `cluster1` spricht, könnten Sie das folgende Beispiel für einen AES/GCM/NoPadding-Vorgang verwenden:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```

Und in derselben Anwendung zur Generierung von „AES“-Schlüsseln auf dem zweiten Cluster unter Verwendung des Anbieternamens könnten Sie auch das folgende Beispiel verwenden:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

Wiederholungsbefehle für JCE

Das Client-SDK 5.8.0 und höher verfügen über eine integrierte automatische Wiederholungsstrategie, mit der HSM-gedrosselte Operationen von der Clientseite aus wiederholt werden. Wenn ein HSM Operationen drosselt, weil es zu sehr mit der Ausführung früherer Operationen beschäftigt ist und keine weiteren Anfragen annehmen kann, versuchen Client-SDKs, gedrosselte Operationen bis zu dreimal zu wiederholen, während es exponentiell abbricht. Diese automatische Wiederholungsstrategie kann auf einen von zwei Modi eingestellt werden: aus und Standard.

- aus: Das Client-SDK führt bei gedrosselten Vorgängen durch das HSM keine Wiederholungsstrategie durch.
- Standard: Dies ist der Standardmodus für das Client-SDK 5.8.0 und höher. In diesem Modus wiederholen die Client-SDKs automatisch gedrosselte Operationen, indem sie sich exponentiell zurückziehen.

Weitere Informationen finden Sie unter HSM-Drosselung.

Stellen Sie den Modus für Wiederholungsbefehle auf Aus

Linux

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Linux

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
$ sudo /opt/cloudhsm/bin/configure-jce --default-retry-mode off
```

Windows

So setzen Sie Wiederholungsbefehle auf off für Client-SDK 5 unter Windows

- Sie können den folgenden Befehl verwenden, um die Konfiguration der Wiederholungen auf den off-Modus zu setzen:

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-jce.exe --default-retry-mode
 off
```

Schlüsselextraktion mit JCE

Die Java Cryptography Extension (JCE) verwendet eine Architektur, mit der verschiedene Kryptografie-Implementierungen integriert werden können. AWS CloudHSM liefert einen solchen JCE-Anbieter aus, der kryptografische Operationen an das HSM auslagert. Damit die meisten anderen JCE-Anbieter mit Schlüsseln arbeiten können, die in AWS CloudHSM gespeichert sind, müssen sie die Schlüsselbytes aus Ihren HSMs im Klartext in den Speicher Ihres Computers extrahieren, damit sie sie verwenden können. HSMs ermöglichen in der Regel nur das Extrahieren von Schlüsseln als verpackte Objekte, nicht als Klartext. Um Anwendungsfälle der anbieterübergreifenden Integration zu unterstützen, lässt AWS CloudHSM jedoch eine Opt-in-Konfigurationsoption zu, mit der die Schlüsselbytes im Klartext extrahiert werden können.

> **Important**
>
> JCE lagert Operationen immer dann an AWS CloudHSM aus, wenn der AWS-CloudHSM-Anbieter angegeben oder ein AWS CloudHSM-Schlüsselobjekt verwendet wird. Sie müssen Schlüssel nicht im Klartext extrahieren, wenn Sie erwarten, dass Ihr Vorgang innerhalb des HSM stattfindet. Die Schlüsselextraktion im Klartext ist nur erforderlich, wenn Ihre Anwendung aufgrund von Einschränkungen durch eine Drittanbieterbibliothek oder einen JCE-Anbieter keine sicheren Mechanismen wie das Ein- und Auspacken eines Schlüssels verwenden kann.

Der AWS CloudHSM-JCE-Anbieter ermöglicht standardmäßig die Extraktion von öffentlichen Schlüsseln, sodass er mit externen JCE-Anbietern funktioniert. Die folgenden Methoden sind immer zulässig:

Klasse	Method	Format (getEncoded)
EcPublicKey	getEncoded()	X.509
	getW()	–
RSAPublicKey	getEncoded()	X.509
	getPublicExponent()	–
CloudHsmRsaPrivateCrtKey	getPublicExponent()	–

Der AWS CloudHSM-JCE-Provider erlaubt standardmäßig keine Extraktion von Schlüsselbytes im Klartext für private oder geheime Schlüssel. Wenn Ihr Anwendungsfall dies erfordert, können Sie die Extraktion von Schlüsselbytes im Klartext für private oder geheime Schlüssel unter den folgenden Bedingungen aktivieren:

1. Das `EXTRACTABLE`-Attribut für private und geheime Schlüssel ist auf wahr gesetzt.
   - Standardmäßig ist das `EXTRACTABLE`-Attribut für private und geheime Schlüssel auf wahr gesetzt. `EXTRACTABLE`-Schlüssel sind Schlüssel, die aus dem HSM exportiert werden dürfen. Weitere Informationen finden Sie unter Unterstützte Java-Attribute für Client-SDK 5.
2. Das `WRAP_WITH_TRUSTED`-Attribut für private und geheime Schlüssel ist auf falsch gesetzt.
   - `getEncoded`, `getPrivateExponent` und `getS` können nicht mit privaten Schlüsseln verwendet werden, die nicht in Klartext exportiert werden können. `WRAP_WITH_TRUSTED` erlaubt nicht, dass Ihre privaten Schlüssel in Klartext aus dem HSM exportiert werden. Weitere Informationen finden Sie unter Verwenden vertrauenswürdiger Schlüssel zur Steuerung des Entpackens von Schlüsseln.

Erlauben Sie dem AWS CloudHSM-JCE-Anbieter, geheime private Schlüssel aus AWS CloudHSM zu extrahieren

> **Important**
>
> Diese Konfigurationsänderung ermöglicht das Extrahieren aller `EXTRACTABLE`-Schlüsselbytes in Clear aus Ihrem HSM-Cluster. Aus Sicherheitsgründen sollten Sie erwägen, Schlüssel mithilfe von Methoden zum Wrapping von Schlüsseln sicher aus dem

HSM zu extrahieren. Dadurch wird ein unbeabsichtigtes Extrahieren Ihrer Schlüsselbytes aus dem HSM verhindert.

1. Verwenden Sie die folgenden Befehle, um das Extrahieren Ihrer privaten oder geheimen Schlüssel in JCE zu ermöglichen:

   Linux

   ```
   $ /opt/cloudhsm/bin/configure-jce --enable-clear-key-extraction-in-software
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\> .\configure-jce.exe --enable-clear-key-extraction-in-software
   ```

2. Sobald Sie die Extraktion Ihres eindeutigen Schlüssels aktivieren, sind die folgenden Methoden zur Extraktion privater Schlüssel in den Speicher aktiviert.

Klasse	Method	Format (getEncoded)
Schlüssel	getEncoded()	RAW
ECPrivateKey	getEncoded()	PKCS#8
	getS()	–
RSAPrivateCrtKey	getEncoded()	X.509
	getPrivateExponent()	–
	getPrimeP()	–
	getPrimeQ()	–
	getPrimeExponentP()	–
	getPrimeExponentQ()	–

Klasse	Method	Format (getEncoded)
	getCrtCoefficient()	–

Wenn Sie das Standardverhalten wiederherstellen und JCE nicht erlauben möchten, Schlüssel in Clear zu exportieren, führen Sie den folgenden Befehl aus:

Linux

```
$ /opt/cloudhsm/bin/configure-jce --disable-clear-key-extraction-in-software
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-jce.exe --disable-clear-key-
extraction-in-software
```

Kryptografie-API: CNG (Next Generation) und Key Storage Providers (KSP) für Microsoft Windows

Der AWS CloudHSM-Client für Windows enthält die CNG- und KSP-Anbieter. Derzeit unterstützt nur Client-SDK 3 CNG- und KSP-Anbieter.

KSPs (Key Storage Provider = Schlüsselspeicher-Anbieter) ermöglichen das Speichern und Abrufen von Schlüsseln. Wenn Sie beispielsweise auf einem Windows-Server die Rolle Microsoft Active Directory-Zertifikatsdienste (AD CS) hinzufügen und einen neuen privaten Schlüssel für die Zertifizierungsstelle (CA) erstellen, können Sie KSP die Verwaltung des Schlüsselspeichers überlassen. Wenn Sie die AD CS-Rolle konfigurieren, können Sie diesen KSP wählen. Weitere Informationen finden Sie unter Erstellen einer Windows Server-CA.

Kryptografie-API: Next-Generation (CNG) ist eine Kryptografie-API für das Betriebssystem Microsoft Windows. Mit CNG können Kryptografietechniken zum Schützen von Windows-Anwendungen einsetzen. Im Großen und Ganzen bietet die AWS CloudHSM-Implementierung von CNG die folgenden Funktionen:

- Kryptografische Grundfunktionen ermöglichen grundlegende kryptografische Operationen.
- Schlüsselimport/-export zum Importieren und Exportieren von asymmetrischen Schlüsseln.
- Datenschutz-API (CNG DPAPI) für die einfache Ver- und Entschlüsselung von Daten.
- Schlüsselspeicherung und -abruf zum sicheren Speichern sowie zum Isolieren des privaten Schlüssels eines asymmetrischen Schlüsselpaars.

Themen

Überprüfen der KSP- und CNG-Anbieter für Windows

Die KSP- und CNG-Provider werden mit dem Windows–AWS CloudHSMClient installiert. Sie können den Client installieren, indem Sie die unter Installieren des Clients (Windows) beschriebenen Schritte durchführen.

Konfigurieren und Ausführen des Windows-AWS CloudHSM-Clients

Bevor Sie den Windows CloudHSM-Client starten können, müssen Sie die Anforderungen unter Voraussetzungen erfüllen. Anschließend aktualisieren Sie die von den Anbietern verwendeten Konfigurationsdateien und starten den Client, indem Sie die folgenden Schritte durchführen:
Sie müssen diese Schritte durchführen, wenn Sie die KSP- und CNG-Provider zum ersten Mal verwenden oder nachdem Sie in Ihrem Cluster HSMs hinzugefügt oder entfernt haben. Auf diese Weise kann AWS CloudHSM Daten synchronisieren und unter allen HSMs im Cluster Konsistenz gewährleisten.

Schritt 1: Beenden des AWS CloudHSM-Clients

Halten Sie den AWS CloudHSM-Client an, bevor Sie die von den Providern verwendeten Konfigurationsdateien aktualisieren. Ist der Client bereits angehalten, hat das Ausführen des Anhaltebefehls keine Auswirkungen.

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

  Verwenden Sie Strg + C in dem Befehlsfenster, in dem Sie den AWS CloudHSM-Client gestartet haben.

Schritt 2: Aktualisieren der AWS CloudHSM-Konfigurationsdateien

Dieser Schritt verwendet den Parameter `-a` des Konfigurationstools, um die IP-Adresse der Elastic Network-Schnittstelle (ENI) eines der HSMs im Cluster in der Konfigurationsdatei einzufügen.

```
C:\Program Files\Amazon\CloudHSM configure.exe -a <HSM ENI IP>
```

Um die ENI-IP-Adresse eines HSM im Cluster zu erhalten, navigieren Sie zunächst zur AWS CloudHSM-Konsole. Wählen Sie dann Cluster und anschließend den gewünschten Cluster aus. Alternativ können Sie auch die Operation DescribeClusters, den Befehl describe-clusters oder das PowerShell-Cmdlet Get-HSM2Cluster verwenden. Geben Sie nur eine ENI-IP-Adresse ein. Es ist unerheblich, welche ENI-IP-Adresse Sie verwenden.

Schritt 3: Starten des AWS CloudHSM-Clients

Als Nächstes starten Sie den AWS CloudHSM-Client oder starten diesen neu. Wenn der AWS CloudHSM-Client startet, wird die ENI IP-Adresse aus der Konfigurationsdatei des Clients verwendet, um den Cluster abzufragen. Anschließend werden die ENI IP-Adressen aller HSMs im Cluster zur Clusterinformationsdatei hinzugefügt.

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Überprüfen der KSP- und CNG-Anbieter

Sie können einen der folgenden Befehle verwenden, um zu ermitteln, welche Provider im System installiert sind. Die Befehle listen die registrierten KSP- und CNG-Provider auf. Der AWS CloudHSM-Client muss zum betreffenden Zeitpunkt nicht ausgeführt werden.

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -enum
```

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -enum
```

Um zu überprüfen, ob die KSP- und CNG-Anbieter auf Ihrer Windows-Server-EC2-Instance installiert sind, sollten Sie die folgenden Einträge in der Liste sehen:

```
Cavium CNG Provider
Cavium Key Storage Provider
```

Wenn der CNG-Provider fehlt, führen Sie folgenden Befehl aus.

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -register
```

Wenn der KSP-Anbieter fehlt, führen Sie folgenden Befehl aus.

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -register
```

Windows AWS CloudHSM-Voraussetzungen

Bevor Sie den Windows AWS CloudHSM-Client starten und die KSP- und CNG-Provider nutzen können, müssen Sie die Anmeldeinformationen für das HSM in Ihrem System einrichten. Sie können Anmeldeinformationen entweder über die Windows-Anmeldeinformationsverwaltung oder die Systemumgebungsvariable festlegen. Wir empfehlen, zum Speichern der Anmeldeinformationen die Windows-Anmeldeinformationsverwaltung zu verwenden. Diese Option ist ab AWS CloudHSM-Client-Version 2.0.4 verfügbar. Die Verwendung von Umgebungsvariablen ist einfacher einzurichten, aber weniger sicher als die Windows-Anmeldeinformationsverwaltung.

Windows-Anmeldeinformationsverwaltung

Sie können entweder das Dienstprogramm `set_cloudhsm_credentials` oder die Schnittstelle der Windows-Anmeldeinformationsverwaltung verwenden.

- Verwenden des Dienstprogramms **`set_cloudhsm_credentials`**:

  Das Dienstprogramm `set_cloudhsm_credentials` ist in Ihrem Windows-Installationsprogramm enthalten. Sie können dieses Dienstprogramm verwenden, um HSM-Anmeldeinformationen bequem an die Windows-Anmeldeinformationsverwaltung zu übergeben. Wenn Sie dieses Dienstprogramm aus der Quelle kompilieren möchten, können Sie den Python-Code verwenden, der im Installationsprogramm enthalten ist.

  1. Rufen Sie den Ordner `C:\Program Files\Amazon\CloudHSM\tools\` auf.
  2. Führen Sie die Datei `set_cloudhsm_credentials.exe` mit den Parametern für CU-Benutzername und -Passwort aus.

     ```
     set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
     ```

- Verwenden der Schnittstelle der Anmeldeinformationsverwaltung:

  Sie können die Schnittstelle der Anmeldeinformationsverwaltung verwenden, um Ihre Anmeldeinformationen manuell zu verwalten.

  1. Um die Anmeldeinformationsverwaltung zu öffnen, geben Sie `credential manager` in das Suchfeld in der Taskleiste ein und wählen Sie Anmeldeinformationsverwaltung aus.
  2. Wählen Sie Windows-Anmeldeinformationen aus, um Windows-Anmeldeinformationen zu verwalten.
  3. Wählen Sie Generische Anmeldeinformationen hinzufügen aus und füllen Sie die Details wie folgt aus:
     - Geben Sie unter Internet- oder Netzwerkadresse den Zielnamen als `cloudhsm_client` ein.
     - Geben Sie unter Benutzername und Passwort die CU-Anmeldeinformationen ein.
     - Klicken Sie auf OK.

Systemumgebungsvariablen

Sie können Systemumgebungsvariablen festlegen, die ein HSM und einen Crypto-Benutzer (Crypto User, CU) für Ihre Windows-Anwendung identifizieren. Sie können mit dem setx-Befehl Systemumgebungsvariablen konfigurieren oder Sie können programmgesteuert oder auf der Registerkarte Advanced (Erweitert) in den System Properties (Systemeigenschaften) der Windows-Systemsteuerung permanente Systemumgebungsvariablen konfigurieren.

> **Warning**
>
> Wenn Sie Anmeldeinformationen über Systemumgebungsvariablen festlegen, ist das Passwort in Klartext auf dem System eines Benutzers verfügbar. Verwenden Sie die Windows-Anmeldeinformationsverwaltung, um dieses Problem zu beheben.

Konfigurieren Sie die folgenden Systemumgebungsvariablen:

`n3fips_password=`*`CU USERNAME`*`:`*`CU PASSWORD`

Identifiziert einen Crypto-Benutzer (CU, Crypto User) im HSM und stellt alle erforderlichen Anmeldeinformationen bereit. Ihre Anwendung führt die Authentifizierung durch und wird als dieser CU ausgeführt. Die Anwendung verfügt über die Berechtigungen dieses CU und kann nur die Schlüssel anzeigen und verwalten, die sich im Besitz dieses CU befinden und von ihm freigegeben werden. Um einen neuen CU zu erstellen, verwenden Sie createUser. Um vorhandene CUs zu suchen, verwenden Sie listUsers.

Beispiele:

```
setx /m n3fips_password test_user:password123
```

Zuordnen eines AWS CloudHSM-Schlüssels zu einem Zertifikat

Bevor Sie AWS CloudHSM-Schlüssel mit Tools von Drittanbietern wie dem SignTool von Microsoft verwenden können, müssen Sie die Metadaten des Schlüssels in den lokalen Zertifikatspeicher importieren und einem Zertifikat zuordnen. Um die Metadaten des Schlüssels zu importieren, verwenden Sie das Dienstprogramm import_key.exe, das in CloudHSM Version 3.0 und höher enthalten ist. Die folgenden Schritte enthalten zusätzliche Informationen und eine Beispielausgabe.

Schritt 1: Importieren Ihres Zertifikats

Unter Windows sollten Sie auf das Zertifikat doppelklicken können, um es in den lokalen Zertifikatspeicher zu importieren.

Wenn ein Doppelklick jedoch nicht funktioniert, verwenden Sie das Microsoft Certreq-Tool, um das Zertifikat in den Zertifikatmanager zu importieren. Beispiele:

```
certreq -accept certificatename
```

Wenn diese Aktion fehlschlägt und Sie den Fehler `Key not found` erhalten, fahren Sie mit Schritt 2 fort. Wenn das Zertifikat in Ihrem Schlüsselspeicher angezeigt wird, haben Sie die Aufgabe abgeschlossen, und es ist keine weitere Aktion erforderlich.

Schritt 2: Erfassen von Zertifikatinformationen

Wenn der vorherige Schritt nicht erfolgreich war, müssen Sie Ihren privaten Schlüssel einem Zertifikat zuordnen. Bevor Sie jedoch die Zuordnung erstellen können, müssen Sie zuerst den eindeutigen Containernamen und die Seriennummer des Zertifikats finden. Verwenden Sie ein Dienstprogramm, z. B. certutil, um die erforderlichen Zertifikatinformationen anzuzeigen. Die folgende Beispielausgabe von certutil zeigt den Containernamen und die Seriennummer.

```
================ Certificate 1 ================ Serial Number:
   72000000047f7f7a9d41851b4e000000000004Issuer: CN=Enterprise-CANotBefore: 10/8/2019
 11:50
   AM NotAfter: 11/8/2020 12:00 PMSubject: CN=www.example.com, OU=Certificate
 Management,
   O=Information Technology, L=Seattle, S=Washington, C=USNon-root CertificateCert
   Hash(sha1): 7f d8 5c 00 27 bf 37 74 3d 71 5b 54 4e c0 94 20 45 75 bc 65No key
 provider
   information Simple container name: CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
 Unique
   container name: CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
```

Schritt 3: Verknüpfen des privaten AWS CloudHSM-Schlüssels mit dem Zertifikat

Um den Schlüssel mit dem Zertifikat zu verknüpfen, müssen Sie zuerst den AWS CloudHSM Client-Daemon starten. Verwenden Sie dann import_key.exe (in CloudHSM Version 3.0 und höher enthalten), um den privaten Schlüssel dem Zertifikat zuzuordnen. Wenn Sie das Zertifikat angeben, verwenden Sie seinen einfachen Containernamen. Das folgende Beispiel zeigt den Befehl und die Antwort. Diese Aktion kopiert nur die Metadaten des Schlüssels; der Schlüssel verbleibt auf dem HSM.

```
$> import_key.exe -RSA CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c

Successfully opened Microsoft Software Key Storage Provider : 0NCryptOpenKey failed :
 80090016
```

Schritt 4: Aktualisieren des Zertifikatspeichers

Stellen Sie sicher, dass der AWS CloudHSM-Client Daemon noch läuft. Verwenden Sie dann das certutil-Verb -repairstore, um die Seriennummer des Zertifikats zu aktualisieren. Das folgende Beispiel zeigt den Befehl und die Ausgabe. Weitere Informationen zum Verb -repairstore finden Sie in der Microsoft-Dokumentation.

```
C:\Program Files\Amazon\CloudHSM>certutil -f -csp "Cavium Key Storage Provider"-
repairstore my "72000000047f7f7a9d41851b4e000000000004"
my "Personal"
================ Certificate 1 ================
Serial Number: 72000000047f7f7a9d41851b4e000000000004
Issuer: CN=Enterprise-CA
NotBefore: 10/8/2019 11:50 AM
NotAfter: 11/8/2020 12:00 PM
Subject: CN=www.example.com, OU=Certificate Management, O=Information Technology,
 L=Seattle, S=Washington, C=US
Non-root CertificateCert Hash(sha1): 7f d8 5c 00 27 bf 37 74 3d 71 5b 54 4e c0 94 20 45
 75 bc 65
SDK Version: 3.0
Key Container = CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
Provider = Cavium Key Storage ProviderPrivate key is NOT exportableEncryption test
 passedCertUtil: -repairstore command completed successfully.
```

Nach dem Aktualisieren der Zertifikatsseriennummer können Sie dieses Zertifikat und den entsprechenden privaten AWS CloudHSM-Schlüssel mit jedem Drittanbieter-Signaturprogramm unter Windows verwenden.

Codebeispiel für CNG-Anbieter

⚠ ** Nur Beispiel-Code – nicht für die Produktion vorgesehen **
Dieser Beispielcode dient lediglich der Veranschaulichung. Führen Sie den Code nicht in der Produktionsumgebung aus.

Das folgende Beispiel zeigt, wie Sie die registrierten Crypto-Anbieter auf Ihrem System auflisten, um den mit dem CloudHSM-Client für Windows installierten CNG-Anbieter zu finden. Das Beispiel zeigt

außerdem, wie ein asymmetrisches Schlüsselpaar erstellt und zum Signieren von Daten verwendet wird.

> **Important**
>
> Bevor Sie dieses Beispiel ausführen, müssen Sie die HSM-Anmeldeinformationen wie in den Voraussetzungen beschrieben einrichten. Details hierzu finden Sie unter Windows AWS CloudHSM-Voraussetzungen.

```
// CloudHsmCngExampleConsole.cpp : Console application that demonstrates CNG
 capabilities.
// This example contains the following functions.
//
//   VerifyProvider()          - Enumerate the registered providers and retrieve Cavium
 KSP and CNG providers.
//   GenerateKeyPair()         - Create an RSA key pair.
//   SignData()                - Sign and verify data.
//

#include "stdafx.h"
#include <Windows.h>

#ifndef NT_SUCCESS
#define NT_SUCCESS(Status) ((NTSTATUS)(Status) >= 0)
#endif

#define CAVIUM_CNG_PROVIDER L"Cavium CNG Provider"
#define CAVIUM_KEYSTORE_PROVIDER L"Cavium Key Storage Provider"

// Enumerate the registered providers and determine whether the Cavium CNG provider
// and the Cavium KSP provider exist.
//
bool VerifyProvider()
{
  NTSTATUS status;
  ULONG cbBuffer = 0;
  PCRYPT_PROVIDERS pBuffer = NULL;
  bool foundCng = false;
  bool foundKeystore = false;
```

```
  // Retrieve information about the registered providers.
  //   cbBuffer - the size, in bytes, of the buffer pointed to by pBuffer.
  //   pBuffer - pointer to a buffer that contains a CRYPT_PROVIDERS structure.
  status = BCryptEnumRegisteredProviders(&cbBuffer, &pBuffer);

  // If registered providers exist, enumerate them and determine whether the
  // Cavium CNG provider and Cavium KSP provider have been registered.
  if (NT_SUCCESS(status))
  {
    if (pBuffer != NULL)
    {
      for (ULONG i = 0; i < pBuffer->cProviders; i++)
      {
        // Determine whether the Cavium CNG provider exists.
        if (wcscmp(CAVIUM_CNG_PROVIDER, pBuffer->rgpszProviders[i]) == 0)
        {
          printf("Found %S\n", CAVIUM_CNG_PROVIDER);
          foundCng = true;
        }

        // Determine whether the Cavium KSP provider exists.
        else if (wcscmp(CAVIUM_KEYSTORE_PROVIDER, pBuffer->rgpszProviders[i]) == 0)
        {
          printf("Found %S\n", CAVIUM_KEYSTORE_PROVIDER);
          foundKeystore = true;
        }
      }
    }
  }
  else
  {
    printf("BCryptEnumRegisteredProviders failed with error code 0x%08x\n", status);
  }

  // Free memory allocated for the CRYPT_PROVIDERS structure.
  if (NULL != pBuffer)
  {
    BCryptFreeBuffer(pBuffer);
  }

  return foundCng == foundKeystore == true;
}
```

```
// Generate an asymmetric key pair. As used here, this example generates an RSA key
 pair
// and returns a handle. The handle is used in subsequent operations that use the key
 pair.
// The key material is not available.
//
// The key pair is used in the SignData function.
//
NTSTATUS GenerateKeyPair(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_KEY_HANDLE *hKey)
{
  NTSTATUS status;

  // Generate the key pair.
  status = BCryptGenerateKeyPair(hAlgorithm, hKey, 2048, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptGenerateKeyPair failed with code 0x%08x\n", status);
    return status;
  }

  // Finalize the key pair. The public/private key pair cannot be used until this
  // function is called.
  status = BCryptFinalizeKeyPair(*hKey, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptFinalizeKeyPair failed with code 0x%08x\n", status);
    return status;
  }

  return status;
}

// Sign and verify data using the RSA key pair. The data in this function is hardcoded
// and is for example purposes only.
//
NTSTATUS SignData(BCRYPT_KEY_HANDLE hKey)
{
  NTSTATUS status;
  PBYTE sig;
  ULONG sigLen;
  ULONG resLen;
  BCRYPT_PKCS1_PADDING_INFO pInfo;

  // Hardcode the data to be signed (for demonstration purposes only).
```

```
  PBYTE message = (PBYTE)"d83e7716bed8a20343d8dc6845e57447";
  ULONG messageLen = strlen((char*)message);

  // Retrieve the size of the buffer needed for the signature.
  status = BCryptSignHash(hKey, NULL, message, messageLen, NULL, 0, &sigLen, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptSignHash failed with code 0x%08x\n", status);
    return status;
  }

  // Allocate a buffer for the signature.
  sig = (PBYTE)HeapAlloc(GetProcessHeap(), 0, sigLen);
  if (sig == NULL)
  {
    return -1;
  }

  // Use the SHA256 algorithm to create padding information.
  pInfo.pszAlgId = BCRYPT_SHA256_ALGORITHM;

  // Create a signature.
  status = BCryptSignHash(hKey, &pInfo, message, messageLen, sig, sigLen, &resLen,
BCRYPT_PAD_PKCS1);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptSignHash failed with code 0x%08x\n", status);
    return status;
  }

  // Verify the signature.
  status = BCryptVerifySignature(hKey, &pInfo, message, messageLen, sig, sigLen,
BCRYPT_PAD_PKCS1);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptVerifySignature failed with code 0x%08x\n", status);
    return status;
  }

  // Free the memory allocated for the signature.
  if (sig != NULL)
  {
    HeapFree(GetProcessHeap(), 0, sig);
    sig = NULL;
```

```
  }

  return 0;
}

// Main function.
//
int main()
{
  NTSTATUS status;
  BCRYPT_ALG_HANDLE hRsaAlg;
  BCRYPT_KEY_HANDLE hKey = NULL;

  // Enumerate the registered providers.
  printf("Searching for Cavium providers...\n");
  if (VerifyProvider() == false) {
    printf("Could not find the CNG and Keystore providers\n");
    return 1;
  }

  // Get the RSA algorithm provider from the Cavium CNG provider.
  printf("Opening RSA algorithm\n");
  status = BCryptOpenAlgorithmProvider(&hRsaAlg, BCRYPT_RSA_ALGORITHM,
 CAVIUM_CNG_PROVIDER, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptOpenAlgorithmProvider RSA failed with code 0x%08x\n", status);
    return status;
  }

  // Generate an asymmetric key pair using the RSA algorithm.
  printf("Generating RSA Keypair\n");
  GenerateKeyPair(hRsaAlg, &hKey);
  if (hKey == NULL)
  {
    printf("Invalid key handle returned\n");
    return 0;
  }
  printf("Done!\n");

  // Sign and verify [hardcoded] data using the RSA key pair.
  printf("Sign/Verify data with key\n");
  SignData(hKey);
  printf("Done!\n");
```

```
  // Remove the key handle from memory.
  status = BCryptDestroyKey(hKey);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptDestroyKey failed with code 0x%08x\n", status);
    return status;
  }

  // Close the RSA algorithm provider.
  status = BCryptCloseAlgorithmProvider(hRsaAlg, NULL);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptCloseAlgorithmProvider RSA failed with code 0x%08x\n", status);
    return status;
  }

  return 0;
}
```

Vorheriges Client-SDK (Client-SDK 3)

AWS CloudHSM umfasst zwei Hauptversionen des Client-SDK:

- Client-SDK 5: Dies ist unser neuestes und standardmäßiges Client-SDK. Informationen zu den Vorteilen und Nutzen, die es bietet, finden Sie unter Vorteile von Client-SDK 5.
- Client-SDK 3: Dies ist unser älteres Client-SDK. Es enthält einen vollständigen Satz von Komponenten für die Kompatibilität von plattform- und sprachbasierten Anwendungen und Verwaltungstools.

Anweisungen zur Migration von Client-SDK 3 zu Client-SDK 5 finden Sie unter Migration von Client-SDK 3 zu Client-SDK 5.

Die Dokumentation zum Client-SDK 3 ist in diesem Thema aufgeführt.

Zum Herunterladen siehe Downloads.

Überprüfen Sie die Version Ihres Client-SDK

Amazon Linux

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

Amazon Linux 2

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

CentOS 6

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

CentOS 7

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

CentOS 8

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

RHEL 6

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

RHEL 7

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

RHEL 8

Verwenden Sie den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

Ubuntu 16.04 LTS

Verwenden Sie den folgenden Befehl:

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 18.04 LTS

Verwenden Sie den folgenden Befehl:

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 20.04 LTS

Verwenden Sie den folgenden Befehl:

```
apt list --installed | grep ^cloudhsm
```

Windows Server

Verwenden Sie den folgenden Befehl:

```
wmic product get name,version
```

Vergleich der Client-SDK-Komponenten

Zusätzlich zu den Befehlszeilentools enthält das Client-SDK 3 Komponenten, die es ermöglichen, kryptografische Operationen von verschiedenen plattform- oder sprachbasierten Anwendungen auf das HSM auszulagern. Client-SDK 5 hat Parität mit Client-SDK 3, unterstützt jedoch noch keine CNG- und KSP-Anbieter. In der folgenden Tabelle wird die Verfügbarkeit der Komponenten in Client-SDK 3 und Client-SDK 5 verglichen.

Komponente	Client-SDK 5	Client-SDK 3
PKCS-#11-Bibliothek	Ja	Ja
JCE-Anbieter	Ja	Ja
OpenSSL Dynamic Engine	Ja	Ja
CNG- und KSP-Anbieter		Ja
CloudHSM-Verwaltungsdienstprogramm (CMU) [1]	Ja	Ja
Schlüsselverwaltungsdienstprogramm (KMU) [1]	Ja	Ja
Configure-Tool	Ja	Ja

[1] CMU- und KMU-Komponenten sind in der CloudHSM-CLI mit Client SDK 5 enthalten.

Themen

Unterstützte Plattformen vom Client-SDK 3

Das Client-SDK 3 erfordert einen Client-Daemon und bietet Befehlszeilentools, darunter das CloudHSM Management Utility (CMU), das Key Management Utility (KMU) und das Configure Tool.

Die Basisunterstützung ist für jede Version des AWS CloudHSM-Client-SDK unterschiedlich. -BDie Plattformunterstützung für Komponenten in einem SDK entspricht typischerweise in der Regel der Basisunterstützung, jedoch nicht immer. Um die Plattformunterstützung für eine bestimmte Komponente zu ermitteln, stellen Sie zunächst sicher, dass die gewünschte Plattform im Basisbereich

des SDK angezeigt wird, und suchen Sie dann im Komponentenabschnitt nach Ausnahmen oder anderen relevanten Informationen.

Die Plattformunterstützung ändert sich im Laufe der Zeit. Frühere Versionen des CloudHSM-Client-SDK unterstützen möglicherweise nicht alle hier aufgeführten Betriebssysteme. Ermitteln Sie anhand der Versionshinweise die Betriebssystemunterstützung für frühere Versionen des CloudHSM-Client-SDK. Weitere Informationen finden Sie unter Downloads für das AWS CloudHSM Client-SDK.

AWS CloudHSM unterstützt nur 64-Bit-Betriebssysteme.

Inhalt

Linux-Support

- Amazon Linux
- Amazon Linux 2
- CentOS 6.10+ [2]
- CentOS 7.3+
- CentOS 8 [1,4]
- Red Hat Enterprise Linux (RHEL) 6.10+ [2]
- Red Hat Enterprise Linux (RHEL) 7.3+
- Red Hat Enterprise Linux (RHEL) 8 [1]
- Ubuntu 16.04 LTS [3]
- Ubuntu 18.04 LTS [1]

[1] Keine Unterstützung für OpenSSL Dynamic Engine. Weitere Informationen finden Sie unter OpenSSL Dynamic Engine.

[2] Keine Unterstützung für Client-SDK 3.3.0 und höher.

[3] SDK 3.4 ist die letzte unterstützte Version auf Ubuntu 16.04.

[4] SDK 3.4 ist die letzte unterstützte Version auf CentOS 8.3+.

Windows-Unterstützung

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Unterstützung für Komponenten

PKCS #11-Bibliothek

Die PKCS #11-Bibliothek ist eine reine Linux-Komponente, die der Linux-Basisunterstützung entspricht. Weitere Informationen finden Sie unter the section called “Linux-Support”.

JCE-Anbieter

Der JCE-Anbieter ist eine reine Linux-Komponente, die der Linux-Basisunterstützung entspricht. Weitere Informationen finden Sie unter the section called “Linux-Support”.

- Erfordert OpenJDK 1.8

OpenSSL Dynamic Engine

Die OpenSSL Dynamic Engine ist eine reine Linux-Komponente, die nicht der Linux-Basisunterstützung entspricht. Sehen Sie sich die folgenden Ausnahmen an.

- Benötigt OpenSSL 1.0.2[f+]

Nicht unterstützte Plattformen:

- CentOS 8
- Red Hat Enterprise Linux (RHEL) 8

- Ubuntu 18.04 LTS

Diese Plattformen werden mit einer Version von OpenSSL ausgeliefert, die mit OpenSSL Dynamic Engine für Client-SDK 3 nicht kompatibel ist. AWS CloudHSM unterstützt diese Plattformen mit OpenSSL Dynamic Engine für Client-SDK 5.

CNG- und KSP-Anbieter

Bei den CNG- und KSP-Anbietern handelt es sich um eine reine Windows-Komponente, die der Windows-Basisunterstützung entspricht. Weitere Informationen finden Sie unter Windows-Unterstützung.

Upgrade des Client-SDK 3 auf Linux

Bei AWS CloudHSM-Client-SDK 3.1 und höher müssen die Version des Client-Daemon und aller Komponenten, die Sie installieren, für das Upgrade übereinstimmen. Für alle Linux-basierten Systeme müssen Sie einen einzigen Befehl verwenden, um den Client-Daemon stapelweise mit derselben Version der PKCS-#11-Bibliothek, dem Java Cryptographic Extension-Anbieter (JCE) oder der OpenSSL Dynamic Engine zu aktualisieren. Diese Anforderung gilt nicht für Windows-basierte Systeme, da die Binärdateien für die CNG- und KSP-Anbieter bereits im Paket des Client-Daemon enthalten sind.

So ermitteln Sie die Client-Daemon-Version

- Verwenden Sie auf einem Red Hat-basierten Linux-System (einschließlich Amazon Linux und CentOS) den folgenden Befehl:

```
rpm -qa | grep ^cloudhsm
```

- Verwenden Sie auf einem Debian-basierten Linux-System den folgenden Befehl:

```
apt list --installed | grep ^cloudhsm
```

- Verwenden Sie auf einem Windows-System den folgenden Befehl:

```
wmic product get name,version
```

Themen

Voraussetzungen

Laden Sie die neueste Version des AWS CloudHSM-Client-Daemon herunter und wählen Sie Ihre Komponenten aus.

> **Note**
>
> Sie müssen nicht alle Komponenten installieren. Für jede installierte Komponente müssen Sie diese Komponente aktualisieren, damit sie mit der Version des Client-Daemon übereinstimmt.

Neuester Linux-Client-Daemon

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

Neueste PKCS-#11-Bibliothek

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-pkcs11_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

Neuester OpenSSL Dynamic Engine

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-dyn_latest_amd64.deb
```

Neuester JCE-Anbieter

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-jce-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-jce_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-jce_latest_u18.04_amd64.deb
```

Schritt 1: Beenden des Client-Daemon

Verwenden Sie den folgenden Befehl, um den Client-Daemon zu beenden.

Amazon Linux

```
$ sudo stop cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

Schritt 2: Upgrade des Client-SDK

Der folgende Befehl zeigt die Syntax, die zum Aktualisieren des Client-Daemon und der Komponenten erforderlich ist. Entfernen Sie vor dem Ausführen des Befehls alle Komponenten, die Sie nicht aktualisieren möchten.

Amazon Linux

```
$ sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm \
              <./cloudhsm-client-pkcs11-latest.el6.x86_64.rpm> \
              <./cloudhsm-client-dyn-latest.el6.x86_64.rpm> \
              <./cloudhsm-client-jce-latest.el6.x86_64.rpm>
```

Amazon Linux 2

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
```

```
                <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

CentOS 7

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

CentOS 8

```
$ sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el8.x86_64.rpm>
```

RHEL 7

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

RHEL 8

```
$ sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el8.x86_64.rpm>
```

Ubuntu 16.04 LTS

```
$ sudo apt install ./cloudhsm-client_latest_amd64.deb \
                <cloudhsm-client-pkcs11_latest_amd64.deb> \
                <cloudhsm-client-dyn_latest_amd64.deb> \
                <cloudhsm-client-jce_latest_amd64.deb>
```

Ubuntu 18.04 LTS

```
$ sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb \
                <cloudhsm-client-pkcs11_latest_amd64.deb> \
```

```
                <cloudhsm-client-jce_latest_amd64.deb>
```

Schritt 3: Starten des Client-Daemon

Verwenden Sie den folgenden Befehl, um den Client-Daemon zu starten.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

PKCS #11-Bibliothek für Client-SDK 3

PKCS #11 ist ein Standard für die Ausführung kryptografischer Operationen auf Hardware-Sicherheitsmodulen (HSM).

Informationen zu Bootstrapping finden Sie unter Verbinden mit dem Cluster.

Themen

- Installation der Bibliothek Client-SDK 3 für PKCS #11
- Authentifizierung bei der PKCS #11-Bibliothek (Client-SDK 3)
- Unterstützte Schlüsseltypen (Client-SDK 3)
- Unterstützte Mechanismen (Client-SDK 3)
- Unterstützte API-Operationen (Client-SDK 3)
- Unterstützte Schlüsselattribute (Client-SDK 3)
- Codebeispiele für die PKCS #11-Bibliothek (Client-SDK 3)

Installation der Bibliothek Client-SDK 3 für PKCS #11

Voraussetzungen für das Client-SDK 3

Für die PKCS #11-Bibliothek ist der AWS CloudHSM-Client erforderlich.

Wenn Sie den AWS CloudHSM-Client nicht installiert und konfiguriert haben, führen Sie jetzt die Schritte unter Installieren des Clients (Linux) durch. Nachdem Sie den Client installiert und konfiguriert haben, verwenden Sie den folgenden Befehl ein, um ihn zu starten.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl cloudhsm-client start
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

CentOS 8

```
$ sudo systemctl cloudhsm-client start
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 8

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Installieren Sie die PKCS #11-Bibliothek für das Client-SDK 3

Mit dem folgenden Befehl laden Sie die PKCS #11-Bibliothek herunter und installieren sie.

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-pkcs11_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-pkcs11_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

- Wenn auf der EC2-Instance, auf der Sie die PKCS #11-Bibliothek installiert haben, keine anderen Komponenten aus dem Client-SDK 3 installiert sind, müssen Sie das Client-SDK 3 booten. Sie müssen dies nur einmal auf jeder Instance mit einer Komponente aus dem Client-SDK 3 tun.
- Die PKCS #11-Bibliotheksdateien finden Sie an den folgenden Speicherorten:

  Linux-Binärdateien, Konfigurationsskripten, Zertifikate und Protokolldateien:

  ```
  /opt/cloudhsm/lib
  ```

Authentifizierung bei der PKCS #11-Bibliothek (Client-SDK 3)

Wenn Sie die PKCS #11-Bibliothek verwenden, läuft Ihre Anwendung als ein bestimmter Crypto-Benutzer (CU) in Ihren HSMs. Ihre Anwendung kann nur die Schlüssel anzeigen und verwalten, die der CU besitzt und freigibt. Sie können eine vorhandene CU in Ihren HSMs verwenden oder

eine neue CU erstellen. Weitere Informationen zum Verwalten von CUs finden Sie unter HSM-Benutzer mit CloudHSM-CLI verwalten und HSM-Benutzer mit CloudHSM Management Utility (CMU) verwalten.

Um den CU für die PKCS #11-Bibliothek anzugeben, verwenden Sie den Pin-Parameter der PKCS #11-Funktion C_Login. Für AWS CloudHSM hat der Pin-Parameter das folgende Format:

```
<CU_user_name>:<password>
```

Mit dem folgenden Befehl wird beispielsweise dem PKCS #11-Bibliothek-Pin der CU mit dem Benutzernamen `CryptoUser` und dem Passwort `CUPassword123!` zugewiesen.

```
CryptoUser:CUPassword123!
```

Unterstützte Schlüsseltypen (Client-SDK 3)

Die PKCS #11-Bibliothek unterstützt die folgenden Schlüsseltypen.

Schlüsseltyp	Beschreibung
RSA	Generieren Sie RSA-Schlüssel mit 2048 bis 4096 Bit, in Schritten von 256 Bit.
EC	Generieren Sie Schlüssel mit den Kurven secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384) und secp521r1 (P-521).
AES	Generieren der 128-, 192- und 256-Bit-AES-Schlüssel.
DES3 (Triple DES)	Generieren Sie 192-Bit-DES3-Schlüssel. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.
GENERIC_SECRET	Generieren Sie generische Geheimnisse mit 1 bis 64 Byte.

- [1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Unterstützte Mechanismen (Client-SDK 3)

Die PKCS #11-Bibliothek unterstützt die folgenden Algorithmen:

- Verschlüsselung und Entschlüsselung – AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP und RSA-PKCS
- Signieren und Verifizieren – RSA, HMAC und ECDSA; mit und ohne Hashing
- Hash/Digest – SHA1, SHA224, SHA256, SHA384 und SHA512
- Key Wrap – AES Key Wrap[4], AES-GCM, RSA-AES und RSA-OAEP
- Schlüsselableitung – ECDH,[5] SP800-108 CTR KDF

Die Funktionstabelle des PKCS #11-Bibliotheksmechanismus

Die PKCS #11-Bibliothek entspricht Version 2.40 der PKCS #11-Spezifikation. Um eine kryptographische Funktion mit PKCS#11 aufzurufen, rufen Sie eine Funktion mit einem bestimmten Mechanismus auf. Die folgende Tabelle fasst die Kombinationen von Funktionen und Mechanismen zusammen, die von AWS CloudHSM unterstützt werden.

Tabelle zur Interpretation der unterstützten PKCS #11-Mechanismen und -Funktionen

Ein ✓ zeigt an, dass AWS CloudHSM den Mechanismus für die Funktion unterstützt. Wir unterstützen nicht alle möglichen Funktionen, die in der PKCS #11 Spezifikation aufgeführt sind. Ein ✖ zeigt an, dass AWS CloudHSM den Mechanismus für die angegebene Funktion noch nicht unterstützt, obwohl der PKCS #11-Standard dies erlaubt. Leere Zellen zeigen an, dass der PKCS #11-Standard den Mechanismus für die angegebene Funktion nicht unterstützt.

Unterstützte PKCS #11-Bibliotheksmechanismen und -funktionen

Mechanismus	Funktionen							
		Schlüssel generieren oder	Signieren und prüfen	SR und VR	Digest	Verschlüsseln/ Ents	Schlüssel ableiten	Wrap und UnWrap

Mechanismus	Funktionen						
	Schlüsselpaar			chlüsseln			
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						
CKM_RSA_X9_31_KEY_PAIR_GEN	✓[2]						
CKM_RSA_X_509		✓			✓		
CKM_RSA_PKCS siehe Hinweis 8		✓[1]	✖		✓[1]		✓[1]
CKM_RSA_PKCS_OAEP					✓[1]		✓[6]
CKM_SHA1_RSA_PKCS		✓[3.2]					
CKM_SHA224_RSA_PKCS		✓[3.2]					
CKM_SHA256_RSA_PKCS		✓[3.2]					
CKM_SHA384_RSA_PKCS		✓[2,3.2]					

Mechanismus	Funktionen						
CKM_SHA512_RSA_PKCS		✓[3.2]					
CKM_RSA_PKCS_PSS		✓[1]					
CKM_SHA1_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA224_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA256_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA384_RSA_PKCS_PSS		✓[2,3.2]					
CKM_SHA512_RSA_PKCS_PSS		✓[3.2]					
CKM_EC_KEY_PAIR_GEN	✓						
CKM_ECDSA		✓[1]					
CKM_ECDSA_SHA1		✓[3.2]					

Mechanismus	Funktionen						
CKM_ECDSA_SHA224		✓[3.2]					
CKM_ECDSA_SHA256		✓[3.2]					
CKM_ECDSA_SHA384		✓[3.2]					
CKM_ECDSA_SHA512		✓[3.2]					
CKM_ECDH1_DERIVE						✓[5]	
CKM_SP800_108_COUNTER_KDF						✓	
CKM_GENERIC_SECRET_KEY_GEN	✓						
CKM_AES_KEY_GEN	✓						
CKM_AES_ECB					✓		✖
CKM_AES_CTR					✓		✖
CKM_AES_CBC					✓[3.3]		✖

Mechanismus	Funktionen						
CKM_AES_CBC_PAD					✓		✖
CKM_DES3_KEY_GEN siehe Hinweis 8	✓						
CKM_DES3_CBC siehe Hinweis 8					✓[3.3]		✖
CKM_DES3_CBC_PAD siehe Hinweis 8					✓		✖
CKM_DES3_ECB siehe Hinweis 8					✓		✖
CKM_AES_GCM					✓[3.3, 4]		✓[7.1]
CKM_CLOUDHSM_AES_GCM					✓[7.1]		✓[7.1]
CKM_SHA_1				✓[3.1]			

Mechanismus	Funktionen						
CKM_SHA_1_HMAC		✓[3.3]					
CKM_SHA224				✓[3.1]			
CKM_SHA224_HMAC		✓[3.3]					
CKM_SHA256				✓[3.1]			
CKM_SHA256_HMAC		✓[3.3]					
CKM_SHA384				✓[3.1]			
CKM_SHA384_HMAC		✓[3.3]					
CKM_SHA512				✓[3.1]			
CKM_SHA512_HMAC		✓[3.3]					
CKM_RSA_AES_KEY_WRAP							✓
CKM_AES_KEY_WRAP							✓

Mechanismus	Funktionen						
CKM_AES_KEY_WRAP_PAD							✓
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD							✓[7.1]
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD							✓[7.1]
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD							✓[7.1]

Anmerkungen zum Mechanismus

- [1] Nur Single-Part-Operationen.
- [2] Der Mechanismus ist funktionell identisch mit dem CKM_RSA_PKCS_KEY_PAIR_GEN-Mechanismus, bietet aber stärkere Garantien für die p- und q-Generierung.
- [3.1] AWS CloudHSM geht je nach Client-SDK unterschiedlich an das Hashing heran. Beim Client-SDK 3 hängt der Ort, an dem wir das Hashing durchführen, von der Datengröße ab und davon, ob Sie einteilige oder mehrteilige Operationen verwenden.

  Einteilige Operationen im Client-SDK 3

  In Tabelle 3.1 ist die maximale Datensatzgröße für jeden Mechanismus für das Client-SDK 3 aufgeführt. Der gesamte Hash wird innerhalb des HSM berechnet. Keine Unterstützung für Datengrößen über 16 KB.

Tabelle 3.1, Maximale Datensatzgröße für einteilige Operationen

Mechanismus	Maximale Datengröße
CKM_SHA_1	16296
CKM_SHA224	16264
CKM_SHA256	16296
CKM_SHA384	16232
CKM_SHA512	16232

Mehrteilige Operationen Client-SDK 3

Support für Datengrößen über 16 KB, aber die Datengröße bestimmt, wo das Hashing stattfindet. Datenpuffer mit weniger als 16 KB werden innerhalb des HSM gehasht. Puffer zwischen 16 KB und der maximalen Datengröße für Ihr System werden lokal in der Software gehasht. Denken Sie daran: Für Hash-Funktionen sind keine kryptografischen Geheimnisse erforderlich, sodass Sie sie sicher außerhalb des HSM berechnen können.

- [3.2] AWS CloudHSM geht je nach Client-SDK unterschiedlich an das Hashing heran. Beim Client-SDK 3 hängt der Ort, an dem wir das Hashing durchführen, von der Datengröße ab und davon, ob Sie einteilige oder mehrteilige Operationen verwenden.

  Einteilige Operationen Client-SDK 3

  In Tabelle 3.2 ist die maximale Datensatzgröße für jeden Mechanismus für das Client-SDK 3 aufgeführt. Keine Unterstützung für Datengrößen über 16 KB.

  Tabelle 3.2, Maximale Datensatzgröße für einteilige Operationen

Mechanismus	Maximale Datengröße
CKM_SHA1_RSA_PKCS	16296
CKM_SHA224_RSA_PKCS	16264
CKM_SHA256_RSA_PKCS	16296

Mechanismus	Maximale Datengröße
CKM_SHA384_RSA_PKCS	16232
CKM_SHA512_RSA_PKCS	16232
CKM_SHA1_RSA_PKCS_PSS	16296
CKM_SHA224_RSA_PKCS_PSS	16264
CKM_SHA256_RSA_PKCS_PSS	16296
CKM_SHA384_RSA_PKCS_PSS	16232
CKM_SHA512_RSA_PKCS_PSS	16232
CKM_ECDSA_SHA1	16296
CKM_ECDSA_SHA224	16264
CKM_ECDSA_SHA256	16296
CKM_ECDSA_SHA384	16232
CKM_ECDSA_SHA512	16232

Mehrteilige Operationen Client-SDK 3

Support für Datengrößen über 16 KB, aber die Datengröße bestimmt, wo das Hashing stattfindet. Datenpuffer mit weniger als 16 KB werden innerhalb des HSM gehasht. Puffer zwischen 16 KB und der maximalen Datengröße für Ihr System werden lokal in der Software gehasht. Denken Sie daran: Für Hash-Funktionen sind keine kryptografischen Geheimnisse erforderlich, sodass Sie sie sicher außerhalb des HSM berechnen können.

- [3.3] Wenn der Datenpuffer die maximale Datengröße überschreitet, führt die Operation bei der Bearbeitung von Daten mittels eines der folgenden Mechanismen zu einem Fehler. Für diese Mechanismen muss die gesamte Datenverarbeitung innerhalb des HSM erfolgen. Die folgende Tabelle listet die maximale Datengröße für jeden Mechanismus auf:

Tabelle 3.3, Maximale Datensatzgröße

Mechanismus	Maximale Datengröße
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288
CKM_SHA384_HMAC	16224
CKM_SHA512_HMAC	16224
CKM_AES_CBC	16272
CKM_AES_GCM	16224
CKM_CLOUDHSM_AES_GCM	16224
CKM_DES3_CBC	16280

- [4] Bei AES-GCM-Verschlüsselungen akzeptiert HSM keine Initialisierungsvektor (IV)-Daten von der Anwendung. Sie müssen einen erzeugten IV verwenden. Das vom HSM bereitgestellte 12 Byte IV wird in den referenzierten Speicherbereich geschrieben, auf den das pIV-Element der von Ihnen bereitgestellten CK_GCM_PARAMS-Parameterstruktur zeigt. Um Missverständnissen vorzubeugen: Das PKCS#11-SDK in Version 1.1.1 und höher erzwingt, dass pIV auf einen auf Null zurückgesetzten Puffer verweist, wenn die AES-GCM-Verschlüsselung initialisiert wird.
- [5] Nur Client-SDK 3. 5 Der Mechanismus ist zur Unterstützung von SSL/TLS-Auslagerungsfällen implementiert und wird nur teilweise innerhalb des HSM ausgeführt. Bevor Sie diesen Mechanismus verwenden, lesen Sie „Problem: ECDH-Schlüsselableitung wird nur teilweise innerhalb des HSM ausgeführt“ in Bekannte Probleme für den PKCS#11-Bibliothek. CKM_ECDH1_DERIVE unterstützt nicht die secp521r1 (P-521) Kurve.
- [6] Die folgenden CK_MECHANISM_TYPE und CK_RSA_PKCS_MGF_TYPE werden als CK_RSA_PKCS_OAEP_PARAMS für CKM_RSA_PKCS_OAEP unterstützt:
  - CKM_SHA_1 mit CKG_MGF1_SHA1
  - CKM_SHA224 mit CKG_MGF1_SHA224
  - CKM_SHA256 mit CKG_MGF1_SHA256

  - CKM_SHA384 mit CKM_MGF1_SHA384
  - CKM_SHA512 mit CKM_MGF1_SHA512
- [7.1] Vom Anbieter definierter Mechanismus. Um die von anbieterdefinierten CloudHSM-Mechanismen zu verwenden, müssen PKCS #11 -Anwendungen während der Kompilierung /opt/cloudhsm/include/pkcs11t.h enthalten.

  CKM_CLOUDHSM_AES_GCM: Dieser proprietäre Mechanismus ist eine programmatisch sicherere Alternative zum Standard CKM_AES_GCM. Er stellt die vom HSM generierte IV dem Chiffretext voran, anstatt sie zurück in die CK_GCM_PARAMS- Struktur zu schreiben, die während der Chiffrierinitialisierung bereitgestellt wird. Sie können diesen Mechanismus mit C_Encrypt-, C_WrapKey-, C_Decrypt- und C_UnwrapKey-Funktionen verwenden. Bei Verwendung dieses Mechanismus muss die pIV-Variable in der CK_GCM_PARAMS-Struktur auf NULL gesetzt werden. Wenn Sie diesen Mechanismus mit C_Decrypt und C_UnwrapKey verwenden, wird erwartet, dass der IV dem Verschlüsselungstext vorangestellt wird, der entpackt werden soll.

  CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: AES-Verschlüsselung mit PKCS #5 Padding

  CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: AES-Verschlüsselung mit Zero Padding

  Weitere Informationen zum AES-Schlüssel-Wrapping finden Sie unter AES Key Wrapping.
- [8] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Unterstützte API-Operationen (Client-SDK 3)

Die PKCS #11-Bibliothek unterstützt die folgenden PKCS #11 API Operationen.

- C_CloseAllSessions
- C_CloseSession
- C_CreateObject
- C_Decrypt
- C_DecryptFinal
- C_DecryptInit
- C_DecryptUpdate
- C_DeriveKey
- C_DestroyObject

- C_Digest
- C_DigestFinal
- C_DigestInit
- C_DigestUpdate
- C_Encrypt
- C_EncryptFinal
- C_EncryptInit
- C_EncryptUpdate
- C_Finalize
- C_FindObjects
- C_FindObjectsFinal
- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList
- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_OpenSession
- C_Sign
- C_SignFinal

- `C_SignInit`
- `C_SignRecover` (Nur Unterstützung für Client-SDK 3)
- `C_SignRecoverInit` (Nur Unterstützung für Client-SDK 3)
- `C_SignUpdate`
- `C_UnWrapKey`
- `C_Verify`
- `C_VerifyFinal`
- `C_VerifyInit`
- `C_VerifyRecover` (Nur Unterstützung für Client-SDK 3)
- `C_VerifyRecoverInit` (Nur Unterstützung für Client-SDK 3)
- `C_VerifyUpdate`
- `C_WrapKey`

Unterstützte Schlüsselattribute (Client-SDK 3)

Bei einem Schlüsselobjekt kann es sich um einen öffentlichen, privaten oder geheimen Schlüssel handeln. Für ein Schlüsselobjekt genehmigte Aktionen werden durch Attribute angegeben. Attribute werden bei der Erstellung des Schlüsselobjekts bestimmt. Wenn Sie die PKCS #11-Bibliothek verwenden, weisen wir Standardwerte zu, wie sie im PKCS #11-Standard festgelegt sind.

AWS CloudHSM unterstützt nicht alle in der PKCS #11-Spezifikation genannten Attribute. Wir erfüllen die Spezifikation für alle von uns unterstützten Attribute. Diese Attribute sind in den entsprechenden Tabellen aufgeführt.

Kryptografische Funktionen wie `C_CreateObject`, `C_GenerateKey`, `C_GenerateKeyPair`, `C_UnwrapKey` und `C_DeriveKey` zum Erstellen, Ändern oder Kopieren von Objekten benötigen eine Attributvorlage für einen ihrer Parameter. Weitere Informationen zum Übertragen von Attributvorlagen während der Erstellung von Objekten finden Sie im Beispiel unter Generieren von Schlüsseln über die PKCS #11-Bibliothek.

Interpretation der PKCS #11-Bibliotheksattribut-Tabelle

Die PKCS #11-Bibliothek-Tabelle enthält eine Liste von Attributen, die je nach Schlüsseltyp unterschiedlich sind. Sie gibt an, ob ein vorhandenes Attribut für einen bestimmten Schlüsseltyp bei Verwendung einer spezifischen kryptografischen Funktion mit AWS CloudHSM unterstützt wird.

Legende:

- ✓ gibt an, dass CloudHSM das Attribut für den spezifischen Schlüsseltyp unterstützt.
- ✖ gibt an, dass CloudHSM das Attribut für den spezifischen Schlüsseltyp nicht unterstützt.
- R gibt an, dass der Attributwert für den spezifischen Schlüsseltyp als schreibgeschützt festgelegt ist.
- S zeigt an, dass das Attribut von `GetAttributeValue` nicht gelesen werden kann, da hierbei Groß- und Kleinschreibung beachtet werden muss.
- Eine leere Zelle in der Spalte „Standardwert“ gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

GenerateKeyPair

Attribut	Schlüsseltyp				Default Value (Standard wert)
	EC privat	EC öffentlich	RSA privat	RSA öffentlich	
`CKA_CLASS`	✓	✓	✓	✓	
`CKA_KEY_TYPE`	✓	✓	✓	✓	
`CKA_LABEL`	✓	✓	✓	✓	
`CKA_ID`	✓	✓	✓	✓	
`CKA_LOCAL`	R	R	R	R	True
`CKA_TOKEN`	✓	✓	✓	✓	Falsch

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✖	✓	✖	✓	False
CKA_DECRYPT	✓	✖	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓	✓	✓	✓	Wahr
CKA_SIGN	✓	✖	✓	✖	False
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	
CKA_VERIFY	✖	✓	✖	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	
CKA_WRAP	✖	✓	✖	✓	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_TRUSTED	✖	✓	✖	✓	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	False
CKA_UNWRAP	✓	✖	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓[2]	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_PRIME_1	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✓[2]	
CKA_EC_PARAMS	✖	✓[2]	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	

Attribut	Schlüsseltyp				Default Value (Standard wert)
CKA_CHECK_VALUE		R	R	R	R	

GenerateKey

Attribut	Schlüsseltyp			Default Value (Standard wert)
	AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	Falsch
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✓	✓	✖	False

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓	✓	✓	Wahr
CKA_SIGN	✓	✓	✓	Wahr
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	Wahr
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✓	✓	✖	
CKA_TRUSTED	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	False

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_UNWRAP	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✓	✖	
CKA_SENSITIVE	✓	✓	✓	Wahr
CKA_ALWAYS_SENSITIVE	✖	✖	✖	
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

CreateObject

Attribut	Schlüsseltyp							Default Value (Standardwert)
	EC privat	EC öffentlich	RSA privat	RSA öffentlich	AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖	False
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖	False

Attribut	Schlüsseltyp							Default Value (Standardwert)
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	Wahr
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	✖	✖	✖	False
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	✖	✖	✖	
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	False

Attribut	Schlüsseltyp							Default Value (Standardwert)
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	R	R	R	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	R	R	R	
CKA_MODULUS	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	

Attribut	Schlüsseltyp							Default Value (Standardwert)
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✓[2]	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_EC_PARAMS	✓[2]	✓[2]	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✓[2]	✖	✖	✖	✖	✖	
CKA_VALUE	✓[2]	✖	✖	✖	✓[2]	✓[2]	✓[2]	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

Attribut	Schlüsseltyp					Default Value (Standard wert)
	EC privat	RSA privat	AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✖	✖	✓	✓	✖	False
CKA_DECRYPT	✖	✓	✓	✓	✖	False

Attribut	Schlüsseltyp					Default Value (Standard wert)
CKA_DERIVE	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓	✓	✓	✓	✓	Wahr
CKA_SIGN	✓	✓	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✓[3]	✖	✖	✖	False
CKA_VERIFY	✖	✖	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	False
CKA_UNWRAP	✖	✓	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	✓	✓	Wahr
CKA_EXTRACTABLE	✓	✓	✓	✓	✓	True

Attribut	Schlüsseltyp					Default Value (Standard wert)
CKA_NEVER_EXTRACTABLE		R	R	R	R	R	
CKA_ALWAYS_SENSITIVE		R	R	R	R	R	
CKA_MODULUS		✖	✖	✖	✖	✖	
CKA_MODULUS_BITS		✖	✖	✖	✖	✖	
CKA_PRIME_1		✖	✖	✖	✖	✖	
CKA_PRIME_2		✖	✖	✖	✖	✖	
CKA_COEFFICIENT		✖	✖	✖	✖	✖	
CKA_EXPONENT_1		✖	✖	✖	✖	✖	
CKA_EXPONENT_2		✖	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT		✖	✖	✖	✖	✖	

Attribut	Schlüsseltyp					Default Value (Standard wert)
CKA_PUBLIC_EXPONENT		✖	✖	✖	✖	✖	
CKA_EC_PARAMS		✖	✖	✖	✖	✖	
CKA_EC_POINT		✖	✖	✖	✖	✖	
CKA_VALUE		✖	✖	✖	✖	✖	
CKA_VALUE_LEN		✖	✖	✖	✖	✖	
CKA_CHECK_VALUE		R	R	R	R	R	

DeriveKey

Attribut	Schlüsseltyp				Default Value (Standard wert)
		AES	DES3	Allgemeiner geheimer Schlüssel	
CKA_CLASS		✓[2]	✓[2]	✓[2]	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	Falsch
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	Wahr
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	Wahr
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	Wahr
CKA_SIGN	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	False

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	Wahr
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	

Attribut	Schlüsseltyp			Default Value (Standard wert)
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

Attribut	Schlüsseltyp						
	EC privat	EC öffentlich	RSA privat	RSA öffentlich	AES	DES3	Allgemeiner geheimer Schlüssel
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓

Attribut	Schlüsseltyp						
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓

Attribut	Schlüsseltyp						
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_NEVER_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R
CKA_MODULUS	✖	✖	✓	✓	✖	✖	✖
CKA_MODULUS_BITS	✖	✖	✖	✓	✖	✖	✖
CKA_PRIME_1	✖	✖	S	✖	✖	✖	✖
CKA_PRIME_2	✖	✖	S	✖	✖	✖	✖
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖

Attribut	Schlüsseltyp						
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓[2]	✓[2]	✓[2]
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

Anmerkungen zu Attributen

- [1] Dieses Attribut wird teilweise von der Firmware unterstützt und muss explizit nur auf den Standardwert festgelegt werden.
- [2] Obligatorisches Attribut.
- [3] Nur Client-SDK 3. Das Attribut `CKA_SIGN_RECOVER` ist vom Attribut `CKA_SIGN` abgeleitet. Falls dieses festgelegt wird, kann es ausschließlich auf den Wert festgelegt werden, der auch für `CKA_SIGN` eingestellt ist. Falls es nicht festgelegt wird, leitet es den Standardwert von `CKA_SIGN` ab. Da CloudHSM nur RSA-basierte, wiederherstellbare Signaturmechanismen unterstützt, ist dieses Attribut derzeit nur auf öffentliche RSA-Schlüssel anwendbar.
- [4] Nur Client-SDK 3. Das Attribut `CKA_VERIFY_RECOVER` ist vom Attribut `CKA_VERIFY` abgeleitet. Falls dieses festgelegt wird, kann es ausschließlich auf den Wert festgelegt werden, der auch für `CKA_VERIFY` eingestellt ist. Falls es nicht festgelegt wird, leitet es den Standardwert von `CKA_VERIFY` ab. Da CloudHSM nur RSA-basierte, wiederherstellbare Signaturmechanismen unterstützt, ist dieses Attribut derzeit nur auf öffentliche RSA-Schlüssel anwendbar.

Ändern von Attributen

Einige Attribute eines Objekts können geändert werden, nachdem das Objekt erstellt wurde, andere dagegen nicht. Um Attribute zu ändern, verwenden Sie den Befehl setAttribute aus cloudhsm_mgmt_util. Sie können auch eine Liste der Attribute und der Konstanten, die sie repräsentieren, ableiten, indem Sie den Befehl listAttribute aus cloudhsm_mgmt_util verwenden.

Die folgende Liste zeigt Attribute, die nach der Erstellung eines Objekts geändert werden dürfen.

- `CKA_LABEL`
- `CKA_TOKEN`

> **Note**
>
> Änderungen sind nur zum Umwandeln eines Sitzungsschlüssels in einen Token-Schlüssel zulässig. Verwenden Sie den Befehl setAttribute aus key_mgmt_util, um den Wert des Attributs zu ändern.

- `CKA_ENCRYPT`
- `CKA_DECRYPT`
- `CKA_SIGN`
- `CKA_VERIFY`
- `CKA_WRAP`
- `CKA_UNWRAP`
- `CKA_LABEL`
- `CKA_SENSITIVE`
- `CKA_DERIVE`

> **Note**
>
> Dieses Attribut unterstützt die Schlüsselableitung. Es muss für alle öffentlichen Schlüssel `False` lauten und kann nicht auf `True` festgelegt werden. Für geheime und private EC-Schlüssel kann es auf `True` oder `False` einstellt werden.

- `CKA_TRUSTED`

> **Note**
>
> Dieses Attribut kann nur vom Verschlüsselungsverantwortlichen (CO) auf `True` oder `False` festgelegt werden.

- `CKA_WRAP_WITH_TRUSTED`

> **Note**
>
> Wenden Sie dieses Attribut auf einen exportierbaren Datenschlüssel an, um anzugeben, dass Sie diesen Schlüssel nur mit Schlüsseln umschließen können, die als `CKA_TRUSTED` markiert sind. Sobald Sie `CKA_WRAP_WITH_TRUSTED` auf wahr setzen, wird das Attribut schreibgeschützt und Sie können das Attribut nicht mehr ändern oder entfernen.

Interpretieren von Fehlercodes

Das Angeben eines nicht von einem spezifischen Schlüssel unterstützten Attributs in der Vorlage führt zu einem Fehler. Die folgende Tabelle enthält die Fehlercodes, die generiert werden, wenn Sie gegen die Spezifikationen verstoßen:

Fehlercode	Beschreibung
`CKR_TEMPLATE_INCONSISTENT`	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage ein Attribut angeben, das zwar der PKCS #11-Spezifikation entspricht, jedoch nicht von CloudHSM unterstützt wird.
`CKR_ATTRIBUTE_TYPE_INVALID`	Diese Fehlermeldung erhalten Sie, wenn Sie den Wert für ein Attribut abrufen, das zwar der PKCS #11-Spezifikation entspricht, jedoch nicht von CloudHSM unterstützt wird.
`CKR_ATTRIBUTE_INCOMPLETE`	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage das obligatorische Attribut nicht angeben.

Fehlercode	Beschreibung
`CKR_ATTRIBUTE_READ_ONLY`	Diese Fehlermeldung erhalten Sie, wenn Sie in der Attributvorlage ein schreibgeschütztes Attribut angeben.

Codebeispiele für die PKCS #11-Bibliothek (Client-SDK 3)

Die Codebeispiele auf GitHub zeigen Ihnen, wie Sie grundlegende Aufgaben mit der PKCS #11-Bibliothek ausführen.

Voraussetzungen für den Beispiel-Code

Bevor Sie die Beispiele ausführen, führen Sie die folgenden Schritte aus, um Ihre Umgebung einzurichten:

- Installieren und konfigurieren Sie die PKCS #11-Bibliothek für das Client-SDK 3.
- Richten Sie einen Crypto-Benutzer (CU) ein. Ihre Anwendung verwendet dieses HSM-Konto, um die Codebeispiele auf dem HSM auszuführen.

Codebeispiele

Codebeispiele für die AWS CloudHSM Softwarebibliothek für PKCS#11 sind auf verfügbarGitHub. Dieses Repository enthält Beispiele für allgemeine Vorgänge mit PKCS #11, einschließlich Verschlüsselung, Entschlüsselung, Signieren und Verifizieren.

- Schlüssel generieren (AES, RSA, EC)
- Schlüsselattribute auflisten
- Verschlüsseln und Entschlüsseln von Daten mit AES GCM
- Verschlüsseln und Entschlüsseln von Daten mit AES_CTR
- Verschlüsseln und Entschlüsseln von Daten mit 3DES
- Signieren und Verifizieren von Daten mit RSA
- Ableiten von Schlüsseln mit HMAC KDF
- Verpacken und Entpacken von Schlüsseln mit AES und PKCS #5 Padding
- Verpacken und Entpacken von Schlüsseln mit AES ohne Padding

- Verpacken und Entpacken von Schlüsseln mit AES und Zero Padding
- Packen und Entpacken von Schlüsseln mit AES-GCM
- Schlüssel mit RSA ver- und entpacken

Installation von Client-SDK 3 für OpenSSL Dynamic Engine

Für das Client-SDK 3 ist ein Client-Daemon erforderlich, um eine Verbindung zum Cluster herzustellen. Es unterstützt:

- RSA-Schlüsselgenerierung für 2048-, 3072- und 4096-Bit-Schlüssel.
- RSA Signieren/Überprüfen.
- RSA Verschlüsseln/Entschlüsseln.
- Generierung einer Zufallszahl, die kryptografisch sicher und FIPS-validiert ist.

Themen

- Voraussetzungen für OpenSSL Dynamic Engine mit Client-SDK 3
- Installieren Sie die OpenSSL Dynamic Engine für Client-SDK 3
- OpenSSL Dynamic Engine für Client-SDK 3 verwenden

Voraussetzungen für OpenSSL Dynamic Engine mit Client-SDK 3

Informationen zur Plattformunterstützung finden Sie unter Unterstützte Plattformen vom Client-SDK 3.

Damit Sie die dynamische AWS CloudHSM-Engine für OpenSSL verwenden können, benötigen Sie den AWS CloudHSM-Client.

Der Client ist ein Daemon, der eine end-to-end verschlüsselte Kommunikation mit den HSMs in Ihrem Cluster herstellt, und die OpenSSL-Engine kommuniziert lokal mit dem Client. Einzelheiten zur Installation und Konfiguration des AWS CloudHSM-Clients finden Sie unter Installieren des Clients (Linux). Verwenden Sie dann den folgenden Befehl, um ihn zu starten.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl cloudhsm-client start
```

CentOS 6

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 6

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Installieren Sie die OpenSSL Dynamic Engine für Client-SDK 3

In den folgenden Schritten wird beschrieben, wie Sie die AWS CloudHSM Dynamic Engine für OpenSSL installieren und konfigurieren. Weitere Informationen zum Upgrade finden Sie unter Upgrade von Client-SDK 3.

Zum Installieren und Konfigurieren der OpenSSL-Engine

1. Verwenden Sie die folgenden Befehle zum Herunterladen und Installieren der OpenSSL-Engine.

   Amazon Linux

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

CentOS 6

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

RHEL 6

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/
cloudhsm-client-dyn_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-dyn_latest_amd64.deb
```

Die OpenSSL-Engine ist unter `/opt/cloudhsm/lib/libcloudhsm_openssl.so` installiert.

2. Verwenden Sie den folgenden Befehl ein, um eine Umgebungsvariable `n3fips_password` mit den Anmeldeinformationen eines Crypto-Benutzers (CU) anzulegen.

```
$ export n3fips_password=<HSM user name>:<password>
```

OpenSSL Dynamic Engine für Client-SDK 3 verwenden

Um die dynamische AWS CloudHSM-Engine für OpenSSL von einer OpenSSL-integrierten Anwendung aus zu verwenden, stellen Sie sicher, dass die Anwendung die dynamische OpenSSL-Engine namens `cloudhsm` verwendet. Die freigegebene Bibliothek für die Dynamic Engine finden Sie unter `/opt/cloudhsm/lib/libcloudhsm_openssl.so`.

Um die dynamische AWS CloudHSM-Engine für OpenSSL von der OpenSSL-Befehlszeile aus zu verwenden, geben Sie mit der Option `-engine` die dynamische OpenSSL-Engine namens `cloudhsm` an. Beispiele:

```
$ openssl s_server -cert server.crt -key server.key -engine cloudhsm
```

Client-SDK 3 für den JCE-Anbieter

Der AWS CloudHSM-JCE-Anbieter ist eine Anbieter-Implementierung, die auf dem Java Cryptographic Extension (JCE)-Anbieter-Framework basiert. JCE ermöglicht es Ihnen, kryptografische Operationen mit dem Java Development Kits (JDK) durchzuführen. In diesem Handbuch wird der AWS CloudHSM-JCE-Anbieter manchmal als JCE-Anbieter bezeichnet.

Verwenden Sie den JCE-Anbieter und das JDK, um kryptografische Operationen auf das HSM auszulagern.

Themen

Installieren und verwenden Sie den AWS CloudHSM-JCE-Anbieter für Client-SDK 3

Bevor Sie den JCE-Anbieter verwenden können, brauchen Sie den AWS CloudHSM-Client.

Der Client ist ein Daemon, der eine verschlüsselte End-to-End-Kommunikation zwischen Ihrer Client-Instanz und den HSMs in Ihrem Cluster herstellt. Der JCE-Anbieter kommuniziert lokal mit dem Client. Wenn Sie das AWS CloudHSM-Client-Paket nicht installiert und konfiguriert haben, machen Sie das jetzt. Befolgen Sie hierzu die Schritte unter Installieren des Clients (Linux). Nachdem Sie den Client installiert und konfiguriert haben, verwenden Sie den folgenden Befehl ein, um ihn zu starten.

Beachten Sie, dass der JCE-Anbieter nur unter Linux und kompatiblen Betriebssystemen unterstützt wird.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl cloudhsm-client start
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

CentOS 8

```
$ sudo systemctl cloudhsm-client start
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 8

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Themen

Den JCE-Anbieter installieren

Verwenden Sie die folgenden Befehle, um den JCE-Anbieter herunterzuladen und zu installieren. Dieser Anbieter wird nur von Linux und kompatiblen Betriebssystemen unterstützt.

> **Note**
>
> Informationen zum Upgraden finden Sie unter Upgrade von Client-SDK 3.

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-jce-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-jce_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-jce_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-jce_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-jce_latest_u18.04_amd64.deb
```

Nachdem Sie die vorangegangenen Befehle ausgeführt haben, finden Sie die folgenden JCE-Anbieter-Dateien:

- `/opt/cloudhsm/java/cloudhsm-version.jar`
- `/opt/cloudhsm/java/cloudhsm-test-version.jar`
- `/opt/cloudhsm/java/hamcrest-all-1.3.jar`
- `/opt/cloudhsm/java/junit.jar`
- `/opt/cloudhsm/java/log4j-api-2.17.1.jar`
- `/opt/cloudhsm/java/log4j-core-2.17.1.jar`
- `/opt/cloudhsm/lib/libcaviumjca.so`

Überprüfen der Installation

Führen Sie grundlegende Funktionen des HSM aus, um die Installation zu überprüfen.

Um die Installation des JCE-Anbieters zu überprüfen

1. (Optional) Falls Sie Java nicht bereits in Ihrer Umgebung installiert haben, verwenden Sie den folgenden Befehl zur Installation.

   Linux (and compatible libraries)

   ```
   $ sudo yum install java-1.8.0-openjdk
   ```

   Ubuntu

   ```
   $ sudo apt-get install openjdk-8-jre
   ```

2. Verwenden Sie die folgenden Befehle, um die erforderlichen Umgebungsvariablen festzulegen. Ersetzen Sie *<HSM user name>* und *<password>* durch die Anmeldeinformationen eines Crypto-Benutzers (CU, Crypto-User).

   ```
   $ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
   ```

   ```
   $ export HSM_PARTITION=PARTITION_1
   ```

   ```
   $ export HSM_USER=<HSM user name>
   ```

   ```
   $ export HSM_PASSWORD=<password>
   ```

3. Führen Sie den folgenden Befehl aus, um den Test der grundlegenden Funktionen zu starten. Wenn dies erfolgreich ist, sollte die Ausgabe des Befehls ähnlich wie die folgende sein.

   ```
   $ java8 -classpath "/opt/cloudhsm/java/*" org.junit.runner.JUnitCore
    TestBasicFunctionality

   JUnit version 4.11
   .2018-08-20 17:53:48,514 DEBUG [main] TestBasicFunctionality
    (TestBasicFunctionality.java:33) - Adding provider.
   2018-08-20 17:53:48,612 DEBUG [main] TestBasicFunctionality
    (TestBasicFunctionality.java:42) - Logging in.
   2018-08-20 17:53:48,612 INFO [main] cfm2.LoginManager (LoginManager.java:104) -
    Looking for credentials in HsmCredentials.properties
   ```

```
2018-08-20 17:53:48,612 INFO [main] cfm2.LoginManager (LoginManager.java:122) -
 Looking for credentials in System.properties
2018-08-20 17:53:48,613 INFO [main] cfm2.LoginManager (LoginManager.java:130) -
 Looking for credentials in System.env
 SDK Version: 2.03
2018-08-20 17:53:48,655 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:54) - Generating AES Key with key size 256.
2018-08-20 17:53:48,698 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:63) - Encrypting with AES Key.
2018-08-20 17:53:48,705 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:84) - Deleting AES Key.
2018-08-20 17:53:48,707 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:92) - Logging out.

Time: 0.205

OK (1 test)
```

Bereitstellung von Anmeldeinformationen für den JCE-Anbieter

HSMs müssen Ihre Java-Anwendung authentifizieren, bevor die Anwendung sie nutzen kann. Jede Anwendung kann eine Sitzung verwenden. HSMs authentifizieren eine Sitzung, indem sie entweder eine explizite oder implizite Anmeldemethode verwenden.

Explizite Anmeldemethode – Mit dieser Methode können Sie CloudHSM-Anmeldeinformationen direkt in der Anwendung bereitstellen. Es verwendet die `LoginManager.login()`-Methode, bei der Sie den CU-Benutzernamen, das Passwort und die HSM-Partitions-ID übergeben. Weitere Informationen zur Verwendung der expliziten Anmeldemethode finden Sie im Code-Beispiel Anmelden bei einem HSM.

Implizite Anmeldemethode – Mit dieser Methode können Sie CloudHSM-Anmeldeinformationen entweder in einer neuen Property-Datei, über Systemeigenschaften oder als Umgebungsvariablen festlegen.

- Neue Eigenschaftsdatei – Erstellen Sie eine neue Datei namens `HsmCredentials.properties` und fügen Sie sie zum `CLASSPATH` Ihrer Anwendung hinzu. Die Datei sollte Folgendes enthalten:

```
HSM_PARTITION = PARTITION_1
HSM_USER = <HSM user name>
HSM_PASSWORD = <password>
```

- Systemeigenschaften – Anmeldeinformationen über Systemeigenschaften fest, wenn Sie Ihre Anwendung ausführen. Die folgenden Beispiele zeigen zwei verschiedene Möglichkeiten:

```
$ java -DHSM_PARTITION=PARTITION_1 -DHSM_USER=<HSM user name> -
DHSM_PASSWORD=<password>
```

```
System.setProperty("HSM_PARTITION","PARTITION_1");
System.setProperty("HSM_USER","<HSM user name>");
System.setProperty("HSM_PASSWORD","<password>");
```

- Umgebungsvariablen – Anmeldeinformationen als Umgebungsvariablen festlegen.

```
$ export HSM_PARTITION=PARTITION_1
$ export HSM_USER=<HSM user name>
$ export HSM_PASSWORD=<password>
```

Anmeldeinformationen sind möglicherweise nicht verfügbar, wenn die Anwendung sie nicht bereitstellt oder wenn Sie eine Operation ausführen, bevor der HSM die Sitzung authentifiziert. In diesen Fällen sucht die CloudHSM-Softwarebibliothek für Java in der folgenden Reihenfolge nach den Anmeldeinformationen:

1. `HsmCredentials.properties`
2. Systemeigenschaften
3. Umgebungsvariablen

Fehlerbehandlung

Die Fehlerbehandlung ist bei der expliziten Anmeldung einfacher als bei der impliziten Anmeldemethode. Wenn Sie die Klasse `LoginManager` verwenden, haben Sie mehr Kontrolle darüber, wie Ihre Anwendung mit Fehlern umgeht. Bei der impliziten Anmeldemethode ist die Fehlerbehandlung schwer nachzuvollziehen, wenn die Anmeldeinformationen ungültig sind oder die HSMs Probleme bei der Authentifizierung der Sitzung haben.

Grundlagen der Schlüsselverwaltung im JCE-Anbieter

Die Grundlagen der Schlüsselverwaltung im JCE-Anbieter umfassen den Import von Schlüsseln, den Export von Schlüsseln, das Laden von Schlüsseln per Handle oder das Löschen von Schlüsseln. Weitere Informationen zur Schlüsselverwaltung finden Sie im Codebeispiel Schlüssel verwalten.

Weitere Codebeispiele für JCE-Anbieter finden Sie auch in Codebeispiele.

Unterstützte Mechanismen für Client-SDK 3

Weitere Informationen über die von AWS CloudHSM unterstützten Java Cryptography Architecture (JCA)-Schnittstellen und Engine-Klassen finden Sie in den folgenden Themen.

Themen

- Unterstützte Schlüssel
- Unterstützte Verschlüsselungen
- Unterstützte Digests
- Unterstützte Hash-basierter Nachrichtenauthentifizierungscode (HMAC)-Algorithmen
- Unterstützte Signatur/Prüf-Mechanismen
- Anmerkungen zum Mechanismus

Unterstützte Schlüssel

Mit der AWS CloudHSM-Softwarebibliothek für Java können Sie die folgenden Schlüsseltypen generieren.

- AES – 128-, 192- und 256-Bit AES-Schlüssel.
- DeSede – 92-Bit-3DES-Schlüssel. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.
- ECC-Schlüsselpaare für die NIST secp256r1- (P-256), secp384r1- (P-384) und secp256k1-Kurven (Blockchain).
- RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.

Zusätzlich zu den Standardparametern unterstützen wir die folgenden Parameter für die einzelnen generierten Schlüssel.

- Label: Eine Schlüsselbezeichnung anhand der Sie nach Schlüsseln suchen können.
- isExtractable: Gibt an, ob ein Schlüssel vom HSM exportiert werden kann.
- isPersistent: Gibt an, ob der Schlüssel nach Beenden der aktuellen Sitzung weiterhin im HSM bleibt.

> **Note**
>
> Die Java-Bibliothek Version 3.1 bietet die Möglichkeit, Parameter genauer zu spezifizieren. Weitere Informationen finden Sie unter Unterstützte Java-Attribute.

Unterstützte Verschlüsselungen

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden Kombinationen aus Algorithmen, Modi und Paddings.

Algorithmus	Mode	Padding	Hinweise
AES	CBC	`AES/CBC/NoPadding` `AES/CBC/PKCS5Padding`	Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE` .
AES	ECB	`AES/ECB/NoPadding` `AES/ECB/PKCS5Padding`	Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE` . Transformation AES verwenden.
AES	CTR	`AES/CTR/NoPadding`	Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE` .
AES	GCM	`AES/GCM/NoPadding`	Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MOD`

Algorithmus	Mode	Padding	Hinweise
			`E` , `Cipher.WR AP_MODE` und `Cipher.UN WRAP_MODE` . HSM ignoriert den Initialisierungsvektor (IV) der Anforderung während der AES-GCM-Datenversc hlüsselung und verwendet stattdess en einen selbst generierten IV. Nach Abschluss der Operation müssen Sie `Cipher.getIV()` abrufen, um den IV zu erhalten.

AESWrap	ECB	`AESWrap/ECB/ ZeroPadding`
		
		`AESWrap/ECB/ NoPadding`
		
		`AESWrap/ECB/ PKCS5Padding`
			Implementiert `Cipher.WR AP_MODE` und `Cipher.UN WRAP_MODE` . Transformation AES verwenden.

Algorithmus	Mode	Padding	Hinweise

DESede (Triple DES)	CBC	`DESede/CBC/NoPadding`
		
		`DESede/CBC/PKCS5Padding`
			Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE`.
			
			Die Schlüsselerstellungsvorgänge akzeptieren die Größen 168 oder 192 Bits. Intern verfügen alle DESede-Schlüssel jedoch über 192 Bits.
			
			Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

DESede (Triple DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE`.
			
			Die Schlüsselerstellungsvorgänge akzeptieren die Größen 168 oder 192 Bits. Intern verfügen alle DESede-Schlüssel jedoch über 192 Bits.
			
			Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

Algorithmus	Mode	Padding	Hinweise

RSA	ECB	`RSA/ECB/NoPadding`
		
		`RSA/ECB/PKCS1Padding`
			Implementiert `Cipher.ENCRYPT_MODE` und `Cipher.DECRYPT_MODE` .
			
			Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

Algorithmus	Mode	Padding	Hinweise

RSA	ECB	`RSA/ECB/OAEPPadding`
		
		`RSA/ECB/OAEPWithSHA-1ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-224ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-256ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-384ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-512ANDMGF1Padding`
			Implementiert `Cipher.ENCRYPT_MODE`, `Cipher.DECRYPT_MODE`, `Cipher.WRAP_MODE` und `Cipher.UNWRAP_MODE`.
			
			`OAEPPadding` ist OAEP mit den SHA-1 Padding-Typ.

RSAAESWrap	ECB	`OAEPPADDING`	Implementiert `Cipher.WRAP_Mode` und `Cipher.UNWRAP_MODE`.

Unterstützte Digests

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden Mitteilungs-Digests.

- `SHA-1`
- `SHA-224`
- `SHA-256`
- `SHA-384`
- `SHA-512`

> **Note**
>
> Daten mit weniger als 16 KB werden im HSM mit einem Hash-Wert versehen. Größere Daten werden lokal in der Software mit einem Hash-Wert versehen.

Unterstützte Hash-basierter Nachrichtenauthentifizierungscode (HMAC)-Algorithmen

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden HMAC-Algorithmen.

- `HmacSHA1`
- `HmacSHA224`
- `HmacSHA256`
- `HmacSHA384`
- `HmacSHA512`

Unterstützte Signatur/Prüf-Mechanismen

Die AWS CloudHSM-Software-Bibliothek für Java unterstützt die folgenden Signatur- und Verifizierungstypen.

RSA-Signaturtypen

- `NONEwithRSA`
- `SHA1withRSA`

- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- SHA1withRSA/PSS
- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS

ECDSA-Signaturtypen

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Anmerkungen zum Mechanismus

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Unterstützte Java-Schlüsselattribute für Client-SDK 3

In diesem Thema wird beschrieben, wie Sie eine proprietäre Erweiterung für die Java-Bibliothek Version 3.1 zum Festlegen von Schlüsselattributen verwenden. Verwenden Sie diese Erweiterung, um unterstützte Schlüsselattribute und ihre Werte während der folgenden Vorgänge festzulegen:

- Schlüsselgenerierung
- Schlüsselimport
- Schlüssel entpacken

> **Note**
>
> Die Erweiterung zum Festlegen benutzerdefinierter Schlüsselattribute ist eine optionale Funktion. Wenn Sie bereits über einen in der Java-Bibliothek Version 3.0 funktionierenden Code verfügen, müssen Sie diesen Code nicht ändern. Schlüssel, die Sie erstellen, enthalten weiterhin dieselben Attribute wie zuvor.

Themen

Grundlegendes zu Attributen

Mithilfe von Schlüsselattributen legen Sie fest, welche Aktionen für Schlüsselobjekte zulässig sind, einschließlich öffentlicher, privater oder geheimer Schlüssel. Schlüsselattribute und -werte definieren Sie bei der Erstellung von Schlüsselobjekten.

Die Java Cryptography Extension (JCE) gibt jedoch nicht an, wie Sie Werte für Schlüsselattribute festlegen sollten. Daher sind die meisten Aktionen standardmäßig zulässig. Im Gegensatz dazu definiert der PKCS# 11-Standard einen umfassenden Satz von Attributen mit restriktiveren Standardeinstellungen. Ab der Java-Bibliothek Version 3.1 bietet CloudHSM eine proprietäre Erweiterung, mit der Sie restriktivere Werte für häufig verwendete Attribute festlegen können.

Unterstützte Attribute

Sie können Werte für die Attribute festlegen, die in der folgenden Tabelle aufgeführt sind. Als bewährte Methode legen Sie nur Werte für Attribute fest, die Sie einschränken möchten. Wenn Sie keinen Wert angeben, verwendet CloudHSM den in der folgenden Tabelle angegebenen Standardwert. Eine leere Zelle in der Spalte „Standardwert“ gibt an, dass dem Attribut kein spezifischer Standardwert zugewiesen ist.

Attribut	Standardwert			Hinweise
	Symmetrischer Schlüssel	Öffentlicher Schlüssel im Schlüsselpaar	Privater Schlüssel im Schlüsselpaar	
CKA_TOKEN	FALSE	FALSE	FALSE	Ein permanenter Schlüssel, der über alle HSMs im Cluster repliziert und in Backups enthalten ist. CKA_TOKEN = FALSE impliziert einen Sitzungsschlüssel, der nur auf ein HSM geladen und automatisch gelöscht wird, wenn die Verbindung zum HSM unterbrochen wird.
CKA_LABEL				Eine benutzerdefinierte Zeichenfolge. Sie ermöglicht Ihnen, Schlüssel auf Ihrem HSM leicht zu identifizieren.
CKA_EXTRACTABLE	TRUE		TRUE	„True“ gibt an, dass Sie

Attribut	Standardwert			Hinweise
				diesen Schlüssel aus dem HSM exportieren können.
CKA_ENCRYPT	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel zur Verschlüsselung eines beliebige n Puffers verwenden können.
CKA_DECRYPT	TRUE		TRUE	„True“ gibt an, dass Sie den Schlüssel zur Entschlüsselung eines beliebige n Puffers verwenden können. Sie legen dies im Allgemeinen auf „FALSE“ fest für einen Schlüssel , dessen CKA_WRAP auf „True“ festgelegt ist.

Attribut	Standardwert			Hinweise
CKA_WRAP	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel zum Packen eines anderen Schlüssels verwenden können. Für private Schlüssel legen Sie dies in der Regel auf „FALSE“ fest.
CKA_UNWRAP	TRUE		TRUE	„True“ gibt an, dass Sie mit dem Schlüssel einen anderen Schlüssel entpacken (importieren) können.

Attribut	Standardwert			Hinweise
CKA_SIGN	TRUE		TRUE	„True“ gibt an, dass Sie den Schlüssel verwenden können, um einen Hashwert zu signieren. Für öffentliche und private Schlüssel, die Sie archiviert haben, ist dies im Allgemeinen auf „FALSE“ festgelegt.
CKA_VERIFY	TRUE	TRUE		„True“ gibt an, dass Sie den Schlüssel verwenden können, um eine Signatur zu überprüfen. Für private Schlüssel ist dies im Allgemeinen auf „FALSE“ festgelegt.

Attribut	Standardwert			Hinweise
CKA_PRIVATE	TRUE	TRUE	TRUE	„True“ gibt an, dass ein Benutzer erst auf den Schlüssel zugreifen darf, wenn der Benutzer authentifiziert ist. Zur Verdeutlichung: Benutzer können erst dann auf Schlüssel in CloudHSM zugreifen, wenn sie authentifiziert sind, selbst wenn dieses Attribut auf „FALSE“ gesetzt ist.

> **Note**
>
> Sie erhalten eine breitere Unterstützung für Attribute in der PKCS #11-Bibliothek. Weitere Informationen finden Sie unter Unterstützte PKCS #11-Attribute.

Festlegen von Attributen für einen Schlüssel

`CloudHsmKeyAttributesMap` ist ein Java-Map-ähnliches Objekt, mit dem Sie Attributwerte für Schlüsselobjekte festlegen können. Die Methoden für `CloudHsmKeyAttributesMap` funktionieren ähnlich den Methoden, die für die Java-Map-Manipulation verwendet werden.

Sie haben zwei Optionen, um benutzerdefinierte Werte für Attribute festzulegen:

- Verwenden Sie die in der folgenden Tabelle aufgeführten Methoden
- Verwenden Sie Builder-Muster, die später in diesem Dokument gezeigt werden

Attributzuordnungsobjekte unterstützen die folgenden Methoden zum Festlegen von Attributen:

Operation	Rückgabewert	**`CloudHSMKeyAttributesMap`** -Methode
Abrufen des Werts eines Schlüsselattributs für einen vorhandenen Schlüssel	Objekt (das den Wert enthält) oder null	get(keyAttribute)
Eingeben des Werts eines Schlüsselattributs	Der vorherige Wert, der dem Schlüsselattribut zugeordnet ist, oder null, wenn keine Zuordnung für ein Schlüsselattribut vorhanden ist	put(keyAttribute, Wert)
Eingeben von Werten für mehrere Schlüsselattribute	–	PutAll(KeyAttributesMap)
Entfernen eines Schlüssel-Wert-Paares aus der Attributzuordnung	Der vorherige Wert, der dem Schlüsselattribut zugeordnet ist, oder null, wenn keine Zuordnung für ein Schlüsselattribut vorhanden ist	remove(keyAttribute)

> **Note**
>
> Alle Attribute, die Sie nicht explizit bestimmen, werden auf die Standardwerte festgelegt, die in der vorherigen Tabelle in the section called “Unterstützte Attribute” aufgeführt sind.

Beispiel für Builder-Muster

Entwickler finden es im Allgemeinen bequemer, Klassen über das Builder-Muster zu verwenden. Beispiele:

```
import com.amazonaws.cloudhsm.CloudHsmKeyAttributes;
import com.amazonaws.cloudhsm.CloudHsmKeyAttributesMap;
import com.amazonaws.cloudhsm.CloudHsmKeyPairAttributesMap;

CloudHsmKeyAttributesMap keyAttributesSessionDecryptionKey =
   new CloudHsmKeyAttributesMap.Builder()
      .put(CloudHsmKeyAttributes.CKA_LABEL, "ExtractableSessionKeyEncryptDecrypt")
      .put(CloudHsmKeyAttributes.CKA_WRAP, false)
      .put(CloudHsmKeyAttributes.CKA_UNWRAP, false)
      .put(CloudHsmKeyAttributes.CKA_SIGN, false)
      .put(CloudHsmKeyAttributes.CKA_VERIFY, false)
      .build();

CloudHsmKeyAttributesMap keyAttributesTokenWrappingKey =
   new CloudHsmKeyAttributesMap.Builder()
      .put(CloudHsmKeyAttributes.CKA_LABEL, "TokenWrappingKey")
      .put(CloudHsmKeyAttributes.CKA_TOKEN, true)
      .put(CloudHsmKeyAttributes.CKA_ENCRYPT, false)
      .put(CloudHsmKeyAttributes.CKA_DECRYPT, false)
      .put(CloudHsmKeyAttributes.CKA_SIGN, false)
      .put(CloudHsmKeyAttributes.CKA_VERIFY, false)
      .build();
```

Entwickler können auch vordefinierte Attributsätze verwenden, um bewährte Methoden in Schlüsselvorlagen leicht durchzusetzen. Beispiel:

```
//best practice template for wrapping keys

CloudHsmKeyAttributesMap commonKeyAttrs = new CloudHsmKeyAttributesMap.Builder()
    .put(CloudHsmKeyAttributes.CKA_EXTRACTABLE, false)
    .put(CloudHsmKeyAttributes.CKA_DECRYPT, false)
    .build();

// initialize a new instance of CloudHsmKeyAttributesMap by copying commonKeyAttrs
// but with an appropriate label

CloudHsmKeyAttributesMap firstKeyAttrs = new CloudHsmKeyAttributesMap(commonKeyAttrs);
firstKeyAttrs.put(CloudHsmKeyAttributes.CKA_LABEL, "key label");

// alternatively, putAll() will overwrite existing values to enforce conformance

CloudHsmKeyAttributesMap secondKeyAttrs = new CloudHsmKeyAttributesMap();
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_DECRYPT, true);
```

```
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_ENCRYPT, true);
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_LABEL, “safe wrapping key”);
secondKeyAttrs.putAll(commonKeyAttrs); // will overwrite CKA_DECRYPT to be FALSE
```

Festlegen von Attributen für ein Schlüsselpaar

Verwenden Sie die Java-Klasse `CloudHsmKeyPairAttributesMap`, um Schlüsselattribute für ein Schlüsselpaar zu verarbeiten. `CloudHsmKeyPairAttributesMap` fasst zwei `CloudHsmKeyAttributesMap`-Objekte zusammen, eines für einen öffentlichen Schlüssel und eines für einen privaten Schlüssel.

Um einzelne Attribute für den öffentlichen und den privaten Schlüssel separat festzulegen, können Sie die `put()`-Methode für das entsprechende `CloudHsmKeyAttributes`-Zuordnungsobjekt für diesen Schlüssel verwenden. Verwenden Sie die `getPublic()`-Methode, um die Attributzuordnung für den öffentlichen Schlüssel abzurufen, und die `getPrivate()`-Methode, um die Attributzuordnung für den privaten Schlüssel abzurufen. Geben Sie den Wert mehrerer Schlüsselattribute für öffentliche und private Schlüsselpaare zusammen ein, indem Sie die `putAll()`-Methode für eine Attributzuordnung von Schlüsselpaaren als Argument verwenden.

Beispiel für Builder-Muster

Entwickler finden es im Allgemeinen bequemer, Schlüsselattribute über das Builder-Muster festzulegen. Beispiele:

```
import com.amazonaws.cloudhsm.CloudHsmKeyAttributes;
import com.amazonaws.cloudhsm.CloudHsmKeyAttributesMap;
import com.amazonaws.cloudhsm.CloudHsmKeyPairAttributesMap;

//specify attributes up-front
CloudHsmKeyAttributesMap keyAttributes =
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_SIGN, false)
        .put(CloudHsmKeyAttributes.CKA_LABEL, "PublicCertSerial12345")
        .build();

CloudHsmKeyPairAttributesMap keyPairAttributes =
    new CloudHsmKeyPairAttributesMap.Builder()
        .withPublic(keyAttributes)
        .withPrivate(
            new CloudHsmKeyAttributesMap.Builder() //or specify them inline
                .put(CloudHsmKeyAttributes.CKA_LABEL, "PrivateCertSerial12345")
```

```
                .put (CloudHSMKeyAttributes.CKA_WRAP, FALSE)
                .build()
        )
        .build();
```

> **Note**
>
> Weitere Informationen zu dieser proprietären Erweiterung finden Sie im Javadoc-Archiv und im Beispiel auf GitHub. Um Javadoc zu erkunden, laden Sie das Archiv herunter und erweitern Sie es.

Zusammenführung

Gehen Sie folgendermaßen vor, um Schlüsselattribute für Ihre Schlüsselvorgänge festzulegen:

1. Instanziieren Sie `CloudHsmKeyAttributesMap` für symmetrische Schlüssel oder `CloudHsmKeyPairAttributesMap` für Schlüsselpaare.
2. Definieren Sie das Attributobjekt aus Schritt 1 mit den erforderlichen Schlüsselattributen und -werten.
3. Instanziieren Sie eine `Cavium*ParameterSpec`-Klasse, die Ihrem spezifischen Schlüsseltyp entspricht, und übergeben Sie dieses konfigurierte Attributeobjekt an seinen Konstruktor.
4. Übergeben Sie dieses `Cavium*ParameterSpec`-Objekt an eine entsprechende Crypto-Klasse oder -Methode.

Als Referenz enthält die folgende Tabelle die `Cavium*ParameterSpec`-Klassen und -Methoden, die benutzerdefinierte Schlüsselattribute unterstützen.

Schlüsseltyp	Parameter-Spezifikationsklasse	Beispiel-Konstruktoren
Basisklasse	`CaviumKeyGenAlgorithmParameterSpec`	`CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)`

Schlüsseltyp	Parameter-Spezifikationsklasse	Beispiel-Konstruktoren
DES	`CaviumDESKeyGenParameterSpec`	`CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap key AttributesMap)`
RSA	`CaviumRSAKeyGenParameterSpec`	`CaviumRSAKeyGenParameterSpec(int keysize, BigInteger publicExponent, CloudHsmKeyPairAttributesMap keyPairAttributesMap)`
Secret	`CaviumGenericSecretKeyGenParameterSpec`	`CaviumGenericSecretKeyGenParameterSpec(int size, CloudHsmKeyAttributesMap key AttributesMap)`
AES	`CaviumAESKeyGenParameterSpec`	`CaviumAESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap key AttributesMap)`

Schlüsseltyp	Parameter-Spezifikationsklasse	Beispiel-Konstruktoren
EC	`CaviumECGenParameterSpec`	`CaviumECGenParameterSpec(String stdName, CloudHsmKeyPairAttributesMap keyPairAttributesMap)`

Beispielcode: Generieren und Packen eines Schlüssels

Diese kurzen Codebeispiele veranschaulichen die Schritte für zwei verschiedene Vorgänge: Schlüsselgenerierung und Schlüsselverpackung:

```
// Set up the desired key attributes

KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec keyAttributes = new CaviumAESKeyGenParameterSpec(
    256,
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_LABEL, "MyPersistentAESKey")
        .put(CloudHsmKeyAttributes.CKA_EXTRACTABLE, true)
        .put(CloudHsmKeyAttributes.CKA_TOKEN, true)
        .build()
);

// Assume we already have a handle to the myWrappingKey
// Assume we already have the wrappedBytes to unwrap

// Unwrap a key using Custom Key Attributes

CaviumUnwrapParameterSpec unwrapSpec = new
 CaviumUnwrapParameterSpec(myInitializationVector, keyAttributes);

Cipher unwrapCipher = Cipher.getInstance("AESWrap", "Cavium");
unwrapCipher.init(Cipher.UNWRAP_MODE, myWrappingKey, unwrapSpec);
Key unwrappedKey = unwrapCipher.unwrap(wrappedBytes, "AES", Cipher.SECRET_KEY);
```

Codebeispiele für die AWS CloudHSM-Softwarebibliothek für Java für Client-SDK 3

Voraussetzungen

Bevor Sie die Beispiele ausführen, müssen Sie die Umgebung einrichten:

- Installieren und konfigurieren Sie den Java Cryptographic Extension (JCE)-Anbieter und das AWS CloudHSM-Client-Paket.
- Legen Sie einen gültigen HSM-Benutzernahmen und ein Passwort fest. Crypto-Benutzer (CU)-Berechtigungen sind ausreichend für diese Aufgaben. Ihre Anwendung verwendet diese Anmeldeinformationen, um sich für die einzelnen Beispiele beim HSM anzumelden.
- Entscheiden Sie, wie Anmeldeinformationen für den JCE-Anbieter bereitgestellt werden sollen.

Codebeispiele

Die folgenden Codebeispiele zeigen Ihnen, wie Sie den AWS CloudHSMJCE-Anbieter verwenden, um grundlegende Aufgaben auszuführen. Weitere Beispiele finden Sie auf GitHub.

- Anmeldung bei einem HSM
- Verwalten von Schlüsseln
- Generieren eines AES-Schlüssels
- Verschlüsseln und Entschlüsseln mit AES-GCM
- Verschlüsseln und Entschlüsseln mit AES-CTR
- Mit D3DES-ECB verschlüsseln und entschlüsseln siehe Hinweis 1
- Packen und Entpacken von Schlüsseln mit AES-GCM
- Verpacken und Entpacken von Schlüsseln mit AES
- Schlüssel mit RSA ver- und entpacken
- Unterstützte Schlüsselattribute verwenden
- Schlüssel im Schlüsselspeicher aufzählen
- Verwenden des CloudHSM-Schlüsselspeichers
- Signieren von Nachrichten in einem Multi-Thread-Beispiel
- Signieren und Verifizieren mit EC-Schlüsseln

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Verwenden der AWS CloudHSM-KeyStore-Java-Klasse für Client-SDK 3

Die AWS CloudHSM `KeyStore`-Klasse bietet einen speziellen PKCS12-Schlüsselspeicher, der den Zugriff auf AWS CloudHSM-Schlüssel über Anwendungen wie keytool und jarsigner ermöglicht. Dieser Schlüsselspeicher kann Zertifikate zusammen mit Ihren Schlüsseldaten speichern und sie mit den in AWS CloudHSM gespeicherten Schlüsseldaten korrelieren.

> **Note**
>
> Da Zertifikate öffentliche Informationen sind, und zur Maximierung der Speicherkapazität für kryptografische Schlüssel unterstützt AWS CloudHSM das Speichern von Zertifikaten auf HSMs nicht.

Die AWS CloudHSM `KeyStore`-Klasse implementiert das `KeyStore` Service Provider Interface (SPI) der Java Cryptography Extension (JCE). Weitere Informationen zur Verwendung von `KeyStore` finden Sie unter Class KeyStore.

Auswählen des passenden Schlüsselspeichers

Der AWS CloudHSM Java Cryptographic Extension (JCE)-Anbieter wird mit einem standardmäßigen, schreibgeschützten Pass-Through-Schlüsselspeicher geliefert, der alle Transaktionen an das HSM weitergibt. Dieser Standard-Schlüsselspeicher unterscheidet sich vom speziellen AWS CloudHSM KeyStore. In den meisten Situationen erhalten Sie eine bessere Laufzeitleistung und mehr Durchsatz, wenn Sie den Standard verwenden. Sie sollten den AWS CloudHSM KeyStore nur für Anwendungen verwenden, bei denen Sie zusätzlich zum Übergeben von Schlüsselvorgängen an das HSM Unterstützung für Zertifikate und zertifikatbasierte Vorgänge benötigen.

Obwohl beide Schlüsselspeicher den JCE-Anbieter für Operationen verwenden, sind sie unabhängige Entitäten und tauschen keine Informationen miteinander aus.

Laden Sie den Standard-Schlüsselspeicher für Ihre Java-Anwendung wie folgt:

```
KeyStore ks = KeyStore.getInstance("Cavium");
```

Laden Sie den CloudHSM KeyStore wie folgt:

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Initialisierung von AWS CloudHSM KeyStore

Melden Sie sich beim AWS CloudHSM KeyStore auf die gleiche Weise an, mit der Sie sich beim JCE-Anbieter anmelden. Sie können entweder Umgebungsvariablen oder die Systemeigenschaftendatei verwenden, und Sie sollten sich anmelden, bevor Sie den CloudHSM KeyStore verwenden. Ein Beispiel für die Anmeldung bei einem HSM mit dem JCE-Anbieter finden Sie unter Anmeldung an HSM.

Falls gewünscht, können Sie ein Passwort angeben, um die lokale PKCS12-Datei zu verschlüsseln, die Schlüsselspeicherdaten enthält. Wenn Sie den AWS CloudHSM Schlüsselspeicher erstellen, legen Sie das Passwort fest und geben es an, wenn Sie die Load-, Set- und Get-Methoden verwenden.

Instanziieren Sie ein neues CloudHSM KeyStore-Objekt wie folgt:

```
ks.load(null, null);
```

Schreiben Sie Schlüsselspeicherdaten mit der `store`-Methode in eine Datei. Von da an können Sie den vorhandenen Schlüsselspeicher mit der `load`-Methode mit der Quelldatei und dem Passwort wie folgt laden:

```
ks.load(inputStream, password);
```

Verwenden von AWS CloudHSM KeyStore

Ein CloudHSM KeyStore-Objekt wird in der Regel über eine Drittanbieter-Anwendung wie jarsigner oder keytoolverwendet. Sie können auch direkt mit Code auf das Objekt zugreifen.

AWS CloudHSM KeyStore entspricht der JCE Class KeyStore-Spezifikation und bietet die folgenden Funktionen.

- `load`

  Lädt den Schlüsselspeicher aus dem angegebenen Eingabe-Stream. Wenn beim Speichern des Schlüsselspeichers ein Passwort festgelegt wurde, muss dasselbe Passwort angegeben werden, damit das Laden erfolgreich ist. Setzen Sie beide Parameter auf null, um einen neuen leeren Schlüsselspeicher zu initialisieren.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
ks.load(inputStream, password);
```

- `aliases`

  Gibt eine Aufzählung der Aliasnamen aller Einträge in der angegebenen Schlüsselspeicherinstanz zurück. Die Ergebnisse umfassen Objekte, die lokal in der PKCS12-Datei gespeichert sind, sowie Objekte, die sich auf dem HSM befinden.

  Beispiel-Code:

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();)
 {
  String label = entry.nextElement();
  System.out.println(label);
 }
```

- `ContainsAlias`

  Gibt „true“ zurück, wenn der Schlüsselspeicher Zugriff auf mindestens ein Objekt mit dem angegebenen Alias hat. Der Schlüsselspeicher prüft lokal in der PKCS12-Datei gespeicherte Objekte und Objekte auf dem HSM.

- `DeleteEntry`

  Löscht einen Zertifikateintrag aus der lokalen PKCS12-Datei. Das Löschen von in einem HSM gespeicherten Schlüsseldaten wird mit dem AWS CloudHSM KeyStore nicht unterstützt. Sie können Schlüssel mit dem Key_mgmt_util-Tool von CloudHSM löschen.

- `GetCertificate`

  Gibt das Zertifikat zurück, das einem Alias zugeordnet ist, falls verfügbar. Wenn der Alias nicht existiert oder auf ein Objekt verweist, das kein Zertifikat ist, gibt die Funktion NULL zurück.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias)
```

- `GetCertificateAlias`

  Gibt den Namen (Alias) des ersten Schlüsselspeichereintrags zurück, dessen Daten mit dem angegebenen Zertifikat übereinstimmen.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert)
```

- `GetCertificateChain`

  Gibt die Zertifikatkette zurück, die dem angegebenen Alias zugeordnet ist. Wenn der Alias nicht existiert oder auf ein Objekt verweist, das kein Zertifikat ist, gibt die Funktion NULL zurück.

- `GetCreationDate`

  Gibt das Erstellungsdatum des durch den angegebenen Alias identifizierten Eintrags zurück. Wenn kein Erstellungsdatum verfügbar ist, gibt die Funktion das Datum zurück, an dem das Zertifikat gültig wurde.

- `GetKey`

  GetKey wird an das HSM übergeben und gibt ein Schlüsselobjekt zurück, das dem angegebenen Label entspricht. Da `getKey` direkt das HSM abfragt, kann es für jeden Schlüssel am HSM verwendet werden, unabhängig davon, ob er vom KeyStore generiert wurde.

  ```
  Key key = ks.getKey(keyLabel, null);
  ```

- `IsCertificateEntry`

  Überprüft, ob der Eintrag mit dem angegebenen Alias einen Zertifikateintrag darstellt.

- `IsKeyEntry`

  Überprüft, ob der Eintrag mit dem angegebenen Alias einen Schlüsseleintrag darstellt. Die Aktion durchsucht sowohl die PKCS12-Datei, als auch das HSM nach dem Alias.

- `SetCertificateEntry`

  Weist dem angegebenen Alias das angegebene Zertifikat zu. Wenn der angegebene Alias bereits verwendet wird, um einen Schlüssel oder ein Zertifikat zu identifizieren, wird ein `KeyStoreException` ausgelöst. Sie können JCE-Code verwenden, um das Schlüsselobjekt abzurufen und dann die KeyStore `SetKeyEntry`-Methode verwenden, um das Zertifikat dem Schlüssel zuzuordnen.

- `SetKeyEntry` mit `byte[]`-Schlüssel

  Diese API wird derzeit vom Client-SDK 3 nicht unterstützt.

- `SetKeyEntry` mit `Key`-Objekt

Weist den angegebenen Schlüssel dem angegebenen Alias zu und speichert ihn im HSM. Wenn das `Key`-Objekt nicht vom Typ `CaviumKey` ist, wird der Schlüssel als extrahierbarer Sitzungsschlüssel in das HSM importiert.

Wenn das `Key`-Objekt vom Typ `PrivateKey` ist, muss es von einer entsprechenden Zertifikatkette begleitet werden.

Wenn der Alias bereits vorhanden ist, wird ein `SetKeyEntry`- Aufruf ausgelöst, und `KeyStoreException` verhindert, dass der Schlüssel überschrieben wird. Wenn der Schlüssel überschrieben werden muss, verwenden Sie hierfür KMU oder JCE.

- `EngineSize`

  Gibt die Anzahl der Einträge im Schlüsselspeicher zurück.

- `Store`

  Speichert den Schlüsselspeicher im angegebenen Ausgabe-Stream als PKCS12-Datei und sichert ihn mit dem angegebenen Passwort. Darüber hinaus bleiben alle geladenen Schlüssel (die über `setKey`-Aufrufe gesetzt werden) bestehen.

Integrieren von Drittanbieter-Anwendungen mit AWS CloudHSM

Einige der Anwendungsfälle für AWS CloudHSM umfassen die Integration von Drittanbieter-Softwareanwendungen mit dem HSM in Ihrem AWS CloudHSM-Cluster. Durch die Integration von Drittanbieter-Software mit AWS CloudHSM können Sie eine Vielzahl von sicherheitsbezogenen Zielen erreichen. In den folgenden Themen wird beschrieben, wie Sie einige dieser Ziele umsetzen.

Themen

Verbessern Sie die Sicherheit Ihres Webservers mit SSL/TLS-Offload in AWS CloudHSM

Webserver und ihre Clients (Webbrowser) können die Protokolle Secure Sockets Layer (SSL) oder Transport Layer Security (TLS) verwenden, um die Identität des Webservers zu bestätigen und eine sichere Verbindung herzustellen, die Webseiten oder andere Daten über das Internet sendet und empfängt. Dies wird allgemein als HTTPS bezeichnet. Der Webserver verwendet ein Schlüsselpaar aus einem öffentlichen und einem privaten Schlüssel sowie ein öffentliches SSL/TLS-Schlüsselzertifikat, um eine HTTPS-Sitzung mit den einzelnen Clients einzurichten. Dieser Prozess erfordert eine Menge Rechenleistung für Webserver, aber Sie können einen Teil davon auf Ihren AWS CloudHSM-Cluster auslagern, was als SSL-Beschleunigung bezeichnet wird. Das Offloading reduziert die Rechenlast auf Ihren Webservern und bietet zusätzliche Sicherheit, indem die privaten Schlüssel der Server in HSMs gespeichert werden.

Die folgenden Themen geben einen Überblick über die Funktionsweise der SSL-TLS-Auslagerung mit AWS CloudHSM und enthalten Anleitungen zum Einrichten der SSL/TLS-Auslagerung mit AWS CloudHSM auf den folgenden Plattformen:

Für Linux verwenden Sie OpenSSL Dynamic Engine auf der NGINX- oder Apache-HTTP-Server-Webserversoftware

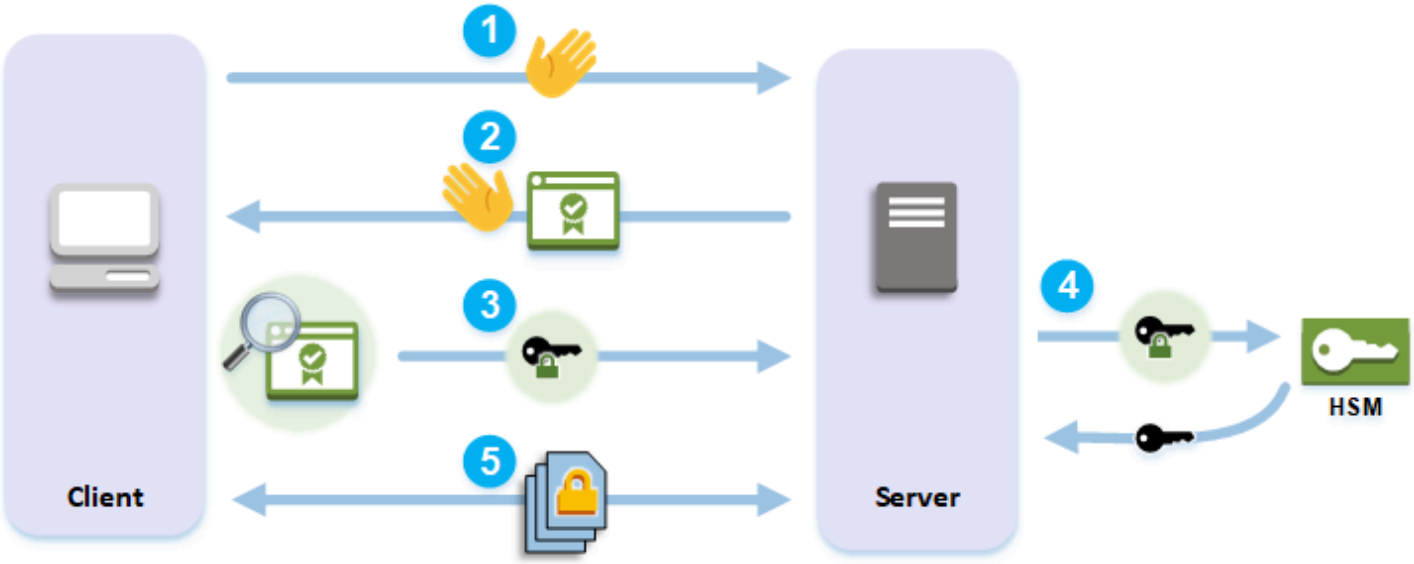
Für Windows verwenden Sie die Webserver-Software Internet Information Services (IIS) für Windows Server.

Themen

So funktioniert SSL/TLS-Offload mit AWS CloudHSM

Um eine HTTPS-Verbindung aufzubauen, führt Ihr Webserver einen Handshake-Prozess mit Clients durch. Als Teil dieses Prozesses überträgt der Server einen Teil der kryptographischen Verarbeitung auf die HSMs, wie in der folgenden Abbildung dargestellt. Jeder Schritt des Prozesses wird im Folgenden erläutert.

> **Note**
>
> Das folgende Bild und der Prozess gehen davon aus, dass RSA für die Serververifikation und den Schlüsselaustausch verwendet wird. Der Prozess sieht etwas anders aus, wenn Diffie–Hellman anstelle von RSA verwendet wird.



1. Der Client sendet eine Hello-Nachricht an den Server.
2. Der Server antwortet mit einer Hello-Nachricht und sendet das Zertifikat des Servers.
3. Der Client führt die folgenden Aktionen durch:
   a. Er überprüft, ob das SSL/TLS-Zertifikat des Servers von einem Root-Zertifikat signiert ist, denen der Client vertraut.
   b. Er extrahiert den öffentlichen Schlüssel aus dem Zertifikat des Servers.
   c. Er erzeugt ein Premaster-Geheimnis und verschlüsselt es mit dem öffentlichen Schlüssel des Servers.
   d. Er sendet das verschlüsselte Premaster-Geheimnis an den Server.
4. Um das Premaster-Geheimnis des Clients zu entschlüsseln, sendet der Server es an das HSM. Das HSM verwendet den privaten Schlüssel im HSM, um das Premaster-Geheimnis zu entschlüsseln, und sendet dann das Premaster-Geheimnis an den Server. Unabhängig davon verwenden Client und Server jeweils das Premaster-Geheimnis und einige Informationen aus den Hello-Nachrichten, um ein Master-Geheimnis zu berechnen.
5. Der Handshake-Vorgang ist beendet. Für den Rest der Sitzung werden alle Nachrichten, die zwischen dem Client und dem Server gesendet werden, mit Derivaten des Master-Geheimnisses verschlüsselt.

Informationen zur Konfiguration der SSL/TLS-Auslagerung in AWS CloudHSM finden Sie in den folgenden Themen:

- SSL/TLS-Auslagerung unter Linux

- Verwenden von IIS mit CNG für SSL/TLS-Offload unter Windows

SSL/TLS-Auslagerung unter Linux

Mit AWS CloudHSM können Sie SSL/TLS-Offload unter Linux mit NGINX, Apache und Tomcat durchführen. Weitere Informationen finden Sie in den entsprechenden Themen unten.

Themen

- Verwenden von NGINX oder Apache mit OpenSSL für SSL/TLS-Offload unter Linux
- Verwenden von Tomcat mit JSSE für SSL/TLS-Offload unter Linux

Verwenden von NGINX oder Apache mit OpenSSL für SSL/TLS-Offload unter Linux

In diesem Thema wird Schritt für Schritt erläutert, wie Sie SSL/TLS-Auslagerung mit AWS CloudHSM auf einem Linux-Webserver einrichten.

Themen

- Übersicht
- Schritt 1: Einrichten der Voraussetzungen
- Schritt 2: Generieren oder Importieren eines privaten Schlüssels und SSL/TLS-Zertifikats
- Schritt 3: Konfigurieren des Webservers
- Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats

Übersicht

Die NGINX- und Apache HTTP Server-Webserver-Software unter Linux ist nativ mit OpenSSL integriert, um HTTPS zu unterstützen. Die dynamische AWS CloudHSM-Engine für OpenSSL stellt eine Schnittstelle zur Verfügung, über die die Webserver-Software die HSMs im Cluster zur Auslagerung kryptografischer Aufgaben und zum Speichern von Schlüsseln verwenden kann. Die OpenSSL-Engine ist eine Brücke, die den Webserver mit dem AWS CloudHSM-Cluster verbindet.

Um diese Anleitung abzuschließen, müssen Sie zuerst entscheiden, ob Sie unter Linux die NGINX- oder Apache Webserver-Software verwenden möchten. Dann erfahren Sie, wie Sie folgende Aufgaben ausführen:

- Installieren der Web-Server-Software auf einer Amazon EC2-Instance

- Konfigurieren Sie die Webserver-Software so, dass sie HTTPS mit einem privaten Schlüssel unterstützt, der in Ihrem AWS CloudHSM-Cluster gespeichert ist.
- (Optional) Verwenden Sie Amazon EC2, um eine zweite Webserver-Instance zu erstellen, und Elastic Load Balancing, um einen Load Balancer zu erstellen. Mit einem Load Balancer kann die Leistung durch Verteilung der Arbeitslast auf mehrere Server gesteigert werden. Sie kann auch für Redundanz und eine höhere Verfügbarkeit sorgen, falls ein oder mehrere Webserver ausfallen.

Wenn Sie bereit sind, sehen Sie sich Schritt 1: Einrichten der Voraussetzungen an.

Schritt 1: Einrichten der Voraussetzungen

Verschiedene Plattformen erfordern unterschiedliche Voraussetzungen. Verwenden Sie den Abschnitt mit den Voraussetzungen unten, der zu Ihrer Plattform passt.

Themen

- Voraussetzungen für das Client-SDK 5
- Voraussetzungen für das Client-SDK 3

Voraussetzungen für das Client-SDK 5

Um SSL/TLS-Offload für Webserver mit Client-SDK 5 einzurichten, benötigen Sie Folgendes:

- Ein aktiver AWS CloudHSM-Cluster mit mindestens zwei Hardware-Sicherheitsmodulen (HSM)

> **Note**
>
> Sie können einen einzelnen HSM-Cluster verwenden, müssen aber zuerst die Haltbarkeit der Client-Schlüssel deaktivieren. Weitere Informationen finden Sie unter Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten und Client-SDK 5 Configure Tool.

- Eine Amazon EC2-Instance, auf der ein Linux-Betriebssystem ausgeführt wird und die folgende Software installiert ist:
  - Ein Webserver (entweder NGINX oder Apache)
  - Die OpenSSL Dynamic Engine für Client-SDK 5
- Ein Crypto-Benutzer (CU), der den privaten Schlüssel des Webservers auf dem HSM besitzen und verwalten soll.

So richten Sie eine Linux-Webserver-Instance auf dem HSM ein und erstellen einen CU

1. Installieren und Konfigurieren von OpenSSL Dynamic Engine für AWS CloudHSM. Weitere Informationen zur Installation der OpenSSL Dynamic Engine finden Sie unter OpenSSL Dynamic Engine für Client-SDK 5.
2. Installieren Sie auf einer EC2-Linux-Instance, die Zugriff auf Ihren Cluster hat, entweder den NGINX- oder den Apache-Webserver:

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

   Amazon Linux 2

   - Informationen zum Herunterladen der neuesten Version von NGINX auf Amazon Linux 2 finden Sie auf der NGINX-Website.

     Die neueste Version von NGINX, die für Amazon Linux 2 verfügbar ist, verwendet eine Version von OpenSSL, die neuer ist als die Systemversion von OpenSSL. Nach der Installation von NGINX müssen Sie einen symbolischen Link von der AWS CloudHSM-OpenSSL-Dynamic-Engine-Bibliothek zu dem Speicherort erstellen, den diese Version von OpenSSL erwartet

     ```
     $ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/
     engines-1.1/cloudhsm.so
     ```

   - Apache

     ```
     $ sudo yum install httpd mod_ssl
     ```

CentOS 7

- Informationen zum Herunterladen der neuesten Version von NGINX auf CentOS 7 finden Sie auf der NGINX-Website.

  Die neueste Version von NGINX, die für CentOS 7 verfügbar ist, verwendet eine Version von OpenSSL, die neuer ist als die Systemversion von OpenSSL. Nach der Installation von NGINX müssen Sie einen symbolischen Link von der AWS CloudHSM-OpenSSL-Dynamic-Engine-Bibliothek zu dem Speicherort erstellen, den diese Version von OpenSSL erwartet

  ```
  $ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/
  engines-1.1/cloudhsm.so
  ```

- Apache

  ```
  $ sudo yum install httpd mod_ssl
  ```

Red Hat 7

- Informationen zum Herunterladen der neuesten Version von NGINX auf Red Hat 7 finden Sie auf der NGINX-Website.

  Die neueste Version von NGINX, die für Red Hat 7 verfügbar ist, verwendet eine Version von OpenSSL, die neuer ist als die Systemversion von OpenSSL. Nach der Installation von NGINX müssen Sie einen symbolischen Link von der AWS CloudHSM-OpenSSL-Dynamic-Engine-Bibliothek zu dem Speicherort erstellen, den diese Version von OpenSSL erwartet

  ```
  $ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/
  engines-1.1/cloudhsm.so
  ```

- Apache

  ```
  $ sudo yum install httpd mod_ssl
  ```

CentOS 8

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 8

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Ubuntu 18.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 20.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 22.04

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

3. Verwenden Sie die CloudHSM-CLI, um eine CU zu erstellen. Weitere Informationen zur Verwaltung von HSM-Benutzern finden Sie unter HSM-Benutzer mit der CloudHSM-CLI verwalten.

> **Tip**
>
> Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen sie später beim Generieren oder Importieren des privaten HTTPS-Schlüssels und -Zertifikats für Ihren Webserver.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Schritt 2: Generieren oder Importieren eines privaten Schlüssels und SSL/TLS-Zertifikats fort.

Hinweise

- Um Security-Enhanced Linux (SELinux) und Webserver zu verwenden, müssen Sie ausgehende TCP-Verbindungen an Port 2223 zulassen, dem Port, den Client-SDK 5 für die Kommunikation mit dem HSM verwendet.
- Um einen Cluster zu erstellen und zu aktivieren und einer EC2-Instance Zugriff auf den Cluster zu gewähren, führen Sie die Schritte unter Erste Schritte mit AWS CloudHSM aus. Die ersten Schritte bieten eine schrittweise Anleitung zum Erstellen eines aktiven Clusters mit einem HSM und einer Amazon EC2-Client-Instance. Sie können diese Client-Instance als Ihren Webserver verwenden.
- Um zu vermeiden, dass die Haltbarkeit von Client-Schlüsseln deaktiviert wird, fügen Sie Ihrem Cluster mehr als ein HSM hinzu. Weitere Informationen finden Sie unter Hinzufügen eines HSM.
- Um sich mit Ihrer Client-Instance zu verbinden, können Sie SSH oder PuTTY verwenden. Weitere Informationen finden Sie unter Herstellung einer Verbindung zu Ihrer Linux-Instance mit SSH oder Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY in der Amazon EC2-Dokumentation.

Voraussetzungen für das Client-SDK 3

Um SSL/TLS-Offload für Webserver mit Client-SDK 3 einzurichten, benötigen Sie Folgendes:

- Einen aktiven AWS CloudHSM-Cluster mit mindestens einem HSM.
- Eine Amazon EC2-Instance, auf der ein Linux-Betriebssystem ausgeführt wird und die folgende Software installiert ist:
  - Die AWS CloudHSM-Client und Befehlszeilen-Tools.
  - Die NGINX- oder Apache-Webserveranwendung.
  - Die dynamische AWS CloudHSM-Engine für OpenSSL.
- Ein Crypto-Benutzer (CU), der den privaten Schlüssel des Webservers auf dem HSM besitzen und verwalten soll.

So richten Sie eine Linux-Webserver-Instance auf dem HSM ein und erstellen einen CU

1. Führen Sie die Schritte unter Erste Schritte aus. Anschließend haben Sie einen aktiven Cluster mit einem HSM und einer Amazon EC2-Client-Instance. Ihre EC2-Instance wird mit den Befehlszeilen-Tools konfiguriert. Verwenden Sie diese Client-Instance als Ihren Webserver.
2. Stellen Sie eine Verbindung mit Ihrer Client-Instance her. Weitere Informationen finden Sie unter Herstellung einer Verbindung zu Ihrer Linux-Instance mit SSH oder Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY in der Amazon EC2-Dokumentation.
3. Installieren Sie auf einer EC2-Linux-Instance, die Zugriff auf Ihren Cluster hat, entweder den NGINX- oder den Apache-Webserver:

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

   Amazon Linux 2

   - NGINX Version 1.19 ist die neueste Version von NGINX, die mit der Client-SDK 3-Engine auf Amazon Linux 2 kompatibel ist.

Weitere Informationen und den Download der NGINX-Version 1.19 finden Sie auf der NGINX-Website.

- Apache

```
$ sudo yum install httpd mod_ssl
```

CentOS 7

- NGINX Version 1.19 ist die neueste Version von NGINX, die mit der Client-SDK 3-Engine auf CentOS 7 kompatibel ist.

  Weitere Informationen und den Download der NGINX-Version 1.19 finden Sie auf der NGINX-Website.

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 7

- NGINX Version 1.19 ist die neueste Version von NGINX, die mit der Client-SDK 3-Engine auf Red Hat 7 kompatibel ist.

  Weitere Informationen und den Download der NGINX-Version 1.19 finden Sie auf der NGINX-Website.

- Apache

```
$ sudo yum install httpd mod_ssl
```

Ubuntu 16.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 18.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

4. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter Hinzufügen eines HSM.
5. Verwenden Sie cloudhsm_mgmt_util zum Erstellen eines CU. Weitere Informationen finden Sie unter Verwalten von HSM-Benutzern. Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen sie später beim Generieren oder Importieren des privaten HTTPS-Schlüssels und -Zertifikats für Ihren Webserver.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Schritt 2: Generieren oder Importieren eines privaten Schlüssels und SSL/TLS-Zertifikats fort.

Schritt 2: Generieren oder Importieren eines privaten Schlüssels und SSL/TLS-Zertifikats

Um HTTPS zu aktivieren benötigt Ihre Webserveranwendung (NGINX oder Apache) einen privaten Schlüssel und ein entsprechendes SSL/TLS-Zertifikat. Um die Webserver-SSL/TLS-Auslagerung mit AWS CloudHSM verwenden zu können, müssen Sie den privaten Schlüssel in einem HSM in Ihrem AWS CloudHSM-Cluster speichern. Sie können dafür eine der folgenden Möglichkeiten auswählen:

- Wenn Sie noch nicht über einen privaten Schlüssel und ein entsprechendes Zertifikat verfügen, generieren Sie einen privaten Schlüssel in einem HSM. Sie verwenden den privaten Schlüssel, um eine Zertifikatsignierungsanforderung (CSR) zu erstellen, mit der Sie das SSL/TLS-Zertifikat erstellen.
- Wenn Sie bereits einen privaten Schlüssel und ein entsprechendes Zertifikat besitzen, importieren Sie den privaten Schlüssel in ein HSM.

Unabhängig davon, für welche der oben genannten Methoden Sie sich entscheiden, exportieren Sie einen gefälschten privaten PEM-Schlüssel aus dem HSM. Dabei handelt es sich um eine private Schlüsseldatei im PEM-Format, die einen Verweis auf den privaten Schlüssel enthält, der auf dem HSM gespeichert ist (es ist nicht der eigentliche private Schlüssel). Ihr Webserver verwendet die gefälschte private PEM-Schlüsseldatei, um den privaten Schlüssel auf dem HSM während des SSL/TLS-Offloads zu identifizieren.

Führen Sie eine der folgenden Aktionen aus:

- Generieren eines privaten Schlüssels und Zertifikats
- Importieren Sie einen vorhandenen privaten Schlüssel und ein vorhandenes Zertifikat

Generieren eines privaten Schlüssels und Zertifikats

Generieren eines privaten Schlüssels

In diesem Abschnitt erfahren Sie, wie Sie mit dem Key Management Utility (KMU) aus dem Client-SDK 3 ein Schlüsselpaar generieren. Sobald Sie ein Schlüsselpaar im HSM generiert haben, können Sie es als gefälschte PEM-Datei exportieren und das entsprechende Zertifikat generieren.

Private Schlüssel, die mit dem Key Management Utility (KMU) generiert wurden, können sowohl mit Client-SDK 3 als auch mit Client-SDK 5 verwendet werden.

Installieren und konfigurieren Sie das Key Management Utility (KMU)

1. Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
2. Installieren und konfigurieren Sie das Client-SDK 3.
3. Führen Sie den folgenden Befehl aus, um den AWS CloudHSM-Client zu starten.

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client start
   ```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

```
$ sudo service cloudhsm-client start
```

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

4. Führen Sie den folgenden Befehl aus, um das Befehlszeilen-Tool key_mgmt_util zu starten.

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. Führen Sie den folgenden Befehl aus, um sich beim HSM anzumelden. Ersetzen Sie *<user name>* und *<password>* durch den Benutzernamen und das Passwort des Krytpobenutzers (CU).

```
Command: loginHSM -u CU -s <user name> -p <password>>
```

Generieren eines privaten Schlüssels

Je nach Anwendungsfall können Sie entweder ein RSA- oder ein EC-Schlüsselpaar generieren. Führen Sie eine der folgenden Aktionen aus:

- So erstellen Sie einen privaten RSA-Schlüssel in einem HSM

  Verwenden Sie den `genRSAKeyPair`-Befehl, um ein RSA-Schlüsselpaar zu erzeugen. In diesem Beispiel wird ein RSA-Schlüsselpaar mit einem Modul von 2048, einem öffentlichen Exponenten von 65537 und der Bezeichnung *`tls_rsa_keypair`* generiert.

```
Command: genRSAKeyPair -m 2048 -e 65537 -l tls_rsa_keypair
```

  Wenn der Befehl erfolgreich war, sollte die folgende Ausgabe angezeigt werden, die darauf hinweist, dass Sie erfolgreich ein RSA-Schlüsselpaar generiert haben.

```
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

                Cfm3GenerateKeyPair:    public key handle: 7    private key handle: 8

                Cluster Status:
                Node id 1 status: 0x00000000 : HSM Return: SUCCESS
```

- So erstellen Sie einen privaten EC-Schlüssel in einem HSM

  Verwenden Sie den `genECCKeyPair`-Befehl, um ein EC-Schlüsselpaar zu generieren. In diesem Beispiel wird ein EC-Schlüsselpaar mit der Kurven-ID 2 (entspricht der `NID_X9_62_prime256v1`-Kurve) und der Bezeichnung *`tls_ec_keypair`* generiert.

```
Command: genECCKeyPair -i 2 -l tls_ec_keypair
```

  Wenn der Befehl erfolgreich war, sollte die folgende Ausgabe angezeigt werden, die darauf hinweist, dass Sie erfolgreich ein EC-Schlüsselpaar generiert haben.

```
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

                Cfm3GenerateKeyPair:    public key handle: 7    private key handle: 8

                Cluster Status:
                Node id 1 status: 0x00000000 : HSM Return: SUCCESS
```

Exportieren einer gefälschten privaten PEM-Schlüsseldatei

Sobald Sie einen privaten Schlüssel auf dem HSM haben, müssen Sie eine gefälschte private PEM-Schlüsseldatei exportieren. Diese Datei enthält nicht die eigentlichen Schlüsseldaten, ermöglicht es der OpenSSL Dynamic Engine jedoch, den privaten Schlüssel auf dem HSM zu identifizieren. Sie können dann den privaten Schlüssel verwenden, um eine Zertifikatsignierungsanforderung (CSR) zu erstellen und die CSR zu signieren, um das Zertifikat zu erstellen.

> **Note**
>
> Gefälschte PEM-Dateien, die mit dem Key Management Utility (KMU) generiert wurden, können sowohl mit Client-SDK 3 als auch mit Client-SDK 5 verwendet werden.

Identifizieren Sie das Schlüssel-Handle, das dem Schlüssel entspricht, den Sie als gefälschtes PEM exportieren möchten, und führen Sie dann den folgenden Befehl aus, um den privaten Schlüssel im gefälschten PEM-Format zu exportieren und in einer Datei zu speichern. Ersetzen Sie die folgenden Werte durch Ihre eigenen.

- *<private_key_handle>* – Handle des generierten privaten Schlüssels. Dieses Handle wurde durch einen der Schlüsselgenerierungsbefehle im vorangegangenen Schritt erzeugt. Im vorhergehenden Beispiel lautet das Handle des privaten Schlüssels 8.
- *<web_server_fake_PEM.key>* – Name der Datei, in die Ihr gefälschter PEM-Schlüssel geschrieben wird.

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

Beenden

Führen Sie den folgenden Befehl aus, um key_mgmt_util zu beenden.

```
Command: exit
```

Sie sollten jetzt eine neue Datei auf Ihrem System haben, die sich unter dem *<web_server_fake_PEM.key>* im vorherigen Befehl angegebenen Pfad befindet. Diese Datei ist die gefälschte private PEM-Schlüsseldatei.

Generieren eines selbstsignierten Zertifikats

Sobald Sie einen gefälschten privaten PEM-Schlüssel generiert haben, können Sie diese Datei verwenden, um eine Zertifikatssignieranforderung (CSR) und ein Zertifikat zu generieren.

In einer Produktionsumgebung verwenden Sie in der Regel eine Zertifikatsstelle (CA) zum Erstellen eines Zertifikats aus einer CSR. Für eine Testumgebung ist keine CA erforderlich. Wenn Sie eine Zertifizierungsstelle verwenden, senden Sie ihnen die CSR-Datei und verwenden Sie das signierte SSL/TLS-Zertifikat, das sie Ihnen auf Ihrem Webserver für HTTPS zur Verfügung stellen.

Alternativ zur Verwendung einer CA können Sie die AWS CloudHSM OpenSSL Dynamic Engine verwenden, um ein selbstsigniertes Zertifikat zu erstellen. Selbstsignierte Zertifikate sind nicht vertrauenswürdig für Browser und sollten in Produktionsumgebungen nicht verwendet werden. Sie können in Testumgebungen verwendet werden.

> **Warning**
>
> Selbstsignierte Zertifikate sollten nur in einer Testumgebung verwendet werden. Für eine Produktionsumgebung, verwenden Sie eine sicherere Methode, wie z. B. eine Zertifikatstelle, um ein Zertifikat zu erstellen.

Installieren und Konfigurieren von OpenSSL Dynamic Engine

1. Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
2. Führen Sie zur Installation und Konfiguration einen der folgenden Schritte aus:
   - the section called “Installieren von OpenSSL Dynamic Engine”
   - the section called “OpenSSL Dynamic Engine”

Generieren eines Zertifikats

1. Besorgen Sie sich eine Kopie Ihrer gefälschten PEM-Datei, die in einem früheren Schritt generiert wurde.
2. Erstellen einer CSR

   Führen Sie den folgenden Befehl aus, um mit der AWS CloudHSM OpenSSL Dynamic Engine eine Zertifikatsignierungsanfrage (CSR) zu erstellen. Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält. Ersetzen Sie *<web_server.csr>* durch den Namen der Datei, die Ihre CSR enthält.

   Der Befehl `req` ist interaktiv. Füllen Sie jedes Feld aus. Die Feldinformationen werden in Ihr SSL/TLS-Zertifikat kopiert.

   ```
   $ openssl req -engine cloudhsm -new -key <web_server_fake_PEM.key> -
   out <web_server.csr>
   ```
3. Erstellen eines selbstsignierten Zertifikats

   Führen Sie den folgenden Befehl aus, um die AWS CloudHSM OpenSSL Dynamic Engine zum Signieren Ihrer CSR mit Ihrem privaten Schlüssel auf Ihrem HSM zu verwenden. Dadurch wird ein selbstsigniertes Zertifikat erstellt. Ersetzen Sie die folgenden Werte in dem Befehl durch Ihre eigenen.

   - *<web_server.csr>* – Name der Datei, die die CSR enthält.
   - *<web_server_fake_PEM.key>* – Name der Datei, die den gefälschten privaten PEM-Schlüssel enthält.
   - *<web_server.crt>* – Name der Datei, die Ihr Webserver-Zertifikat enthalten wird.

   ```
   $ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -
   signkey <web_server_fake_PEM.key> -out <web_server.crt>
   ```

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Schritt 3: Konfigurieren des Webservers fort.

Importieren Sie einen vorhandenen privaten Schlüssel und ein vorhandenes Zertifikat

Möglicherweise sind bereits ein privater Schlüssel und ein entsprechendes SSL/TLS-Zertifikat für die Verwendung von HTTPS auf Ihrem Webserver verfügbar. Wenn ja, können Sie diesen Schlüssel in ein HSM importieren, indem Sie die Schritte in diesem Abschnitt ausführen.

> **Note**
>
> Einige Hinweise zum Import privater Schlüssel und zur Client-SDK-Kompatibilität:
>
> - Für den Import eines vorhandenen privaten Schlüssels ist Client-SDK 3 erforderlich.
> - Sie können private Schlüssel aus dem Client-SDK 3 mit dem Client-SDK 5 verwenden.
> - OpenSSL Dynamic Engine für Client-SDK 3 unterstützt die neuesten Linux-Plattformen nicht, die Implementierung von OpenSSL Dynamic Engine für Client-SDK 5 jedoch schon. Sie können einen vorhandenen privaten Schlüssel mit dem im Client-SDK 3 enthaltenen Key Management Utility (KMU) importieren und dann diesen privaten Schlüssel und die Implementierung von OpenSSL Dynamic Engine mit Client-SDK 5 verwenden, um SSL/TLS-Offload auf den neuesten Linux-Plattformen zu unterstützen.

So importieren Sie einen vorhandenen privaten Schlüssel in ein HSM mit Client-SDK 3

1. Stellen Sie eine Verbindung zu Ihrer Amazon-EC2-Client-Instance her. Falls erforderlich, kopieren Sie Ihren vorhandenen privaten Schlüssel und das Zertifikat in die Instance.
2. Installieren und konfigurieren Sie das Client-SDK 3
3. Führen Sie den folgenden Befehl aus, um den AWS CloudHSM-Client zu starten.

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client start
   ```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

```
$ sudo service cloudhsm-client start
```

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

4. Führen Sie den folgenden Befehl aus, um das Befehlszeilen-Tool key_mgmt_util zu starten.

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. Führen Sie den folgenden Befehl aus, um sich beim HSM anzumelden. Ersetzen Sie *<user name>* und *<password>* durch den Benutzernamen und das Passwort des Krytpobenutzers (CU).

```
Command: loginHSM -u CU -s <user name> -p <password>
```

6. Führen Sie die folgenden Befehle aus, um Ihren privaten Schlüssel in ein HSM zu importieren.

a. Führen Sie den folgenden Befehl aus, um einen symmetrischen Verpackungsschlüssel zu erstellen, der nur für die aktuelle Sitzung gültig ist. Der Befehl und die Ausgabe sind gezeigt.

```
Command: genSymKey -t 31 -s 16 -sess -l wrapping_key_for_import

Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS
Symmetric Key Created.  Key Handle: 6
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

b. Führen Sie den folgenden Befehl aus, um Ihren vorhandenen privaten Schlüssel in ein HSM zu importieren. Der Befehl und die Ausgabe sind gezeigt. Ersetzen Sie die folgenden Werte durch Ihre eigenen:

- *<web_server_existing.key>* – Name der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.
- *<web_server_imported_key>* – Bezeichnung für Ihren importierten, privaten Schlüssel.
- *<wrapping_key_handle>* – Verpackungsschlüsselreferenz, die vom vorherigen Befehl generiert wurde. Im vorherigen Beispiel lautet die Verpackungsschlüssel-Handle 6.

```
Command: importPrivateKey -f <web_server_existing.key> -
l <web_server_imported_key> -w <wrapping_key_handle>

BER encoded key length is 1219
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS
Private Key Unwrapped.  Key Handle: 8
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

7. Führen Sie den folgenden Befehl aus, um den privaten Schlüssel im gefälschten PEM-Format zu exportieren und speichern Sie ihn in einer Datei. Ersetzen Sie die folgenden Werte durch Ihre eigenen.

- *<private_key_handle>* – Handle des importierten, privaten Schlüssels. Dieses Handle wurde im zweiten Befehl des vorhergehenden Schritts generiert. Im vorhergehenden Beispiel lautet das Handle des privaten Schlüssels 8.

- *<web_server_fake_PEM.key>* – Name der Datei, die Ihren exportierten, gefälschten privaten PEM-Schlüssel enthält.

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

8. Führen Sie den folgenden Befehl aus, um key_mgmt_util zu beenden.

```
Command: exit
```

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Schritt 3: Konfigurieren des Webservers fort.

Schritt 3: Konfigurieren des Webservers

Aktualisieren Sie die Konfiguration Ihrer Webserver-Software, um das HTTPS-Zertifikat und den zugehörigen gefälschten privaten PEM-Schlüssel zu verwenden, die Sie im vorherigen Schritt erstellt haben. Denken Sie daran, Ihre vorhandenen Zertifikate und Schlüssel zu sichern, bevor Sie beginnen. Damit schließen Sie die Einrichtung Ihrer Linux-Webserver-Software für SSL/TLS-Auslagerung mit AWS CloudHSM ab.

Führen Sie die Schritte aus einem der folgenden Abschnitte aus.

Themen

- Konfigurieren eines NGINX-Webservers
- Konfigurieren des Apache-Webservers

Konfigurieren eines NGINX-Webservers

Verwenden Sie diesen Abschnitt, um NGINX auf unterstützten Plattformen zu konfigurieren.

So aktualisieren Sie die Webserverkonfiguration für NGINX

1. Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
2. Führen Sie den folgenden Befehl aus, um die erforderlichen Verzeichnisse für das Webserverzertifikat und den gefälschten privaten PEM-Schlüssel zu erstellen.

```
$ sudo mkdir -p /etc/pki/nginx/private
```

3. Führen Sie den folgenden Befehl aus, um Ihr Webserverzertifikat an die erforderliche Stelle zu kopieren. Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

```
$ sudo cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. Führen Sie den folgenden Befehl aus, um den gefälschten privaten PEM-Schlüssel an die erforderliche Stelle zu kopieren. Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/nginx/private/server.key
```

5. Führen Sie den folgenden Befehl aus, um die Eigentümerschaft für diese Dateien zu ändern, sodass der Benutzer mit dem Namen nginx diese lesen kann.

```
$ sudo chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. Führen Sie den folgenden Befehl aus, um die Datei `/etc/nginx/nginx.conf` zu sichern.

```
$ sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

7. Aktualisieren Sie die NGINX-Konfiguration.

> **Note**
>
> Jeder Cluster kann maximal 1000 NGINX-Worker-Prozesse auf allen NGINX-Webservern unterstützen.

Amazon Linux

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

- Bei Verwendung von Client-SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Bei Verwendung von Client-SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
```

```
}
```

Amazon Linux 2

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

- Bei Verwendung von Client-SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Bei Verwendung von Client-SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
```

```
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

CentOS 7

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

- Bei Verwendung von Client-SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Bei Verwendung von Client-SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
```

```
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

CentOS 8

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;
```

```
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Red Hat 7

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

- Bei Verwendung von Client-SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Bei Verwendung von Client-SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;
```

```
    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Red Hat 8

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
```

```
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Ubuntu 16.04 LTS

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;
    env n3fips_password;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
    server {
        listen       443 ssl http2 default_server;
        listen       [::]:443 ssl http2 default_server;
        server_name  _;
        root         /usr/share/nginx/html;
```

```
        ssl_certificate "/etc/pki/nginx/server.crt";
        ssl_certificate_key "/etc/pki/nginx/private/server.key";
        # It is *strongly* recommended to generate unique DH parameters
        # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem
 2048
        #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_protocols TLSv1.2;
        ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
        ssl_prefer_server_ciphers on;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
        }

        error_page 404 /404.html;
        location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
    }
```

Ubuntu 18.04 LTS

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
    server {
        listen       443 ssl http2 default_server;
        listen       [::]:443 ssl http2 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        ssl_certificate "/etc/pki/nginx/server.crt";
        ssl_certificate_key "/etc/pki/nginx/private/server.key";
        # It is *strongly* recommended to generate unique DH parameters
        # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem
 2048
        #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
        ssl_prefer_server_ciphers on;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
        }

        error_page 404 /404.html;
        location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
    }
```

Ubuntu 20.04 LTS

Verwenden Sie einen Texteditor, um die Datei `/etc/nginx/nginx.conf` zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie am Anfang der Datei die folgende Zeilen hinzu:

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

Fügen Sie dann dem Abschnitt TLS der Datei Folgendes hinzu:

```
# Settings for a TLS enabled server.
    server {
        listen       443 ssl http2 default_server;
        listen       [::]:443 ssl http2 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        ssl_certificate "/etc/pki/nginx/server.crt";
        ssl_certificate_key "/etc/pki/nginx/private/server.key";
        # It is *strongly* recommended to generate unique DH parameters
        # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem
 2048
        #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
        ssl_prefer_server_ciphers on;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
        }

        error_page 404 /404.html;
```

```
        location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
    }
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

Speichern Sie die Datei.

8. Sichern Sie die `systemd`-Konfigurationsdatei und legen Sie dann den `EnvironmentFile`-Pfad fest.

   Amazon Linux

   Es ist keine Aktion erforderlich.

   Amazon Linux 2

   1. Sichern Sie die Datei `nginx.service`.

      ```
      $ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
      nginx.service.backup
      ```

   2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt [Service] den folgenden Pfad hinzu:

      ```
      EnvironmentFile=/etc/sysconfig/nginx
      ```

   CentOS 7

   Es ist keine Aktion erforderlich.

   CentOS 8

   1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt [Service] den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 7

Es ist keine Aktion erforderlich.

Red Hat 8

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt [Service] den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 16.04

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt [Service] den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 18.04

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt [Service] den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 20.04 LTS

1. Sichern Sie die Datei `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Öffnen Sie die Datei `/lib/systemd/system/nginx.service` in einem Texteditor. Fügen Sie dann im Abschnitt [Service] den folgenden Pfad hinzu:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

9. Überprüfen Sie, ob die Datei `/etc/sysconfig/nginx` vorhanden ist, und führen Sie dann einen der folgenden Schritte aus:

- Wenn die Datei vorhanden ist, sichern Sie die Datei, indem Sie den folgenden Befehl ausführen:

```
$ sudo cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- Öffnen Sie einen Texteditor und erstellen Sie im Ordner `/etc/sysconfig/` eine Datei mit dem Namen `nginx`, wenn die Datei nicht vorhanden ist.

10. Konfigurieren Sie die NGINX-Umgebung.

> **Note**
>
> Das Client-SDK 5 führt die `CLOUDHSM_PIN`-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein.

Amazon Linux

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für Crypto-Benutzer (CU) hinzu:

- Bei Verwendung von Client-SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Bei Verwendung von Client-SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

Amazon Linux 2

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

- Bei Verwendung von Client-SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Bei Verwendung von Client-SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

CentOS 7

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

- Bei Verwendung von Client-SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Bei Verwendung von Client-SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

CentOS 8

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

Red Hat 7

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

- Bei Verwendung von Client-SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Bei Verwendung von Client-SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

Red Hat 8

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

Ubuntu 16.04 LTS

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
n3fips_password=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

Ubuntu 18.04 LTS

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

Ubuntu 20.04 LTS

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Speichern Sie die Datei.

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

11. Starten Sie den NGINX-Webserver.

Amazon Linux

Öffnen Sie die Datei `/etc/sysconfig/nginx` in einem Text-Editor. Dies erfordert Linux-Root-Berechtigungen. Fügen Sie die Anmeldeinformationen für den Crypto-Benutzer (CU) hinzu:

```
$ sudo service nginx start
```

Amazon Linux 2

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die `systemd`-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

CentOS 7

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die `systemd`-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

CentOS 8

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die `systemd`-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Red Hat 7

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die `systemd`-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Red Hat 8

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die `systemd`-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Ubuntu 16.04 LTS

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die `systemd`-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Ubuntu 18.04 LTS

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die `systemd`-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Ubuntu 20.04 LTS

Stoppen Sie alle laufenden NGINX-Prozesse

```
$ sudo systemctl stop nginx
```

Laden Sie die `systemd`-Konfiguration neu, um die neuesten Änderungen zu erhalten

```
$ sudo systemctl daemon-reload
```

Starten Sie den NGINX-Prozess

```
$ sudo systemctl start nginx
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

12. (Optional) Konfigurieren Sie Ihre Plattform so, dass NGINX beim Start gestartet wird.

Amazon Linux

```
$ sudo chkconfig nginx on
```

Amazon Linux 2

```
$ sudo systemctl enable nginx
```

CentOS 7

Es ist keine Aktion erforderlich.

CentOS 8

```
$ sudo systemctl enable nginx
```

Red Hat 7

Es ist keine Aktion erforderlich.

Red Hat 8

```
$ sudo systemctl enable nginx
```

Ubuntu 16.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 18.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 20.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

Nachdem Sie Ihre Webserverkonfiguration aktualisiert haben, gehen Sie zu Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats.

Konfigurieren des Apache-Webservers

Verwenden Sie diesen Abschnitt, um Apache auf unterstützten Plattformen zu konfigurieren.

So aktualisieren Sie die Webserverkonfiguration für Apache

1. Stellen Sie eine Verbindung zu Ihrer Amazon-EC2-Client-Instance her.
2. Definieren Sie Standardspeicherorte für Zertifikate und private Schlüssel für Ihre Plattform.

   Amazon Linux

   Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

   ```
   SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
   SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
   ```

   Amazon Linux 2

   Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

   ```
   SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
   SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
   ```

CentOS 7

Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

CentOS 8

Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 7

Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 8

Stellen Sie sicher, dass in der `/etc/httpd/conf.d/ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Ubuntu 16.04 LTS

Stellen Sie sicher, dass in der `/etc/apache2/sites-available/default-ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 18.04 LTS

Stellen Sie sicher, dass in der `/etc/apache2/sites-available/default-ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 20.04 LTS

Stellen Sie sicher, dass in der `/etc/apache2/sites-available/default-ssl.conf`-Datei die folgenden Werte vorhanden sind:

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

3. Kopieren Sie Ihr Webserver-Zertifikat an den für Ihre Plattform erforderlichen Speicherort.

Amazon Linux

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

Amazon Linux 2

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

CentOS 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

CentOS 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

Red Hat 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

Red Hat 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

Ubuntu 16.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

Ubuntu 18.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

Ubuntu 20.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Ersetzen Sie *<web_server.crt>* durch den Namen Ihres Webserverzertifikats.

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

4. Kopieren Sie Ihren gefälschten privaten PEM-Schlüssel an den für Ihre Plattform erforderlichen Speicherort.

Amazon Linux

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Amazon Linux 2

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

CentOS 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

CentOS 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Red Hat 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Red Hat 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Ubuntu 16.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Ubuntu 18.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Ubuntu 20.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

Ersetzen Sie *<web_server_fake_PEM.key>* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält.

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

5. Ändern Sie den Besitz dieser Dateien, falls Ihre Plattform dies erfordert.

Amazon Linux

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

Amazon Linux 2

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

CentOS 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

CentOS 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

Red Hat 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

Red Hat 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Gewährt dem Benutzer mit dem Namen Apache Leseberechtigung.

Ubuntu 16.04 LTS

Es ist keine Aktion erforderlich.

Ubuntu 18.04 LTS

Es ist keine Aktion erforderlich.

Ubuntu 20.04 LTS

Es ist keine Aktion erforderlich.

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

6. Konfigurieren Sie Apache-Direktiven für Ihre Plattform.

Amazon Linux

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Speichern Sie die Datei.

Amazon Linux 2

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
```

```
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Speichern Sie die Datei.

CentOS 7

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Speichern Sie die Datei.

CentOS 8

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
SSLProtocol TLSv1.2 TLSv1.3
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProxyCipherSuite HIGH:!aNULL
```

Speichern Sie die Datei.

Red Hat 7

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Speichern Sie die Datei.

Red Hat 8

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/httpd/conf.d/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
SSLProtocol TLSv1.2 TLSv1.3
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProxyCipherSuite HIGH:!aNULL
```

Speichern Sie die Datei.

Ubuntu 16.04 LTS

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/apache2/mods-available/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Speichern Sie die Datei.

Aktivieren Sie das SSL-Modul und die standardmäßige SSL-Site-Konfiguration:

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 18.04 LTS

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/apache2/mods-available/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProtocol TLSv1.2 TLSv1.3
```

Speichern Sie die Datei.

Aktivieren Sie das SSL-Modul und die standardmäßige SSL-Site-Konfiguration:

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 20.04 LTS

Suchen Sie die SSL-Datei für diese Plattform:

```
/etc/apache2/mods-available/ssl.conf
```

Diese Datei enthält Apache-Direktiven, die definieren, wie Ihr Server laufen soll. Direktiven erscheinen auf der linken Seite, gefolgt von einem Wert. Verwenden Sie einen Texteditor, um diese Datei zu bearbeiten. Dies erfordert Linux-Root-Berechtigungen.

Aktualisieren Sie die folgenden Direktiven oder geben Sie sie mit diesen Werten ein:

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProtocol TLSv1.2 TLSv1.3
```

Speichern Sie die Datei.

Aktivieren Sie das SSL-Modul und die standardmäßige SSL-Site-Konfiguration:

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

7. Konfigurieren Sie eine Datei mit Umgebungswerten für Ihre Plattform.

Amazon Linux

Es ist keine Aktion erforderlich. Umgebungswerte werden gehen zu `/etc/sysconfig/httpd`

Amazon Linux 2

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

Fügen Sie unter dem Abschnitt `[Service]` Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 7

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

Fügen Sie unter dem Abschnitt `[Service]` Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 8

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

Fügen Sie unter dem Abschnitt `[Service]` Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 7

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

Fügen Sie unter dem Abschnitt `[Service]` Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 8

Öffnen Sie die HTTPD-Servicedatei:

```
/lib/systemd/system/httpd.service
```

Fügen Sie unter dem Abschnitt `[Service]` Folgendes hinzu:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Ubuntu 16.04 LTS

Es ist keine Aktion erforderlich. Umgebungswerte werden gehen zu `/etc/sysconfig/httpd`

Ubuntu 18.04 LTS

Es ist keine Aktion erforderlich. Umgebungswerte werden gehen zu `/etc/sysconfig/httpd`

Ubuntu 20.04 LTS

Es ist keine Aktion erforderlich. Umgebungswerte werden gehen zu `/etc/sysconfig/httpd`

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

8. Legen Sie in der Datei, in der Umgebungsvariablen für Ihre Plattform gespeichert sind, eine Umgebungsvariable fest, die die Anmeldeinformationen für den Crypto-Benutzer (CU) enthält:

   Amazon Linux

   Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

   - Bei Verwendung von Client-SDK 3

     ```
     n3fips_password=<CU user name>:<password>
     ```

   - Bei Verwendung von Client-SDK 5

     ```
     CLOUDHSM_PIN=<CU user name>:<password>
     ```

   Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Amazon Linux 2

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

- Bei Verwendung von Client-SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Bei Verwendung von Client-SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

CentOS 7

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

- Bei Verwendung von Client-SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Bei Verwendung von Client-SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

CentOS 8

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Red Hat 7

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

- Bei Verwendung von Client-SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Bei Verwendung von Client-SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Red Hat 8

Verwenden Sie einen Texteditor, um `/etc/sysconfig/httpd` zu bearbeiten.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

> **Note**
>
> Das Client-SDK 5 führt die `CLOUDHSM_PIN`-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein.

Ubuntu 16.04 LTS

Verwenden Sie einen Texteditor, um `/etc/apache2/envvars` zu bearbeiten.

```
export n3fips_password=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

Ubuntu 18.04 LTS

Verwenden Sie einen Texteditor, um `/etc/apache2/envvars` zu bearbeiten.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

> **Note**
>
> Das Client-SDK 5 führt die `CLOUDHSM_PIN`-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein. Im Client-SDK 3 speichern Sie die CU-Anmeldeinformationen in der `n3fips_password`-Umgebungsvariable. Client-SDK 5 unterstützt beide Umgebungsvariablen, wir empfehlen jedoch, `CLOUDHSM_PIN` zu verwenden.

Ubuntu 20.04 LTS

Verwenden Sie einen Texteditor, um `/etc/apache2/envvars` zu bearbeiten.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

Ersetzen Sie den *<CU user name>* und *<password>* durch die Anmeldeinformationen des Crypto-Benutzers.

> **Note**
>
> Das Client-SDK 5 führt die `CLOUDHSM_PIN`-Umgebungsvariable zum Speichern der Anmeldeinformationen der CU ein. Im Client-SDK 3 speichern Sie die CU-Anmeldeinformationen in der `n3fips_password`-Umgebungsvariable. Client-SDK 5 unterstützt beide Umgebungsvariablen, wir empfehlen jedoch, `CLOUDHSM_PIN` zu verwenden.

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

9. Starten Sie den Apache-Webserver.

Amazon Linux

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Amazon Linux 2

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

CentOS 7

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

CentOS 8

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Red Hat 7

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Red Hat 8

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Ubuntu 16.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 18.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 20.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

10. (Optional) Konfigurieren Sie Ihre Plattform so, dass Apache beim Start gestartet wird.

Amazon Linux

```
$ sudo chkconfig httpd on
```

Amazon Linux 2

```
$ sudo chkconfig httpd on
```

CentOS 7

```
$ sudo chkconfig httpd on
```

CentOS 8

```
$ systemctl enable httpd
```

Red Hat 7

```
$ sudo chkconfig httpd on
```

Red Hat 8

```
$ systemctl enable httpd
```

Ubuntu 16.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 18.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 20.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 22.04 LTS

Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar.

Nachdem Sie Ihre Webserverkonfiguration aktualisiert haben, gehen Sie zu Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats.

Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats

Nachdem Sie Ihren Webserver für SSL/TLS-Offload mit AWS CloudHSM konfiguriert haben, fügen Sie Ihre Webserver-Instanz einer Sicherheitsgruppe hinzu, die eingehenden HTTPS-Verkehr zulässt. Dadurch können Clients, wie z. B. Webbrowser, eine HTTPS-Verbindung mit Ihrem Webserver herstellen. Stellen Sie dann eine HTTPS-Verbindung zu Ihrem Webserver her und überprüfen Sie, ob er das Zertifikat verwendet, das Sie für SSL/TLS-Offload mit AWS CloudHSM konfiguriert haben.

Themen

- Aktivieren von eingehenden HTTPS-Verbindungen
- Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Aktivieren von eingehenden HTTPS-Verbindungen

Zum Herstellen einer Verbindung zu Ihrem Webserver von einem Client (z. B. ein Webbrowser) aus, erstellen Sie eine Sicherheitsgruppe, die eingehende HTTPS-Verbindungen zulässt. Insbesondere sollten eingehende TCP-Verbindungen auf Port 443 erlaubt werden. Weisen Sie diese Sicherheitsgruppe Ihrem Webserver zu.

So erstellen Sie eine Sicherheitsgruppe für HTTPS und weisen sie Ihrem Webserver zu

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie im Navigationsbereich Sicherheitsgruppen aus.
3. Wählen Sie Sicherheitsgruppe erstellen.
4. Führen Sie für Sicherheitsgruppe erstellen die folgenden Schritte aus:
    a. Geben Sie in das Feld Sicherheitsgruppenname einen Namen für die Sicherheitsgruppe ein, die Sie erstellen.
    b. (Optional) Geben Sie eine Beschreibung der Sicherheitsgruppe ein, die Sie erstellen.
    c. Wählen Sie für VPC die VPC aus, die Ihre Amazon-EC2-Instance enthält.
    d. Wählen Sie Regel hinzufügen aus.
    e. Wählen Sie im Drop-down-Fenster für Typ die Option HTTPS aus.
    f. Geben Sie für Quelle einen Quellspeicherort ein.
    g. Wählen Sie Sicherheitsgruppe erstellen.
5. Wählen Sie im Navigationsbereich Instances aus.
6. Aktivieren Sie das Kontrollkästchen neben Ihrer Webserver-Instance.
7. Wählen Sie das Drop-down-Menü Aktionen oben auf der Seite. Wählen Sie Sicherheit und dann Sicherheitsgruppen ändern aus.
8. Wählen Sie unter Zugeordnete Sicherheitsgruppen das Suchfeld aus und wählen Sie die Sicherheitsgruppe, die Sie für HTTPS erstellt haben, aus. Wählen Sie dann Sicherheitsgruppen hinzufügen aus.
9. Wählen Sie Speichern.

Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Nachdem Sie den Webserver zu einer Sicherheitsgruppe hinzugefügt haben, können Sie überprüfen, ob beim SSL/TLS-Offload Ihr selbstsigniertes Zertifikat verwendet wird. Sie können dazu einen Webbrowser oder ein Tool wie OpenSSL s_client nutzen.

So überprüfen Sie die SSL/TLS-Auslagerung mit einem Webbrowser

1. Verwenden Sie einen Web-Browser, um eine Verbindung zum Webserver unter Verwendung des öffentlichen DNS-Namen oder der IP-Adresse des Servers herzustellen. Stellen Sie sicher, dass die URL in die Adresszeile mit https:// beginnt. Zum Beispiel **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

   > **Tip**
   >
   > Sie können einen DNS-Service wie Amazon Route 53 nutzen, um den Domainnamen Ihrer Website (beispielsweise https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter Routing des Datenverkehrs zu einer Amazon-EC2-Instance im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:

   - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter Zertifikat anzeigen an.
   - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter Sicherheitsprobleme verstehen an.

   Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.

3. Stellen Sie sicher, dass das SSL/TLS-Zertifikat dasjenige ist, das Sie für die Nutzung in Ihrem Webserver konfiguriert haben.

So überprüfen Sie die SSL/TLS-Auslagerung mit OpenSSL s_client

1. Führen Sie den folgenden OpenSSL-Befehl aus, um mittels HTTPS eine Verbindung zu Ihrem Webserver herzustellen. Ersetzen Sie *server name>* durch den öffentlichen DNS-Namen oder die IP-Adresse Ihres Webservers.

   ```
   openssl s_client -connect <server name>:443
   ```

   > **Tip**
   >
   > Sie können einen DNS-Service wie Amazon Route 53 nutzen, um den Domainnamen Ihrer Website (beispielsweise https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter Routing des Datenverkehrs zu einer Amazon-EC2-Instance im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass das SSL/TLS-Zertifikat dasjenige ist, das Sie für die Nutzung in Ihrem Webserver konfiguriert haben.

Sie verfügen jetzt über eine mit HTTPS gesicherte Website. Der private Schlüssel für den Webserver ist in einem HSM in Ihrem AWS CloudHSM-Cluster gespeichert.

Informationen zum Hinzufügen eines Load Balancers finden Sie unter Load Balancer mit Elastic Load Balancing hinzufügen (optional).

Verwenden von Tomcat mit JSSE für SSL/TLS-Offload unter Linux

Dieses Thema enthält schrittweise Anleitungen zur Einrichtung von SSL/TLS-Offload mithilfe der Java Secure Socket Extension (JSSE) mit dem AWS CloudHSM JCE SDK.

Themen

Übersicht

In AWS CloudHSM funktionieren Tomcat-Webserver unter Linux, um HTTPS zu unterstützen. Das AWS CloudHSM JCE SDK bietet eine Schnittstelle, die mit JSSE (Java Secure Socket Extension) verwendet werden kann, um die Verwendung von HSMs für solche Webserver zu ermöglichen. AWS CloudHSM JCE ist die Brücke, die JSSE mit Ihrem AWS-CloudHSM-Cluster verbindet. JSSE ist eine Java-API für Secure Sockets Layer (SSL) und Transport Layer Security (TLS).

Schritt 1: Einrichten der Voraussetzungen

Erfüllen Sie diese Voraussetzungen, um einen Tomcat-Webserver mit AWS CloudHSM für SSL/TLS-Offload unter Linux zu verwenden. Diese Voraussetzungen müssen erfüllt sein, um den SSL/TLS-Offload des Webservers mit dem Client-SDK 5 und einem Tomcat-Webserver einzurichten.

> **Note**
>
> Verschiedene Plattformen erfordern unterschiedliche Voraussetzungen. Folgen Sie immer den richtigen Installationsschritten für Ihre Plattform.

Voraussetzungen

- Eine Amazon EC2-Instance, auf der ein Linux-Betriebssystem mit einem installierten Tomcat-Webserver ausgeführt wird.
- Ein Crypto-Benutzer (CU), der den privaten Schlüssel des Webservers auf dem HSM besitzen und verwalten soll.
- Ein aktiver AWS CloudHSM-Cluster mit mindestens zwei Hardware-Sicherheitsmodulen (HSMs), auf denen JCE für Client-SDK 5 installiert und konfiguriert ist.

> **Note**
>
> Sie können einen einzelnen HSM-Cluster verwenden, müssen aber zuerst die Haltbarkeit der Client-Schlüssel deaktivieren. Weitere Informationen finden Sie unter Einstellungen für die Haltbarkeit von Client-Schlüsseln verwalten und Client-SDK 5 Configure Tool.

Wie erfüllt man die Voraussetzungen

1. Installieren und konfigurieren Sie das JCE für AWS CloudHSM auf einem aktiven AWS CloudHSM-Cluster mit mindestens zwei Hardware-Sicherheitsmodulen (HSMs). Weitere Informationen zur Installation finden Sie unter JCE für Client-SDK 5.
2. Folgen Sie auf einer EC2-Linux-Instance, die Zugriff auf Ihren AWS CloudHSM-Cluster hat, den Apache-Tomcat-Anweisungen, um den Tomcat-Webserver herunterzuladen und zu installieren.
3. Verwenden Sie die CloudHSM-CLI, um einen Crypto-Benutzer (CU) zu erstellen. Weitere Informationen zur Verwaltung von HSM-Benutzern finden Sie unter HSM-Benutzer mit der CloudHSM-CLI verwalten.

   > **Tip**
   >
   > Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen sie später beim Generieren oder Importieren des privaten HTTPS-Schlüssels und -Zertifikats für Ihren Webserver.

4. Folgen Sie den Anweisungen unter Verwendung von Client-SDK 5 zur Integration mit Java Keytool und Jarsigner, um JCE mit Java Keytool einzurichten.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Schritt 2: Generieren oder Importieren eines privaten Schlüssels und SSL/TLS-Zertifikats fort.

Hinweise

- Um Security-Enhanced Linux (SELinux) und Webserver zu verwenden, müssen Sie ausgehende TCP-Verbindungen an Port 2223 zulassen, dem Port, den Client-SDK 5 für die Kommunikation mit dem HSM verwendet.
- Um einen Cluster zu erstellen und zu aktivieren und einer EC2-Instance Zugriff auf den Cluster zu gewähren, führen Sie die Schritte unter Erste Schritte mit AWS CloudHSM aus. Dieser Abschnitt enthält step-by-step Anweisungen zum Erstellen eines aktiven Clusters mit einem HSM und einer Amazon EC2-Client-Instance. Sie können diese Client-Instance als Ihren Webserver verwenden.
- Um zu vermeiden, dass die Haltbarkeit von Client-Schlüsseln deaktiviert wird, fügen Sie Ihrem Cluster mehr als ein HSM hinzu. Weitere Informationen finden Sie unter Hinzufügen eines HSM.
- Um sich mit Ihrer Client-Instance zu verbinden, können Sie SSH oder PuTTY verwenden. Weitere Informationen finden Sie unter Herstellung einer Verbindung zu Ihrer Linux-Instance mit SSH oder

Herstellung einer Verbindung zu Ihrer Linux-Instance von Windows mit PuTTY in der Amazon EC2-Dokumentation.

Schritt 2: Generieren oder Importieren eines privaten Schlüssels und SSL/TLS-Zertifikats

Um HTTPS zu aktivieren, benötigt Ihre Tomcat-Webserver-Anwendung einen privaten Schlüssel und ein entsprechendes SSL/TLS-Zertifikat. Um die Webserver-SSL/TLS-Auslagerung mit AWS CloudHSM verwenden zu können, müssen Sie den privaten Schlüssel in einem HSM in Ihrem AWS CloudHSM-Cluster speichern.

> **Note**
>
> Wenn Sie noch nicht über einen privaten Schlüssel und ein entsprechendes Zertifikat verfügen, generieren Sie einen privaten Schlüssel in einem HSM. Sie verwenden den privaten Schlüssel, um eine Zertifikatsignierungsanforderung (CSR) zu erstellen, mit der Sie das SSL/TLS-Zertifikat erstellen.

Sie erstellen eine lokale AWS CloudHSM-KeyStore-Datei, die einen Verweis auf Ihren privaten Schlüssel auf dem HSM und das zugehörige Zertifikat enthält. Ihr Webserver verwendet die AWS CloudHSM-KeyStore-Datei, um den privaten Schlüssel auf dem HSM während des SSL/TLS-Offloads zu identifizieren.

Themen

- Generieren eines privaten Schlüssels
- Generieren eines selbstsignierten Zertifikats

Generieren eines privaten Schlüssels

In diesem Abschnitt erfahren Sie, wie Sie mit dem KeyTool von JDK ein Schlüsselpaar generieren. Sobald Sie ein Schlüsselpaar im HSM generiert haben, können Sie es als KeyStore-Datei exportieren und das entsprechende Zertifikat generieren.

Je nach Anwendungsfall können Sie entweder ein RSA- oder ein EC-Schlüsselpaar generieren. Im Folgenden wird beschrieben, wie Sie ein RSA-Schlüsselpaar generieren.

Verwenden Sie den **genkeypair**-Befehl in KeyTool, um ein RSA-Schlüsselpaar zu erzeugen.

1. Nachdem Sie die *<VARIABLES>* unten durch Ihre spezifischen Daten ersetzt haben, verwenden Sie den folgenden Befehl, um eine Keystore-Datei mit dem Namen `jsse_keystore.keystore`, zu erzeugen, die einen Verweis auf Ihren privaten Schlüssel auf dem HSM enthält.

```
$ keytool -genkeypair -alias <UNIQUE ALIAS FOR KEYS> -keyalg <KEY ALGORITHM> -
keysize <KEY SIZE> -sigalg <SIGN ALGORITHM> \
        -keystore <PATH>/<JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
        -dname CERT_DOMAIN_NAME \
        -J-classpath '-J'$JAVA_LIB'/*:/opt/cloudhsm/java/*:./*' \
        -provider "com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider" \
        -providerpath "$CLOUDHSM_JCE_LOCATION" \
        -keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

   - ***<PATH>***: Der Pfad, in dem Sie Ihre Keystore-Datei generieren möchten.
   - ***<UNIQUE ALIAS FOR KEYS>***: Dies wird verwendet, um Ihren Schlüssel auf dem HSM eindeutig zu identifizieren. Dieser Alias wird als LABEL-Attribut für den Schlüssel festgelegt.
   - ***<KEY PASSWORD>***: Wir speichern den Verweis auf Ihren Schlüssel in der lokalen Keystore-Datei, und dieses Passwort schützt diese lokale Referenz.
   - ***<KEYSTORE PASSWORD>***: Dies ist das Passwort für Ihre lokale Keystore-Datei.
   - ***<JSSE KEYSTORE NAME>***: Name der Keystore-Datei.
   - ***<CERT DOMAIN NAME>***: X.500 Eindeutiger Name.
   - ***<KEY ALGORITHM>***: Schlüsselalgorithmus zur Generierung des Schlüsselpaars (z. B. RSA und EC).
   - ***<KEY SIZE>***: Schlüsselgröße zum Generieren des Schlüsselpaars (z. B. 2048, 3072 und 4096).
   - ***<SIGN ALGORITHM>***: Schlüsselgröße zum Generieren des Schlüsselpaars (z. B. SHA1withRSA, SHA224withRSA, SHA256withRSA, SHA384withRSA und SHA512withRSA).

2. Um zu bestätigen, dass der Befehl erfolgreich war, geben Sie den folgenden Befehl ein und überprüfen Sie, ob Sie erfolgreich ein RSA-Schlüsselpaar generiert haben.

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

Generieren eines selbstsignierten Zertifikats

Nachdem Sie zusammen mit der Keystore-Datei einen privaten Schlüssel generiert haben, können Sie diese Datei verwenden, um eine Certificate Signing Request (CSR) und ein Zertifikat zu generieren.

In einer Produktionsumgebung verwenden Sie in der Regel eine Zertifikatsstelle (CA) zum Erstellen eines Zertifikats aus einer CSR. Für eine Testumgebung ist keine CA erforderlich. Wenn Sie eine Zertifizierungsstelle verwenden, senden Sie ihnen die CSR-Datei und verwenden Sie das signierte SSL/TLS-Zertifikat, das sie Ihnen auf Ihrem Webserver für HTTPS zur Verfügung stellen.

Alternativ zur Verwendung einer CA können Sie mit dem KeyTool ein selbstsigniertes Zertifikat erstellen. Selbstsignierte Zertifikate sind nicht vertrauenswürdig für Browser und sollten in Produktionsumgebungen nicht verwendet werden. Sie können in Testumgebungen verwendet werden.

> **Warning**
>
> Selbstsignierte Zertifikate sollten nur in einer Testumgebung verwendet werden. Für eine Produktionsumgebung, verwenden Sie eine sicherere Methode, wie z. B. eine Zertifikatstelle, um ein Zertifikat zu erstellen.

Generieren eines Zertifikats

1. Besorgen Sie sich eine Kopie Ihrer Keystore-Datei, die in einem früheren Schritt generiert wurde.
2. Führen Sie den folgenden Befehl aus, um mit dem KeyTool eine Zertifikatsignierungsanfrage (CSR) zu erstellen.

```
$ keytool -certreq -keyalg RSA -alias unique_alias_for_key -file certreq.csr \
        -keystore <JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
        -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
        -keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

> **Note**
>
> Die Ausgabedatei der Zertifikatssignierungsanforderung ist `certreq.csr`.

Ein Zertifikat signieren

- Nachdem Sie die *<VARIABLES>* unten durch Ihre spezifischen Daten ersetzt haben, führen Sie den folgenden Befehl aus, um Ihre CSR mit Ihrem privaten Schlüssel auf Ihrem HSM zu signieren. Dadurch wird ein selbstsigniertes Zertifikat erstellt.

```
$ keytool -gencert -infile certreq.csr -outfile certificate.crt \
    -alias <UNIQUE ALIAS FOR KEYS> -keypass <KEY_PASSWORD> -
storepass <KEYSTORE_PASSWORD> -sigalg SIG_ALG \
    -storetype CLOUDHSM -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
    -keystore jsse_keystore.keystore
```

> **Note**
>
> `certificate.crt` ist das signierte Zertifikat, das den privaten Schlüssel des Alias verwendet.

Ein Zertifikat in Keystore importieren

- Nachdem Sie die *<VARIABLES>* unten durch Ihre spezifischen Daten ersetzt haben, führen Sie den folgenden Befehl aus, um ein signiertes Zertifikat als vertrauenswürdiges Zertifikat zu importieren. In diesem Schritt wird das Zertifikat in dem durch den Alias identifizierten Keystore-Eintrag gespeichert.

```
$ keytool -import -alias <UNIQUE ALIAS FOR KEYS> -keystore jsse_keystore.keystore \
    -file certificate.crt -storetype CLOUDHSM \
    -v -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
    -keypass <KEY PASSWORD> -storepass <KEYSTORE_PASSWORD>
```

Konvertiert ein Zertifikat in ein PEM

- Führen Sie den folgenden Befehl aus, um die signierte Zertifikatsdatei (.crt) in eine PEM zu konvertieren. Die PEM-Datei wird verwendet, um die Anfrage vom HTTP-Client zu senden.

```
$ openssl x509 -inform der -in certificate.crt -out certificate.pem
```

Nachdem Sie diese Schritte ausgeführt haben, fahren Sie mit Schritt 3: Konfigurieren des Webservers fort.

Schritt 3: Konfigurieren des Tomcat-Webservers

Aktualisieren Sie die Konfiguration Ihrer Webserver-Software, um das HTTPS-Zertifikat und die entsprechende PEM-Datei zu verwenden, die Sie im vorherigen Schritt erstellt haben. Denken Sie daran, Ihre vorhandenen Zertifikate und Schlüssel zu sichern, bevor Sie beginnen. Damit schließen Sie die Einrichtung Ihrer Linux-Webserver-Software für SSL/TLS-Auslagerung mit AWS CloudHSM ab. Weitere Informationen finden Sie in der Konfigurationsreferenz für Apache Tomcat 9.

Den Server beenden

- Nachdem Sie *<VARIABLES>* die folgenden Daten durch Ihre spezifischen Daten ersetzt haben, führen Sie den folgenden Befehl aus, um Tomcat Server zu beenden, bevor Sie die Konfiguration aktualisieren

  ```
  $ /<TOMCAT DIRECTORY>/bin/shutdown.sh
  ```

  - ***<TOMCAT DIRECTORY>***: Ihr Tomcat-Installationsverzeichnis.

Aktualisieren Sie den Klassenpfad von Tomcat

1. Stellen Sie eine Verbindung mit Ihrer Client-Instance her.
2. Suchen Sie den Tomcat-Installationsordner.
3. Nachdem Sie die *<VARIABLES>* unten durch Ihre spezifischen Daten ersetzt haben, fügen Sie mit dem folgenden Befehl die Java-Bibliothek und den Cloudhsm-Java-Pfad in den Tomcat-Klassenpfad ein, der sich in der Datei Tomcat/bin/catalina.sh befindet.

   ```
   $ sed -i 's@CLASSPATH="$CLASSPATH""$CATALINA_HOME"\/bin\/
   bootstrap.jar@CLASSPATH="$CLASSPATH""$CATALINA_HOME"\/bin\/bootstrap.jar:'"
           <JAVA LIBRARY>"'\/*:\/opt\/cloudhsm\/java\/*:.\/*@' <TOMCAT PATH> /bin/
   catalina.sh
   ```

   - ***<JAVA LIBRARY>***: Speicherort der Java JRE-Bibliothek.
   - ***<TOMCAT PATH>***: Tomcat-Installationsordner.

Fügen Sie der Serverkonfiguration einen HTTPS-Connector hinzu.

1. Gehen Sie zum Tomcat-Installationsordner.
2. Nachdem Sie die *<VARIABLES>* unten durch Ihre spezifischen Daten ersetzt haben, verwenden Sie den folgenden Befehl, um einen HTTPS-Connector hinzuzufügen, der die in den Voraussetzungen generierten Zertifikate verwendet:

```
$ sed -i '/<Connector port="8080"/i <Connector port=\"443\" maxThreads=\"200\"
 scheme=\"https\" secure=\"true\" SSLEnabled=\"true\" keystoreType=\"CLOUDHSM\"
 keystoreFile=\"
       <CUSTOM DIRECTORY>/<JSSE KEYSTORE NAME>.keystore\" keystorePass=\"<KEYSTORE
 PASSWORD>\" keyPass=\"<KEY PASSWORD>
       \" keyAlias=\"<UNIQUE ALIAS FOR KEYS>" clientAuth=\"false\" sslProtocol=
\"TLS\"/>' <TOMCAT PATH>/conf/server.xml
```

   - ***<CUSTOM DIRECTORY>***: Verzeichnis, in dem sich die Keystore-Datei befindet.
   - ***<JSSE KEYSTORE NAME>***: Name der Keystore-Datei.
   - ***<KEYSTORE PASSWORD>***: Dies ist das Passwort für Ihre lokale Keystore-Datei.
   - ***<KEY PASSWORD>***: Wir speichern den Verweis auf Ihren Schlüssel in der lokalen Keystore-Datei, und dieses Passwort schützt diese lokale Referenz.
   - ***<UNIQUE ALIAS FOR KEYS>***: Dies wird verwendet, um Ihren Schlüssel auf dem HSM eindeutig zu identifizieren. Dieser Alias wird als LABEL-Attribut für den Schlüssel festgelegt.
   - ***<TOMCAT PATH>***: Der Pfad zu Ihrem Tomcat-Ordner.

Starten des Servers

- Nachdem Sie die *<VARIABLES>* unten durch Ihre spezifischen Daten ersetzt haben, verwenden Sie den folgenden Befehl, um Tomcat Server zu starten:

```
$ /<TOMCAT DIRECTORY>/bin/startup.sh
```

> **Note**
>
> ***<TOMCAT DIRECTORY>*** ist der Name Ihres Tomcat-Installationsverzeichnisses.

Nachdem Sie Ihre Webserverkonfiguration aktualisiert haben, gehen Sie zu Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats.

Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats

Nachdem Sie Ihren Webserver für SSL/TLS-Offload mit AWS CloudHSM konfiguriert haben, fügen Sie Ihre Webserver-Instanz einer Sicherheitsgruppe hinzu, die eingehenden HTTPS-Verkehr zulässt. Dadurch können Clients, wie z. B. Webbrowser, eine HTTPS-Verbindung mit Ihrem Webserver herstellen. Stellen Sie dann eine HTTPS-Verbindung zu Ihrem Webserver her und überprüfen Sie, ob er das Zertifikat verwendet, das Sie für SSL/TLS-Offload mit AWS CloudHSM konfiguriert haben.

Themen

- Aktivieren von eingehenden HTTPS-Verbindungen
- Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Aktivieren von eingehenden HTTPS-Verbindungen

Zum Herstellen einer Verbindung zu Ihrem Webserver von einem Client (z. B. ein Webbrowser) aus, erstellen Sie eine Sicherheitsgruppe, die eingehende HTTPS-Verbindungen zulässt. Insbesondere sollten eingehende TCP-Verbindungen auf Port 443 erlaubt werden. Weisen Sie diese Sicherheitsgruppe Ihrem Webserver zu.

So erstellen Sie eine Sicherheitsgruppe für HTTPS und weisen sie Ihrem Webserver zu

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie im Navigationsbereich Sicherheitsgruppen aus.
3. Wählen Sie Sicherheitsgruppe erstellen.
4. Führen Sie für Sicherheitsgruppe erstellen die folgenden Schritte aus:
    a. Geben Sie in das Feld Sicherheitsgruppenname einen Namen für die Sicherheitsgruppe ein, die Sie erstellen.
    b. (Optional) Geben Sie eine Beschreibung der Sicherheitsgruppe ein, die Sie erstellen.
    c. Wählen Sie für VPC die VPC aus, die Ihre Amazon-EC2-Instance enthält.
    d. Wählen Sie Regel hinzufügen aus.
    e. Wählen Sie im Drop-down-Fenster für Typ die Option HTTPS aus.
    f. Geben Sie für Quelle einen Quellspeicherort ein.
    g. Wählen Sie Sicherheitsgruppe erstellen.

5. Wählen Sie im Navigationsbereich Instances aus.
6. Aktivieren Sie das Kontrollkästchen neben Ihrer Webserver-Instance.
7. Wählen Sie das Drop-down-Menü Aktionen oben auf der Seite. Wählen Sie Sicherheit und dann Sicherheitsgruppen ändern aus.
8. Wählen Sie unter Zugeordnete Sicherheitsgruppen das Suchfeld aus und wählen Sie die Sicherheitsgruppe, die Sie für HTTPS erstellt haben, aus. Wählen Sie dann Sicherheitsgruppen hinzufügen aus.
9. Wählen Sie Speichern.

Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Nachdem Sie den Webserver zu einer Sicherheitsgruppe hinzugefügt haben, können Sie überprüfen, ob beim SSL/TLS-Offload Ihr selbstsigniertes Zertifikat verwendet wird. Sie können dazu einen Webbrowser oder ein Tool wie OpenSSL s_client nutzen.

So überprüfen Sie die SSL/TLS-Auslagerung mit einem Webbrowser

1. Verwenden Sie einen Web-Browser, um eine Verbindung zum Webserver unter Verwendung des öffentlichen DNS-Namen oder der IP-Adresse des Servers herzustellen. Stellen Sie sicher, dass die URL in die Adresszeile mit https:// beginnt. Zum Beispiel **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

   > **Tip**
   >
   > Sie können einen DNS-Service wie Amazon Route 53 nutzen, um den Domainnamen Ihrer Website (beispielsweise https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter Routing des Datenverkehrs zu einer Amazon-EC2-Instance im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:

   - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter Zertifikat anzeigen an.
   - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter Sicherheitsprobleme verstehen an.

Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.

3. Stellen Sie sicher, dass das SSL/TLS-Zertifikat dasjenige ist, das Sie für die Nutzung in Ihrem Webserver konfiguriert haben.

So überprüfen Sie die SSL/TLS-Auslagerung mit OpenSSL s_client

1. Führen Sie den folgenden OpenSSL-Befehl aus, um mittels HTTPS eine Verbindung zu Ihrem Webserver herzustellen. Ersetzen Sie *server name>* durch den öffentlichen DNS-Namen oder die IP-Adresse Ihres Webservers.

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Sie können einen DNS-Service wie Amazon Route 53 nutzen, um den Domainnamen Ihrer Website (beispielsweise https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter Routing des Datenverkehrs zu einer Amazon-EC2-Instance im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass das SSL/TLS-Zertifikat dasjenige ist, das Sie für die Nutzung in Ihrem Webserver konfiguriert haben.

Sie verfügen jetzt über eine mit HTTPS gesicherte Website. Der private Schlüssel für den Webserver ist in einem HSM in Ihrem AWS CloudHSM-Cluster gespeichert.

Informationen zum Hinzufügen eines Load Balancers finden Sie unter Load Balancer mit Elastic Load Balancing hinzufügen (optional).

Verwenden von IIS mit CNG für SSL/TLS-Offload unter Windows

In dieser Anleitung ist Schritt für Schritt erläutert, wie Sie die SSL/TLS-Auslagerung mit AWS CloudHSM auf einem Windows-Webserver einrichten.

Themen

Übersicht

Unter Windows bietet die Webserveranwendung Internet Information Services (IIS) für Windows Server native Unterstützung für HTTPS. Der AWS CloudHSM-Schlüsselspeicher-Provider (KSP, Key Storage Provider) für Microsoft Cryptography API: Next Generation (CNG) stellt die Schnittstelle bereit, um IIS zu ermöglichen HSMs in Ihrem Cluster für verschlüsselte Auslagerungen und Schlüsselspeicherung zu verwenden. Der AWS CloudHSM-KSP bildet die Brücke, die IIS mit Ihrem AWS CloudHSM-Cluster verbindet.

In diesem Tutorial erfahren Sie, wie Sie folgende Aufgaben ausführen:

- Installieren der Web-Server-Software auf einer Amazon EC2-Instance
- Konfigurieren Sie die Webserver-Software so, dass sie HTTPS mit einem privaten Schlüssel unterstützt, der in Ihrem AWS CloudHSM-Cluster gespeichert ist.
- (Optional) Verwenden Sie Amazon EC2, um eine zweite Webserver-Instance zu erstellen, und Elastic Load Balancing, um einen Load Balancer zu erstellen. Mit einem Load Balancer kann die Leistung durch Verteilung der Arbeitslast auf mehrere Server gesteigert werden. Sie kann auch für Redundanz und eine höhere Verfügbarkeit sorgen, falls ein oder mehrere Webserver ausfallen.

Wenn Sie bereit sind, sehen Sie sich Schritt 1: Einrichten der Voraussetzungen an.

Schritt 1: Einrichten der Voraussetzungen

Für die Webserver-SSL/TLS-Auslagerung mit AWS CloudHSM ist Folgendes erforderlich:

- Einen aktiven AWS CloudHSM-Cluster mit mindestens einem HSM.
- Eine Amazon EC2-Instance, auf der ein Windows-Betriebssystem ausgeführt wird und die folgende Software installiert ist:
  - Die AWS CloudHSM-Client-Software für Windows
  - Internet Information Services (IIS) für Windows Server

- Ein Crypto-Benutzer (CU), der den privaten Schlüssel des Webservers auf dem HSM besitzen und verwalten soll.

> **Note**
>
> In diesem Tutorial wird Microsoft Windows Server 2016 verwendet. Microsoft Windows Server 2012 wird ebenfalls unterstützt, Microsoft Windows Server 2012 R2 jedoch nicht.

So richten Sie eine Windows-Webserver-Instance auf dem HSM ein und erstellen einen CU

1. Führen Sie die Schritte unter Erste Schritte aus. Beim Starten des Amazon EC2-Clients wählen Sie ein Windows Server 2016- oder Windows Server 2012-AMI aus. Wenn Sie diese Schritte ausgeführt haben, verfügen Sie über einen aktiven Cluster mit mindestens einem HSM. Sie verfügen auch über eine Amazon EC2-Client-Instance, auf der Windows Server ausgeführt wird und auf der die AWS CloudHSM-Client-Software für Windows installiert ist.
2. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter Hinzufügen eines HSM.
3. Stellen Sie eine Verbindung zu Ihrem Windows-Server her. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
4. Verwenden Sie die CloudHSM-CLI, um einen Crypto-Benutzer (CU) zu erstellen. Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen diese im nächsten Schritt.

   > **Note**
   >
   > Informationen zum Erstellen eines Benutzers finden Sie unter HSM-Benutzer mit der CloudHSM-CLI verwalten.

5. Legen Sie die Anmeldeinformationen für das HSM fest. Verwenden Sie hierfür den im vorherigen Schritt erstellten CU-Benutzernamen und das entsprechende Passwort.
6. Wenn Sie in Schritt 5 die Windows-Anmeldeinformationsverwaltung verwendet haben, um HSM-Anmeldeinformationen festzulegen, laden Sie `psexec.exe` von SysInternals herunter, um den folgenden Befehl als NT Authority\SYSTEM auszuführen:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
 --username <USERNAME> --password <PASSWORD>
```

Ersetzen Sie *<username>* und *<password>* durch die HSM-Anmeldeinformationen.

So installieren Sie IIS auf Ihrem Windows-Server

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
2. Starten Sie auf Ihrem Windows-Server Server Manager.
3. Wählen Sie im Dashboard Server-Manager die Option Rollen und Features hinzufügen.
4. Lesen Sie die Informationen unter Before you begin (Zur Vorbereitung) und klicken Sie dann auf Weiter.
5. Wählen Sie unter Installationsart die Rollenbasierte oder funktionsbasierte Installation. Wählen Sie anschließend Weiter.
6. Wählen Sie unter Serverauswahl die Option Einen Server aus dem Serverpool auswählen. Wählen Sie anschließend Weiter.
7. Gehen Sie für Serverrollen wie folgt vor:
   a. Wählen Sie Web Server (IIS) (Webserver (IIS)) aus.
   b. Klicken Sie für Add features that are required für Web Server (IIS) (Für Webserver (IIS) erforderliche Funktionen hinzufügen) auf Add Features (Funktionen hinzufügen).
   c. Klicken Sie auf Weiter, um das Auswählen von Serverrollen abzuschließen.
8. Für Features (Funktionen), übernehmen Sie die Standardeinstellungen. Wählen Sie anschließend Weiter.
9. Lesen Sie die Informationen unter Web Server Role (IIS) (Webserverrolle (IIS)). Wählen Sie anschließend Weiter.
10. Übernehmen Sie für Select role services (Rollenservices auswählen) die Standardwerte oder ändern Sie die Einstellungen. Wählen Sie anschließend Weiter.
11. Lesen Sie unter Confirmation (Bestätigung) die Informationen zur Bestätigung. Wählen Sie anschließend Installieren.
12. Nach abgeschlossener Installation klicken Sie auf Close (Schließen).

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Schritt 2: Erstellen einer Zertifikatsignierungsanforderung (CSR) und eines Zertifikats fort.

Schritt 2: Erstellen einer Zertifikatsignierungsanforderung (CSR) und eines Zertifikats

Um HTTPS zu aktivieren, benötigt Ihr Webserver ein SSL/TLS-Zertifikat sowie einen zugehörigen privaten Schlüssel. Um die SSL/TLS-Auslagerung mit AWS CloudHSM verwenden zu können, müssen Sie den privaten Schlüssel im HSM in Ihrem AWS CloudHSM-Cluster speichern. Verwenden Sie dazu den AWS CloudHSM-Schlüsselspeicher-Anbieter (Key Storage Provider, KSP) für die Microsoft Cryptography API: Next Generation (CNG), um eine Zertifikatsignierungsanforderung (CSR) zu erstellen. Senden Sie die CSR dann an eine Zertifizierungsstelle (Certificate Authority, CA), um die CSR signieren zu lassen und ein Zertifikat zu erhalten.

Themen

- Erstellen einer CSR
- Erhalten und Importieren eines signierten Zertifikats

Erstellen einer CSR

Erstellen Sie mithilfe des AWS CloudHSM-KSP auf Ihrem Windows-Server eine CSR.

So erstellen Sie eine CSR

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
2. Verwenden Sie den folgenden Befehl, um den AWS CloudHSM-Client-Daemon zu starten.

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client start
   ```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

3. Verwenden Sie auf Ihrem Windows-Server einen Texteditor, um eine Zertifikatanforderungsdatei namens `IISCertRequest.inf` zu erstellen. Den Inhalt einer Beispieldatei `IISCertRequest.inf` sehen Sie nachfolgend. Weitere Informationen zu den Abschnitten,

Schlüsseln und Werten in dieser Datei finden Sie in der Microsoft-Dokumentation. Ändern Sie den Wert `ProviderName` nicht.

```
[Version]
Signature = "$Windows NT$"
[NewRequest]
Subject = "CN=example.com,C=US,ST=Washington,L=Seattle,O=ExampleOrg,OU=WebServer"
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "Cavium Key Storage Provider"
KeyUsage = 0xf0
MachineKeySet = True
[EnhancedKeyUsageExtension]
OID=1.3.6.1.5.5.7.3.1
```

4. Verwenden Sie den Windows-Befehl „certreq“, um eine CSR aus der Datei `IISCertRequest.inf` zu erstellen, die Sie im vorherigen Schritt erstellt haben. Im folgenden Beispiel wird die CSR in einer Datei namens `IISCertRequest.csr` gespeichert. Wenn Sie einen anderen Dateinamen für Ihre Zertifikatanforderungsdatei verwenden, ersetzen Sie *`IISCertRequest.inf`* durch den entsprechenden Dateinamen. Sie können *`IISCertRequest.csr`* optional durch einen anderen Dateinamen für Ihre CSR-Datei ersetzen.

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr
        SDK Version: 2.03

CertReq: Request Created
```

Die Datei `IISCertRequest.csr` enthält Ihre CSR. Sie benötigen diese CSR, um ein signiertes Zertifikat zu erhalten.

Erhalten und Importieren eines signierten Zertifikats

In einer Produktionsumgebung verwenden Sie in der Regel eine Zertifikatsstelle (CA) zum Erstellen eines Zertifikats aus einer CSR. Für eine Testumgebung ist keine CA erforderlich. Wenn Sie eine CA verwenden, senden Sie die CSR-Datei (`IISCertRequest.csr`) an sie und verwenden die CA, um ein signiertes SSL/TLS-Zertifikat zu erstellen.

Als Alternative zur Verwendung einer Zertifizierungsstelle können Sie ein Tool wie OpenSSL verwenden, um ein selbstsigniertes Zertifikat zu erstellen.

> **Warning**
>
> Selbstsignierte Zertifikate sind nicht vertrauenswürdig für Browser und sollten in Produktionsumgebungen nicht verwendet werden. Sie können in Testumgebungen verwendet werden.

Nachfolgend wird beschrieben, wie Sie ein selbstsigniertes Zertifikat erstellen und dieses für die Signierung der CSR Ihres Webservers verwenden.

So erstellen Sie ein selbstsigniertes Zertifikat

1. Verwenden Sie den folgenden OpenSSL-Befehl, um einen privaten Schlüssel zu erstellen. Optional können Sie *SelfSignedCA.key* durch den Dateinamen ersetzen, der in Ihrem privaten Schlüssel enthalten ist.

   ```
   openssl genrsa -aes256 -out SelfSignedCA.key 2048
   Generating RSA private key, 2048 bit long modulus
   ..............................................................+++
   .....................................+++
   e is 65537 (0x10001)
   Enter pass phrase for SelfSignedCA.key:
   Verifying - Enter pass phrase for SelfSignedCA.key:
   ```

2. Verwenden Sie den folgenden OpenSSL-Befehl, um ein selbstsigniertes Zertifikat mit dem privaten Schlüssel zu erstellen, den Sie im vorherigen Schritt erstellt haben. Hierbei handelt es sich um einen interaktiven Befehl. Lesen Sie die Anweisungen auf dem Bildschirm und befolgen Sie die Eingabeaufforderungen. Ersetzen Sie *SelfSignedCA.key* durch den Namen der Datei, die Ihren gefälschten privaten PEM-Schlüssel enthält (falls abweichend). Optional können Sie *SelfSignedCA.crt* durch den Dateinamen ersetzen, der in Ihrem selbstsignierten Zertifikat enthalten ist.

   ```
   openssl req -new -x509 -days 365 -key SelfSignedCA.key -out SelfSignedCA.crt
   Enter pass phrase for SelfSignedCA.key:
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   ```

```
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

So verwenden Sie ein selbstsigniertes Zertifikat, um die CSR Ihres Webservers zu signieren

- Verwenden Sie den folgenden OpenSSL-Befehl, um mit Ihrem privaten Schlüssel und Ihrem selbstsignierten Zertifikat die CSR zu signieren. Ersetzen Sie die folgenden Namen durch die Namen der Dateien, die die jeweiligen Daten enthalten, sofern diese abweichen.
  - *IISCertRequest.csr* – Der Name der Datei, die die CSR Ihres Webservers enthält
  - *SelfSignedCA.crt* – Der Name der Datei, die Ihr selbstsigniertes Zertifikat enthält
  - *SelfSignedCA.key* – Der Name der Datei, die Ihren privaten Schlüssel enthält
  - *IISCert.crt* – Der Name der Datei, die das selbstsignierte Zertifikat Ihres Webservers enthalten soll

```
openssl x509 -req -days 365 -in IISCertRequest.csr \
                            -CA SelfSignedCA.crt \
                            -CAkey SelfSignedCA.key \
                            -CAcreateserial \
                            -out IISCert.crt
Signature ok
subject=/ST=IIS-HSM/L=IIS-HSM/OU=IIS-HSM/O=IIS-HSM/CN=IIS-HSM/C=IIS-HSM
Getting CA Private Key
Enter pass phrase for SelfSignedCA.key:
```

Nachdem Sie den vorherigen Schritt abgeschlossen haben, haben Sie ein signiertes Zertifikat für Ihren Webserver (`IISCert.crt`) sowie ein selbstsigniertes Zertifikat (`SelfSignedCA.crt`). Wenn Sie diese Dateien haben, gehen Sie zu Schritt 3: Konfigurieren des Webservers.

Schritt 3: Konfigurieren des Webservers

Aktualisieren Sie die Konfiguration Ihrer IIS-Website, um das HTTPS-Zertifikat zu verwenden, das Sie am Ende des vorherigen Schritts erstellt haben. Damit schließen Sie die Einrichtung Ihrer Windows-Webserver-Software (IIS) für SSL/TLS-Auslagerung mit AWS CloudHSM ab.

Wenn Sie ein selbstsigniertes Zertifikat zum Signieren Ihrer CSR verwendet haben, müssen Sie zunächst das selbstsignierte Zertifikat in Windows unter „Vertrauenswürdige Stammzertifizierungsstellen“ importieren.

So importieren Sie Ihr selbstsigniertes Zertifikat in Windows unter „Vertrauenswürdige Stammzertifizierungsstellen“

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
2. Kopieren Sie Ihr selbstsigniertes Zertifikat auf Ihren Windows-Server.
3. Öffnen Sie auf Ihrem Windows-Server die Systemsteuerung.
4. Geben Sie für Search Control Panel (Systemsteuerung durchsuchen) **`certificates`** ein. Klicken Sie dann auf Computerzertifikate verwalten.
5. Doppelklicken Sie im Fenster Zertifikate - Lokaler Computer auf Vertrauenswürdige Stammzertifizierungsstellen.
6. Klicken Sie mit der rechten Maustaste auf Zertifikate, Alle Aufgaben und Importieren.
7. Klicken Sie im Zertifikatimport-Assistent auf Weiter.
8. Klicken Sie auf Durchsuchen und wählen Sie Ihr selbstsigniertes Zertifikat aus. Wenn Sie Ihr selbstsigniertes Zertifikat nach der Anleitung im vorherigen Schritt dieser Anleitung erstellt haben, hat Ihr selbstsigniertes Zertifikat den Namen `SelfSignedCA.crt`. Klicken Sie auf Open.
9. Wählen Sie Weiter.
10. Klicken Sie für Zertifikatspeicher auf Alle Zertifikate in folgendem Speicher speichern. Stellen Sie dann sicher, dass für Zertifikatspeicher Vertrauenswürdige Stammzertifizierungsstellen ausgewählt ist.
11. Wählen Sie Weiter und anschließend Beenden aus.

So aktualisieren Sie die Konfiguration Ihrer IIS-Website

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
2. Starten Sie den AWS CloudHSM-Client-Daemon.
3. Kopieren Sie das signierte Zertifikat Ihres Webservers – dasjenige, das Sie am Ende des vorherigen Schritts dieses Tutorials erstellt haben – auf Ihren Windows-Server.
4. Führen Sie auf Ihrem Windows-Server den Windows-Befehl „certreq“ aus, um das signierte Zertifikat zu übernehmen, wie im folgenden Beispiel gezeigt. Ersetzen Sie *IISCert.crt* durch den Namen der Datei, die das signierte Zertifikat Ihres Webservers enthält.

   ```
   C:\>certreq -accept IISCert.crt
           SDK Version: 2.03
   ```
5. Starten Sie auf Ihrem Windows-Server Server Manager.
6. Klicken Sie rechts oben im Dashboard Server-Manager auf Extras, Internet Information Services (IIS)-Manager.
7. Doppelklicken Sie im Fenster Internet Information Services (IIS)-Manager auf den Servernamen. Doppelklicken Sie dann auf Websites. Wählen Sie Ihre Website aus.
8. Klicken Sie auf SSL-Einstellungen. Klicken Sie dann rechts im Fenster auf Bindungen.
9. Klicken Sie im Fenster Websitebindungen auf Hinzufügen.
10. Wählen Sie für Typ die Option HTTPS aus. Wählen Sie für SSL-Zertifikat das HTTPS-Zertifikat aus, das Sie am Ende des vorherigen Schritts dieser Anleitung erstellt haben.

    > **Note**
    >
    > Wenn während dieser Zertifikat-Bindung ein Fehler auftritt, starten Sie Ihren Server neu und wiederholen Sie diesen Schritt.
11. Wählen Sie OK.

Nachdem Sie die Konfiguration Ihrer Website aktualisiert haben, gehen Sie zu Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats.

Schritt 4: Aktivieren von HTTPS-Datenverkehr und Verifizieren des Zertifikats

Nachdem Sie Ihren Webserver für SSL/TLS-Offload mit AWS CloudHSM konfiguriert haben, fügen Sie Ihre Webserver-Instanz einer Sicherheitsgruppe hinzu, die eingehenden HTTPS-Verkehr zulässt. Dadurch können Clients, wie z. B. Webbrowser, eine HTTPS-Verbindung mit Ihrem Webserver herstellen. Stellen Sie dann eine HTTPS-Verbindung zu Ihrem Webserver her und überprüfen Sie, ob er das Zertifikat verwendet, das Sie für SSL/TLS-Offload mit AWS CloudHSM konfiguriert haben.

Themen

Aktivieren von eingehenden HTTPS-Verbindungen

Zum Herstellen einer Verbindung zu Ihrem Webserver von einem Client (z. B. ein Webbrowser) aus, erstellen Sie eine Sicherheitsgruppe, die eingehende HTTPS-Verbindungen zulässt. Insbesondere sollten eingehende TCP-Verbindungen auf Port 443 erlaubt werden. Weisen Sie diese Sicherheitsgruppe Ihrem Webserver zu.

So erstellen Sie eine Sicherheitsgruppe für HTTPS und weisen sie Ihrem Webserver zu

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie im Navigationsbereich Sicherheitsgruppen aus.
3. Wählen Sie Sicherheitsgruppe erstellen.
4. Führen Sie für Sicherheitsgruppe erstellen die folgenden Schritte aus:
    a. Geben Sie in das Feld Sicherheitsgruppenname einen Namen für die Sicherheitsgruppe ein, die Sie erstellen.
    b. (Optional) Geben Sie eine Beschreibung der Sicherheitsgruppe ein, die Sie erstellen.
    c. Wählen Sie für VPC die VPC aus, die Ihre Amazon-EC2-Instance enthält.
    d. Wählen Sie Regel hinzufügen aus.
    e. Wählen Sie im Drop-down-Fenster für Typ die Option HTTPS aus.
    f. Geben Sie für Quelle einen Quellspeicherort ein.
    g. Wählen Sie Sicherheitsgruppe erstellen.
5. Wählen Sie im Navigationsbereich Instances aus.

6. Aktivieren Sie das Kontrollkästchen neben Ihrer Webserver-Instance.
7. Wählen Sie das Drop-down-Menü Aktionen oben auf der Seite. Wählen Sie Sicherheit und dann Sicherheitsgruppen ändern aus.
8. Wählen Sie unter Zugeordnete Sicherheitsgruppen das Suchfeld aus und wählen Sie die Sicherheitsgruppe, die Sie für HTTPS erstellt haben, aus. Wählen Sie dann Sicherheitsgruppen hinzufügen aus.
9. Wählen Sie Speichern.

Verifizieren, dass HTTPS das konfigurierte Zertifikat verwendet

Nachdem Sie den Webserver zu einer Sicherheitsgruppe hinzugefügt haben, können Sie überprüfen, ob beim SSL/TLS-Offload Ihr selbstsigniertes Zertifikat verwendet wird. Sie können dazu einen Webbrowser oder ein Tool wie OpenSSL s_client nutzen.

So überprüfen Sie die SSL/TLS-Auslagerung mit einem Webbrowser

1. Verwenden Sie einen Web-Browser, um eine Verbindung zum Webserver unter Verwendung des öffentlichen DNS-Namen oder der IP-Adresse des Servers herzustellen. Stellen Sie sicher, dass die URL in die Adresszeile mit https:// beginnt. Zum Beispiel **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

   > **Tip**
   >
   > Sie können einen DNS-Service wie Amazon Route 53 nutzen, um den Domainnamen Ihrer Website (beispielsweise https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter Routing des Datenverkehrs zu einer Amazon-EC2-Instance im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:

   - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter Zertifikat anzeigen an.
   - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter Sicherheitsprobleme verstehen an.

Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.

3. Stellen Sie sicher, dass das SSL/TLS-Zertifikat dasjenige ist, das Sie für die Nutzung in Ihrem Webserver konfiguriert haben.

So überprüfen Sie die SSL/TLS-Auslagerung mit OpenSSL s_client

1. Führen Sie den folgenden OpenSSL-Befehl aus, um mittels HTTPS eine Verbindung zu Ihrem Webserver herzustellen. Ersetzen Sie *server name>* durch den öffentlichen DNS-Namen oder die IP-Adresse Ihres Webservers.

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Sie können einen DNS-Service wie Amazon Route 53 nutzen, um den Domainnamen Ihrer Website (beispielsweise https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter Routing des Datenverkehrs zu einer Amazon-EC2-Instance im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass das SSL/TLS-Zertifikat dasjenige ist, das Sie für die Nutzung in Ihrem Webserver konfiguriert haben.

Sie verfügen jetzt über eine mit HTTPS gesicherte Website. Der private Schlüssel für den Webserver ist in einem HSM in Ihrem AWS CloudHSM-Cluster gespeichert.

Informationen zum Hinzufügen eines Load Balancers finden Sie unter Load Balancer mit Elastic Load Balancing hinzufügen (optional).

Load Balancer mit Elastic Load Balancing hinzufügen (optional)

Nachdem Sie die SSL/TLS-Auslagerung für einen Webserver eingerichtet haben, können Sie weitere Webserver sowie einen Elastic Load Balancing Load Balancer einrichten, der den HTTPS-Datenverkehr zu den Webservern umleitet. Ein Load Balancer kann die Auslastung Ihrer einzelnen Webserver reduzieren, indem der Verkehr auf zwei oder mehr Webserver verteilt wird. Er kann

zudem die Verfügbarkeit Ihrer Website erhöhen, da er den Zustand Ihrer Webserver überwacht und den Datenverkehr nur an stabile Server weiterleitet. Wenn ein Webserver ausfällt, beendet der Load Balancer automatisch die Weiterleitung des Datenverkehrs an diesen.

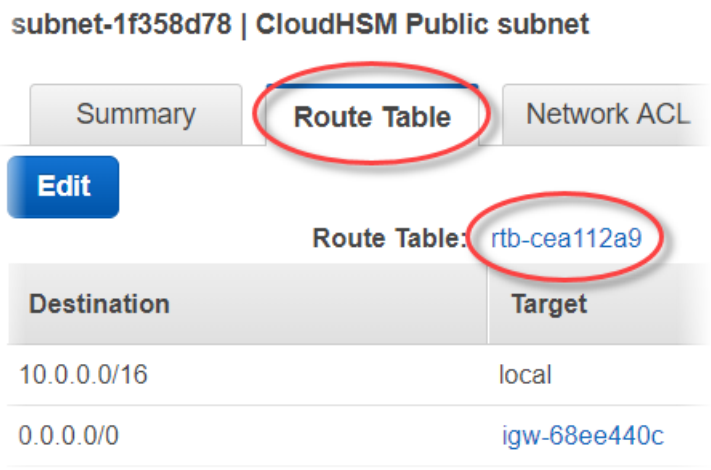
Themen

Erstellen eines Subnetzes für einen zweiten Webserver

Bevor Sie einen weiteren Webserver erstellen können, müssen Sie ein neues Subnetz in derselben VPC erstellen, die Ihren vorhandenen Webserver und Ihren AWS CloudHSM-Cluster enthält.

So erstellen Sie ein neues Subnetz

1. Öffnen Sie den Abschnitt Subnetze der Amazon VPC-Konsole.
2. Wählen Sie Create Subnet aus.
3. Führen Sie im Dialogfeld Create Subnet Folgendes aus:
    a. Geben Sie unter Name tag(Namens-Tag) einen Namen für Ihr Subnetz ein.
    b. Wählen Sie für VPC die AWS CloudHSM-VPC, die Ihren bestehenden Webserver und AWS CloudHSM-Cluster enthält.
    c. Wählen Sie als Availability Zone eine Availability Zone aus, die sich von der unterscheidet, die Ihren vorhandenen Webserver enthält.
    d. Geben Sie unter IPv4 CIDR block (IPv4 CIDR-Block) den CIDR-Block für das Subnetz ein. Geben Sie beispielsweise **`10.0.10.0/24`** ein.
    e. Wählen Sie Ja, erstellen aus.
4. Aktivieren Sie das Kontrollkästchen neben dem öffentlichen Subnetz, das Ihren vorhandenen Webserver enthält. Dieses unterscheidet sich von dem öffentlichen Subnetz, das Sie im vorherigen Schritt erstellt haben.
5. Klicken Sie im Inhaltsbereich auf die Registerkarte Route Table (Routing-Tabelle). Wählen Sie dann den Link für die Routing-Tabelle aus.



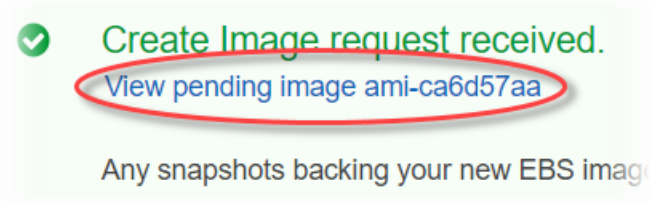
6. Aktivieren Sie das Kontrollkästchen neben der Routing-Tabelle.
7. Wählen Sie die Registerkarte Subnet Associations (Subnetz-Verknüpfungen) aus. Wählen Sie dann Bearbeiten aus.
8. Wählen Sie das Kontrollkästchen neben dem öffentlichen Subnetz aus, das Sie zuvor erstellt haben. Wählen Sie dann Speichern.

Erstellen des zweiten Webservers

Führen Sie die folgenden Schritte aus, um einen zweiten Webserver mit der gleichen Konfiguration wie der des vorhandenen Webservers zu erstellen.

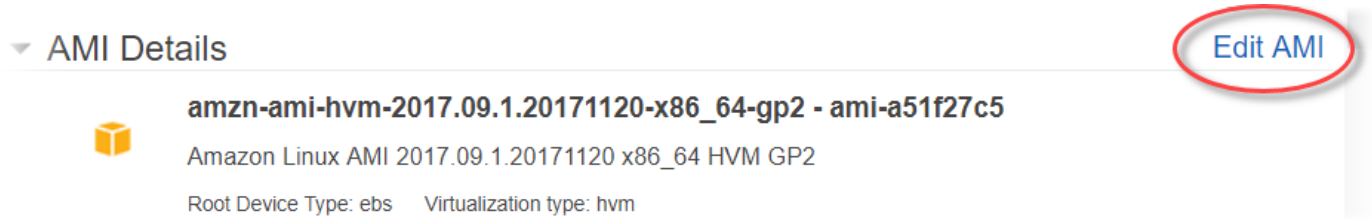
So erstellen Sie einen zweiten Webserver

1. Öffnen Sie den Bereich Instances in der Amazon-EC2-Konsole unter.
2. Aktivieren Sie das Kontrollkästchen neben Ihrer vorhandenen Webserver-Instance.
3. Wählen Sie Aktionen, Image und dann Image erstellen.
4. Führen Sie im Dialogfeld Create Image (Abbild erstellen) die folgenden Schritte aus:
    a. Geben Sie für Image name (Image-Name) einen eindeutigen Namen für das Image ein.
    b. Geben Sie unter Image description (Image-Beschreibung) eine Beschreibung für das Image ein.
    c. Wählen Sie Image erstellen aus. Mit dieser Aktion wird Ihr vorhandener Webserver neu gestartet.
    d. Wählen Sie den Link View pending image ami-***<AMI ID>*** aus.



Beachten Sie den Abbildstatus in der Status-Spalte. Lautet der Abbildstatus available (verfügbar), dies kann einige Minuten dauern, fahren Sie mit dem nächsten Schritt fort.

5. Wählen Sie im Navigationsbereich Instances aus.
6. Aktivieren Sie das Kontrollkästchen neben Ihrem vorhandenen Webserver.
7. Wählen Sie Aktionen und dann Weitere solche starten aus.
8. Wählen Sie AMI bearbeiten aus.



9. Wählen Sie im linken Navigationsbereich die Option My AMIs (Meine AMIs) aus. Löschen Sie anschließend den Text im Suchfeld.
10. Wählen Sie neben Ihrem Webserverabbild Auswählen aus.
11. Wählen Sie Ja, ich möchte mit diesem AMI fortfahren (***<image name>*** - ami-***<AMI ID>***).
12. Wählen Sie Weiter.
13. Wählen Sie einen Instance-Typen und danach Weiter: Instance-Details konfigurieren aus.
14. Führen Sie für Step 3: Configure Instance Details (Schritt 3: Instance-Details konfigurieren) Folgendes aus:

    a. Wählen Sie für Netzwerk die VPC mit Ihrem vorhandenen Webserver aus.

    b. Wählen Sie für Subnetz das öffentliche Subnetz aus, das Sie für den zweiten Webserver erstellt haben.

    c. Wählen Sie für Öffentliche IP automatisch zuweisen Aktivieren aus.

    d. Ändern Sie die verbleibenden Instance-Details wie gewünscht. Wählen Sie dann Weiter: Speicher hinzufügen aus.
15. Ändern Sie die Speichereinstellungen wie gewünscht. Wählen Sie dann Weiter: Tags hinzufügen aus.

16. Bearbeiten Sie Tags nach Bedarf oder fügen Sie diese hinzu. Wählen Sie dann Als Nächstes: Sicherheitsgruppe konfigurieren aus.
17. Führen Sie für Schritt 6: Sicherheitsgruppe konfigurieren die folgenden Schritte aus:

    a. Wählen Sie für Sicherheitsgruppe zuweisen die Option Existierende Sicherheitsgruppe auswählen aus.

    b. Aktivieren Sie das Kontrollkästchen neben der Sicherheitsgruppe mit dem Namen cloudhsm-***<cluster ID>***-sg. AWS CloudHSM hat diese Sicherheitsgruppe in Ihrem Namen erstellt, als Sie den Cluster angelegt haben. Sie müssen diese Sicherheitsgruppe auswählen, damit die Webserver-Instance eine Verbindung mit den HSMs im Cluster herstellen kann.

    c. Aktivieren Sie das Kontrollkästchen neben der Sicherheitsgruppe, die eingehenden HTTPS-Datenverkehr zulässt. Sie haben diese Sicherheitsgruppe zuvor erstellt.

    d. (Optional) Aktivieren Sie das Kontrollkästchen neben einer Sicherheitsgruppe, die eingehenden SSH- (für Linux) oder RDP-Datenverkehr (für Windows) aus Ihrem Netzwerk zulässt. Das heißt, die Sicherheitsgruppe muss ein- und ausgehenden TCP-Datenverkehr auf Port 22 (für SSH unter Linux) bzw. Port 3389 (für RDP unter Windows) zulassen. Andernfalls können Sie keine Verbindung zu Ihrer Client-Instance herstellen. Wenn Sie keine solche Sicherheitsgruppe haben, müssen Sie sie erstellen und Ihrer Client-Instance zu einem späteren Zeitpunkt hinzufügen.

    Klicken Sie auf Review and Launch.
18. Überprüfen Sie die Instance-Details. Wählen Sie anschließend Start aus.
19. Legen Sie fest, ob Ihre Instance mit einem vorhandenen Schlüsselpaar oder ohne Schlüsselpaar gestartet werden soll, oder ob ein neues Schlüsselpaar erstellt werden soll.

    - Um ein vorhandenes Schlüsselpaar zu verwenden, führen Sie die folgenden Schritte aus:
      1. Wählen Sie Vorhandenes Schlüsselpaar auswählen aus.
      2. Wählen Sie für Schlüsselpaar auswählen das zu verwendende Schlüsselpaar aus.
      3. Markieren Sie das Kontrollkästchen neben I acknowledge that I have access to the selected private key file (***private key file name***.pem), and that without this file, I won't be able to log into my instance. (Ich bestätige, dass ich Zugriff auf die ausgewählte private Schlüsseldatei habe und dass ich mich ohne diese Datei nicht bei meiner Instance anmelden kann.)
    - Wenn Sie ein neues Schlüsselpaar erstellen möchten, führen Sie die folgenden Schritte aus:

1. Wählen Sie Ein neues Schlüsselpaar erstellen aus.
2. Geben Sie für Key pair name (Schlüsselpaarname) einen Schlüsselpaarnamen ein.
3. Wählen Sie Schlüsselpaar herunterladen aus und speichern Sie die Datei mit dem privaten Schlüssel an einem sicheren, zugänglichen Ort.

> **Warning**
>
> Nach diesem Zeitpunkt können Sie die Datei mit dem privaten Schlüssel nicht erneut herunterladen. Wenn Sie die Datei mit dem privaten Schlüssel jetzt nicht herunterzuladen, können Sie sich auf die Client-Instance zugreifen.

- Zum Starten Ihrer Instance ohne Schlüsselpaar führen Sie die folgenden Schritte aus:
  1. Wählen Sie Ohne Schlüsselpaar fortfahren aus.
  2. Aktivieren Sie das Kontrollkästchen neben I acknowledge that I will not be able to connect to this instance unless I already know the password built into this AMI. (Ich bestätige, dass ich mich nicht bei dieser Instance anmelden kann, wenn ich das in dieses AMI integrierte Passwort nicht kenne).

Wählen Sie Instances starten aus.

Erstellen Sie den Load Balancer

Führen Sie die folgenden Schritte aus, um einen Elastic Load Balancing Load Balancer zu erstellen, der den HTTPS-Datenverkehr an Ihre Webserver weiterleitet.

Erstellen Sie einen Load Balancer wie folgt:

1. Öffnen Sie den Bereich Load Balancers in der Amazon EC2-Konsole.
2. Klicken Sie auf Load Balancer erstellen.
3. Klicken Sie im Bereich Network Load Balancer auf Create.
4. Führen Sie für Step 1: Configure Load Balancer (Schritt 1; Konfigurieren von Load Balancer) die folgenden Schritte aus:
   a. Geben Sie als Name einen Namen für den Load Balancer ein, den Sie erstellen.
   b. Ändern Sie im Bereich Listeners (Listener) für Load Balancer Port (Load Balancer-Port) den Wert in **443**.

c. Wählen Sie im Bereich Availability Zones als VPC die VPC aus, die Ihre Webserver enthält.

d. Wählen Sie im Bereich Availability Zones die Subnetze aus, die Ihre Webserver enthalten.

e. Wählen Sie Weiter: Routing konfigurieren aus.

5. Führen Sie für Step 2: Configure Routing (Schritt 2: Routing konfigurieren) die folgenden Schritte aus:

   a. Geben Sie als Name einen Namen für die Zielgruppe ein, die Sie erstellen.

   b. Ändern Sie für den Port den Wert in **443**.

   c. Klicken Sie auf Weiter: Ziele registrieren.

6. Führen Sie für Step 3: Register Targets (Schritt 3: Ziele registrieren) die folgenden Schritte aus:

   a. Aktivieren Sie im Bereich Instances die Kontrollkästchen neben Ihren Webserver-Instances. Wählen Sie dann Zu registrierten hinzufügen aus.

   b. Wählen Sie Weiter: Prüfen aus.

7. Überprüfen Sie die Load Balancer-Details und wählen Sie dann Erstellen aus.

8. Wenn der Load Balancer erfolgreich erstellt wurde, klicken Sie auf Close (Schließen).

Nachdem Sie die vorangegangenen Schritte durchgeführt haben, zeigt die Amazon EC2-Konsole Ihren Elastic Load-Balancing Load Balancer an.

Wenn der Status Ihres Load Balancers „aktiv“ ist, können Sie verifizieren, dass der Load Balancer funktioniert. Das heißt, Sie können verifizieren, dass der Load Balancer den HTTPS-Datenverkehr mittels AWS CloudHSM an Ihre Webserver mit SSL/TLS-Auslagerung sendet. Sie können dazu einen Webbrowser oder ein Tool wie OpenSSL s_client verwenden.

So stellen Sie sicher, dass der Load Balancer mit einem Webbrowser arbeitet

1. Suchen Sie in der Amazon EC2-Konsole den DNS-Namen für den Load Balancer, den Sie gerade erstellt haben. Wählen Sie anschließend den DNS-Namen aus und kopieren Sie ihn.

2. Verwenden Sie einen Webbrowser wie Mozilla Firefox oder Google Chrome für die Verbindung Ihres Load Balancers über dessen DNS-Namen. Stellen Sie sicher, dass die URL in die Adresszeile mit https:// beginnt.

> **Tip**
>
> Sie können einen DNS-Service wie Amazon Route 53 nutzen, um den Domainnamen Ihrer Website (beispielsweise https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter Routing des Datenverkehrs zu einer Amazon-EC2-Instance im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

3. Zeigen Sie das Webserverzertifikat mit Ihrem Webbrowser an. Weitere Informationen finden Sie hier:

   - Wenn Sie Mozilla Firefox nutzen, sehen Sie sich die Informationen auf der Mozilla Support-Website unter Zertifikat anzeigen an.
   - Wenn Sie Google Chrome verwenden, sehen Sie sich die Informationen auf der „Google Tools für Web Developers“-Website unter Sicherheitsprobleme verstehen an.

   Andere Webbrowser unterstützen möglicherweise ähnliche Funktionen, über die Sie das Webserverzertifikat anzeigen können.
4. Stellen Sie sicher, dass das Zertifikat dasjenige ist, das Sie für die Nutzung im Webserver konfiguriert haben.

So stellen Sie sicher, dass der Load Balancer mit OpenSSL s_client arbeitet

1. Verbinden Sie mit dem folgenden OpenSSL-Befehl Ihrer Load Balancer mittels HTTPS. Ersetzen Sie *DNS name* durch den DNS-Namen Ihres Load Balancers.

```
openssl s_client -connect <DNS name>:443
```

> **Tip**
>
> Sie können einen DNS-Service wie Amazon Route 53 nutzen, um den Domainnamen Ihrer Website (beispielsweise https://www.example.com/) an Ihren Webserver weiterzuleiten. Weitere Informationen finden Sie unter Routing des Datenverkehrs zu einer Amazon-EC2-Instance im Entwicklerleitfaden zu Amazon Route 53 oder in der Dokumentation für Ihren DNS-Service.

2. Stellen Sie sicher, dass das Zertifikat dasjenige ist, das Sie für die Nutzung im Webserver konfiguriert haben.

Sie verfügen jetzt über eine Website, die mit HTTPS durch den privaten Schlüssel des Webservers, der in einem HSM in Ihrem AWS CloudHSMCluster gespeichert ist, gesichert ist. Ihre Website verfügt über zwei Webserver und einen Load Balancer, um Effizienz und Verfügbarkeit zu verbessern.

Konfigurieren von Windows Server als Zertifizierungsstelle (CA, Certificate Authority) mit AWS CloudHSM

In einer Public Key-Infrastruktur (PKI) ist eine Zertifizierungsstelle (CA) eine vertrauenswürdige Entität, die digitale Zertifikate ausstellt. Diese digitalen Zertifikate binden einen öffentlichen Schlüssel unter Verwendung von Kryptografie öffentlicher Schlüssel und von digitalen Signaturen an eine Identität (eine Person oder Organisation). Um eine CA zu betreiben, müssen Sie das Vertrauen aufrechterhalten, indem Sie den privaten Schlüssel schützen, der die von Ihrer CA ausgestellten Zertifikate signiert. Sie können den privaten Schlüssel im HSM in Ihrem AWS CloudHSM-Cluster speichern und das HSM zur Durchführung der kryptografischen Signiervorgänge verwenden.

In diesem Tutorial verwenden Sie Windows Server und AWS CloudHSM zum Konfigurieren einer Zertifizierungsstelle. Nach der Installation der AWS CloudHSM-Client-Software für Windows auf Ihrem Windows Server fügen Sie Ihrem Windows Server die Active Directory-Zertifikatsdienste (AD CS)-Rolle hinzu. Wenn Sie diese Rolle konfigurieren, verwenden Sie einen AWS CloudHSM-Schlüsselspeicher-Provider (KSP, Key Storage Provider) zum Erstellen und Speichern des privaten CA-Schlüssels in Ihrem AWS CloudHSM-Cluster. Der KSP bildet die Brücke, die Ihren Windows-Server mit Ihrem AWS CloudHSM-Cluster verbindet. Im letzten Schritt signieren Sie mit Ihrer Windows Server-CA eine Zertifikatssignierungsanforderung (Certificate Signing Request, CSR).

Weitere Informationen finden Sie unter den folgenden Themen:

Themen

- Schritt 1 für die Windows Server-CA: Einrichten der Voraussetzungen
- Schritt 2 für die Windows Server-CA: Erstellen einer Windows Server-CA mit AWS CloudHSM
- Schritt 3 für die Windows Server-CA: Signieren einer Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) mit Ihrer Windows Server-Zertifizierungsstelle (CA) mit AWS CloudHSM

Schritt 1 für die Windows Server-CA: Einrichten der Voraussetzungen

Um Windows Server als Zertifizierungsstelle (CA) mit AWS CloudHSM einzurichten, benötigen Sie Folgendes:

- Einen aktiven AWS CloudHSM-Cluster mit mindestens einem HSM.
- Eine Amazon EC2-Instance, die ein Windows Server-Betriebssystem ausführt und auf der die AWS CloudHSM-Client-Software für Windows installiert ist. In diesem Tutorial wird Microsoft Windows Server 2016 verwendet.
- Einen Crypto-Benutzer (CU), der den privaten Schlüssel der CA auf dem HSM besitzen und verwalten soll.

So richten Sie die Voraussetzungen für eine Windows Server-CA mit AWS CloudHSM ein

1. Führen Sie die Schritte unter Erste Schritte aus. Beim Starten der Amazon EC2-Clients wählen Sie ein Windows Server-AMI aus. In diesem Tutorial wird Microsoft Windows Server 2016 verwendet. Wenn Sie diese Schritte ausgeführt haben, verfügen Sie über einen aktiven Cluster mit mindestens einem HSM. Sie verfügen auch über eine Amazon EC2-Client-Instance, auf der Windows Server ausgeführt wird und auf der die AWS CloudHSM-Client-Software für Windows installiert ist.
2. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter Hinzufügen eines HSM.
3. Stellen Sie eine Verbindung mit Ihrer Client-Instance her. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
4. Erstellen Sie einen Crypto-Benutzer (CU) mit Verwalten von HSM-Benutzern mit CloudHSM CLI oder Verwalten von HSM-Benutzern mit CloudHSM Management Utility (CMU). Merken Sie sich den CU-Benutzernamen und das Passwort. Sie benötigen diese im nächsten Schritt.
5. Legen Sie die Anmeldeinformationen für das HSM fest. Verwenden Sie hierfür den im vorherigen Schritt erstellten CU-Benutzernamen und das entsprechende Passwort.
6. Wenn Sie in Schritt 5 die Windows-Anmeldeinformationsverwaltung verwendet haben, um HSM-Anmeldeinformationen festzulegen, laden Sie `psexec.exe` von SysInternals herunter, um den folgenden Befehl als NT Authority\SYSTEM auszuführen:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
 --username <USERNAME> --password <PASSWORD>
```

Ersetzen Sie *<username>* und *<password>* durch die HSM-Anmeldeinformationen.

Um eine Windows Server-CA mit AWS CloudHSM zu erstellen, navigieren Sie zu Erstellen einer Windows Server-CA.

Schritt 2 für die Windows Server-CA: Erstellen einer Windows Server-CA mit AWS CloudHSM

Um eine Windows Server-CA zu erstellen, fügen Sie Ihrem Windows Server die Active Directory-Zertifikatsdienste (Active Directory Certificate Services, AD CS)-Rolle hinzu. Wenn Sie diese Rolle hinzufügen, verwenden Sie einen AWS CloudHSM-Schlüsselspeicher-Provider (KSP, Key Storage Provider) zum Erstellen und Speichern des privaten CA-Schlüssels in Ihrem AWS CloudHSM-Cluster.

> **Note**
>
> Wenn Sie Ihre Windows Server-CA erstellen, können Sie sich für das Erstellen einer Stamm-Zertifizierungsstelle oder einer untergeordneten Zertifizierungsstelle entscheiden. Sie treffen diese Entscheidung in der Regel basierend auf dem Entwurf Ihrer Public Key-Infrastruktur und der Sicherheitsrichtlinien Ihrer Organisation. In diesem Tutorial wird der Einfachheit halber erklärt, wie eine Stamm-CA erstellt wird.

So fügen Sie Ihrem Windows Server die AD-CS-Rolle hinzu und erstellen den privaten Schlüssel der CA

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
2. Starten Sie auf Ihrem Windows-Server Server Manager.
3. Wählen Sie im Dashboard Server-Manager die Option Rollen und Features hinzufügen.
4. Lesen Sie die Informationen unter Before you begin (Zur Vorbereitung) und klicken Sie dann auf Weiter.
5. Wählen Sie unter Installationsart die Rollenbasierte oder funktionsbasierte Installation. Wählen Sie anschließend Weiter.

6. Wählen Sie unter Serverauswahl die Option Einen Server aus dem Serverpool auswählen. Wählen Sie anschließend Weiter.
7. Gehen Sie für Serverrollen wie folgt vor:
   a. Wählen Sie Active Directory-Zertifikatdienste.
   b. Wählen Sie für Sollen für Active Directory-Zertifikatsdienste erforderliche Features hinzugefügt werden? die Option Features hinzufügen.
   c. Klicken Sie auf Weiter, um das Auswählen von Serverrollen abzuschließen.
8. Übernehmen Sie für Features die Standardwerte und klicken Sie dann auf Weiter.
9. Verfahren Sie für AD CS wie folgt:
   a. Wählen Sie Weiter aus.
   b. Wählen Sie Zertifizierungsstelle und klicken Sie dann auf Weiter.
10. Lesen Sie unter Bestätigung die Informationen zur Bestätigung und klicken Sie dann auf Installieren. Schließen Sie das Fenster nicht.
11. Wählen Sie den hervorgehobenen Link Active Directory-Zertifikatdienste auf dem Zielserver konfigurieren.
12. Prüfen oder ändern Sie die Anmeldeinformationen unter Anmeldeinformationen. Wählen Sie anschließend Weiter.
13. Wählen Sie für Rollendienste die Option Zertifizierungsstelle. Wählen Sie anschließend Weiter.
14. Wählen Sie für Installationsart die Option Eigenständige Zertifizierungsstelle. Wählen Sie anschließend Weiter.
15. Wählen Sie für Zertifizierungsstellentyp die Option Stammzertifizierungsstelle. Wählen Sie anschließend Weiter.

    > **Note**
    >
    > Sie können basierend auf dem Entwurf Ihrer Public Key-Infrastruktur und der Sicherheitsrichtlinien Ihrer Organisation entscheiden, ob eine Stammzertifizierungsstelle oder eine untergeordnete Zertifizierungsstelle erstellt werden soll. In diesem Tutorial wird der Einfachheit halber erklärt, wie eine Stamm-CA erstellt wird.

16. Wählen Sie für Privater Schlüssel die Option Neuen privaten Schlüssel erstellen. Wählen Sie anschließend Weiter.
17. Führen Sie für Kryptografie einen der folgenden Schritte aus:

a. Wählen Sie für Wählen Sie einen Kryptografie-Provider aus eine der Optionen für Cavium-Schlüsselspeicher-Provider im Menü aus. Dies sind die AWS CloudHSM-Schlüsselspeicher-Provider. Beispielsweise können Sie RSA#Cavium Key Storage Provider (RSA#Cavium-Schlüsselspeicher-Provider) wählen.

b. Wählen Sie für Schlüssellänge eine der verfügbaren Schlüssellängen aus.

c. Wählen Sie für Select the hash algorithm für signing certificates issued by this CA (Wählen Sie den Hashalgorithmus zum Signieren von Zertifikaten aus, die von dieser Zertifizierungsstelle ausgestellt wurden) einen der verfügbaren Hashalgorithmen aus.

Wählen Sie Weiter.

18. Führen Sie für Zertifizierungsstelle einen der folgenden Schritte aus:

    a. (Optional) Bearbeiten Sie den allgemeinen Namen.

    b. (Optional) Geben Sie ein spezifisches Namenssuffix ein.

    Wählen Sie Weiter.

19. Geben Sie für Gültigkeitsdauer einen Zeitraum in Jahren, Monaten, Wochen oder Tagen ein. Wählen Sie anschließend Weiter.

20. Für Zertifikatdatenbankkönnen Sie die Standardwerte übernehmen. Sie können optional den Speicherort für die Datenbank und das Datenbankprotokoll ändern. Wählen Sie anschließend Weiter.

21. Überprüfen Sie die Informationen zur Zertifizierungsstelle unter Bestätigung. Klicken Sie dann auf Konfigurieren.

22. Klicken Sie auf Schließen und dann erneut auf Schließen.

Sie verfügen jetzt über eine Windows Server-CA mit AWS CloudHSM. Weitere Informationen zum Signieren einer Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) mit Ihrer CA finden Sie unter Signieren einer CSR.

Schritt 3 für die Windows Server-CA: Signieren einer Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) mit Ihrer Windows Server-Zertifizierungsstelle (CA) mit AWS CloudHSM

Sie können Ihre Windows Server-CA mit AWS CloudHSM zum Signieren einer Zertifikatssignierungsanforderung (Certificate Signing Request, CSR) verwenden. Um diese Schritte ausführen zu können, benötigen Sie eine gültige CSR. Es gibt mehrere Möglichkeiten, eine CSR zu erstellen, einschließlich der Folgenden:

- Verwenden von OpenSSL
- Verwenden des Windows Server Internet Information Services (IIS) Manager
- Verwenden des Zertifikate-Snap-In in der Microsoft Management Console
- Verwenden des Befehlszeilendienstprogramms certreq unter Windows

Die Schritte zum Erstellen einer CSR werden in dieser Anleitung allerdings nicht behandelt. Wenn Sie über eine CSR verfügen, können Sie sie mit Ihrer Windows Server-Zertifizierungsstelle signieren.

So signieren Sie eine CSR mit Ihrer Windows Server-CA

1. Stellen Sie eine Verbindung mit Ihrem Windows-Server her, sofern noch nicht geschehen. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
2. Starten Sie auf Ihrem Windows-Server Server Manager.
3. Klicken Sie rechts oben im Dashboard Server-Manager auf Extras, Zertifizierungsstelle.
4. Wählen Sie im Fenster Zertifizierungsstelle den Namen Ihres Computers aus.
5. Wählen Sie im Menü Aktion die Option Alle Aufgaben, Neue Anforderung einreichen.
6. Wählen Sie Ihre CSR-Datei aus und klicken Sie auf Öffnen.
7. Doppelklicken Sie im Fenster Zertifizierungsstelle auf Ausstehende Anforderungen.
8. Wählen Sie die ausstehende Anforderung aus. Wählen Sie dann im Menü Aktion die Option Alle Aufgaben, Ausstellen.
9. Doppelklicken Sie im Fenster Zertifizierungsstelle auf Ausgestellte Anforderungen, um das signierte Zertifikat anzuzeigen.
10. (Optional) Um das signierte Zertifikat in eine Datei zu exportieren, führen Sie die folgenden Schritte aus:

a. Doppelklicken Sie im Fenster Zertifizierungsstelle auf das Zertifikat.

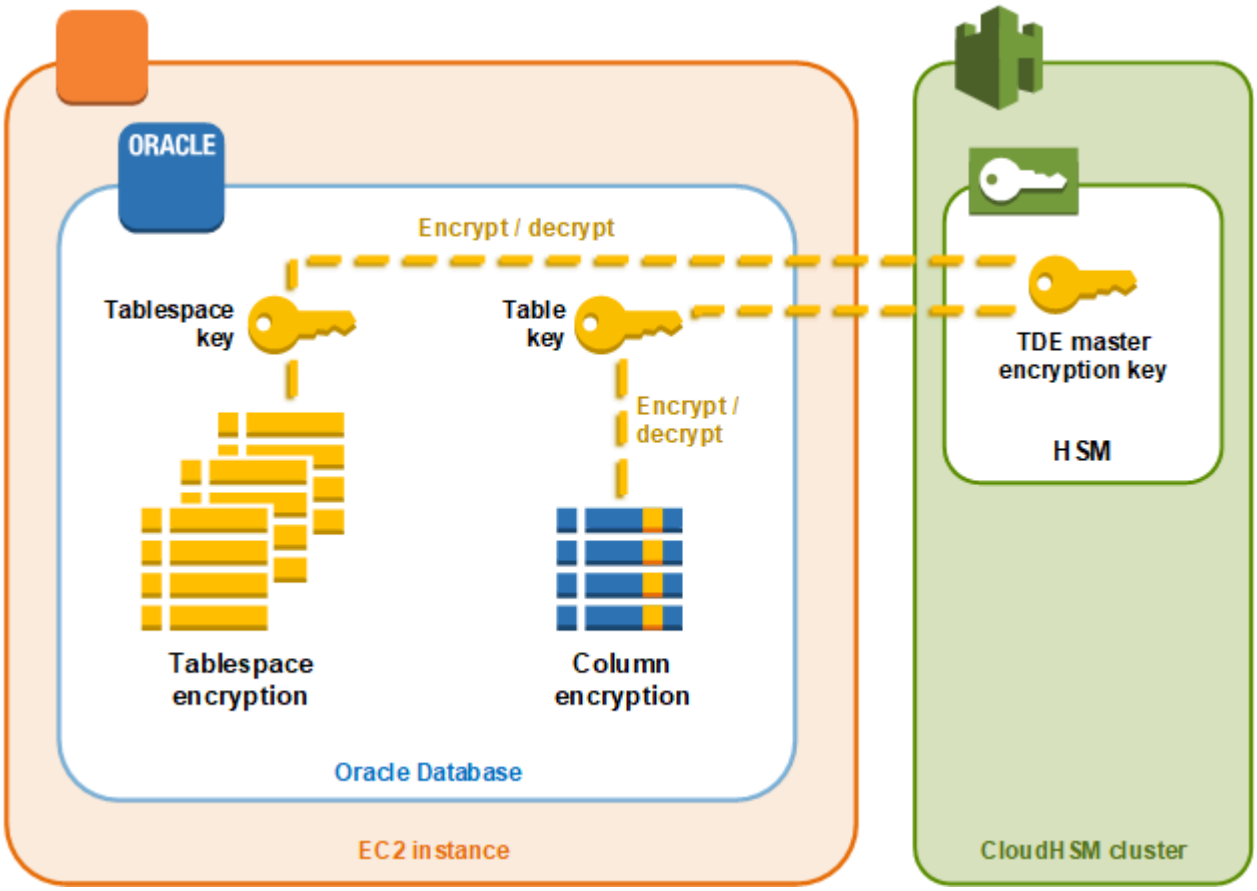
b. Wählen Sie die Registerkarte Details und anschließend In Datei kopieren.

c. Befolgen Sie die Anweisungen im Zertifikatexport-Assistent.

Sie verfügen jetzt über eine Windows Server-CA mit AWS CloudHSM und ein gültiges Zertifikat, das von der Windows Server-CA signiert ist.

Oracle Database Transparent Data Encryption (TDE) mit AWS CloudHSM

Transparent Data Encryption (TDE) wird für die Verschlüsselung von Datenbankdateien eingesetzt. Mit TDE verschlüsselt die Datenbanksoftware die Daten, bevor sie auf der Festplatte gespeichert werden. Die Daten in den Tabellenspalten oder Tablespaces der Datenbank werden mit einem Tabellenschlüssel oder einem Tablespace-Schlüssel verschlüsselt. Einige Versionen der Datenbanksoftware von Oracle bieten TDE. In Oracle TDE werden diese Schlüssel mit einem TDE-Master-Schlüssel verschlüsselt. Sie können mehr Sicherheit erreichen, indem Sie den TDE-Master-Verschlüsselungsschlüssel in den HSMs in Ihrem AWS CloudHSM-Cluster speichern.



In dieser Lösung verwenden Sie Oracle Database, das in einer Amazon EC2-Instance installiert ist. Oracle Database ist mit der AWS CloudHSM-Softwarebibliothek für PKCS #11 integriert, um den TDE-Masterschlüssel in den HSMs in Ihrem Cluster zu speichern.

> **Important**
>
> - Wir empfehlen die Installation der Oracle-Datenbank auf einer Amazon-EC2-Instance.

Führen Sie die folgenden Schritte aus, um die Oracle TDE-Integration mit AWS CloudHSM zu vollenden.

So konfigurieren Sie die Oracle TDE-Integration mit AWS CloudHSM

1. Führen Sie die Schritte unter Einrichten der Voraussetzungen aus, um die Umgebung vorzubereiten.
2. Führen Sie die Schritte in Konfigurieren der Datenbank aus, um Oracle Database für die Integration mit Ihrem AWS CloudHSM-Cluster zu konfigurieren.

Oracle TDE mit AWS CloudHSM: Einrichten der Voraussetzungen

Für die Integration von Oracle TDE in AWS CloudHSM benötigen Sie Folgendes:

- Einen aktiven AWS CloudHSM-Cluster mit mindestens einem HSM.
- Eine Amazon EC2-Instance, auf der das Amazon Linux-Betriebssystem mit der folgenden Software installiert ist:
  - Die AWS CloudHSM-Client und Befehlszeilen-Tools.
  - Die AWS CloudHSM-Softwarebibliothek für PKCS #11.
  - Oracle-Datenbank. AWS CloudHSM unterstützt die Oracle-TDE-Integration. Client-SDK 5.6 und höher unterstützen Oracle-Datenbank 19c. Das Client-SDK 3 unterstützt Oracle TDE für die Oracle-Datenbankversionen 11g und 12c.
- Ein Crypto-Benutzer (CU), der den TDE-Master-Verschlüsselungsschlüssel in den HSMs Ihres Clusters besitzt und verwaltet.

Führen Sie zum Einrichten aller Voraussetzungen die folgenden Schritte aus.

So richten Sie die Voraussetzungen für die Oracle TDE-Integration in AWS CloudHSM ein

1. Führen Sie die Schritte unter Erste Schritte aus. Anschließend verfügen Sie über einen aktiven Cluster mit einem HSM. Außerdem verfügen Sie über eine Amazon-EC2-Instance, auf der das Amazon-Linux-Betriebssystem läuft. Der AWS CloudHSM-Client und die Befehlszeilen-Tools werden ebenfalls installiert und konfiguriert.
2. (Optional) Fügen Sie Ihrem Cluster weitere HSMs hinzu. Weitere Informationen finden Sie unter Hinzufügen eines HSM.
3. Stellen Sie eine Verbindung zur Ihrer Amazon EC2-Client-Instance her und führen Sie die folgenden Schritte aus:

   a. Installieren Sie die AWS CloudHSM-Softwarebibliothek für PKCS #11.

   b. Installieren Sie Oracle Database. Weitere Informationen finden Sie in der Oracle Database-Dokumentation. Client-SDK 5.6 und höher unterstützen Oracle-Datenbank 19c. Das Client-SDK 3 unterstützt Oracle TDE für die Oracle-Datenbankversionen 11g und 12c.

   c. Verwenden Sie zum Erstellen eines Crypto-Benutzers (CU) in Ihrem Cluster das Befehlszeilen-Tool cloudhsm_mgmt_util. Weitere Informationen zum Erstellen eines CU finden Sie unter So verwalten Sie HSM-Benutzer mit CMU und Verwalten von HSM-Benutzern.

Nachdem Sie diese Schritte abgeschlossen haben, fahren Sie mit Konfigurieren der Datenbank fort.

Oracle TDE mit AWS CloudHSM: Konfigurieren der Datenbank und Erstellen des Master-Verschlüsselungsschlüssels

Lesen Sie in den folgenden Themen nach, um Oracle TDE in Ihren AWS CloudHSM-Cluster zu integrieren:

1. Aktualisieren der Oracle Database-Konfiguration für die Nutzung der HSMs in Ihrem Cluster als externes Sicherheitsmodul. Weitere Informationen zu externen Sicherheitsmodulen finden Sie unter Introduction to Transparent Data Encryption im Oracle Database Advanced Security Guide.
2. Generieren des Oracle TDE-Master-Verschlüsselungsschlüssels auf den HSMs in Ihrem Cluster.

Aktualisieren der Oracle Database-Konfiguration

Um die Oracle Database-Konfiguration zu aktualisieren, um ein HSM in Ihrem Cluster als externes Sicherheitsmodul zu verwenden, führen Sie die folgenden Schritte aus. Weitere Informationen zu externen Sicherheitsmodulen finden Sie unter Introduction to Transparent Data Encryption im Oracle Database Advanced Security Guide.

So aktualisieren Sie die Oracle-Konfiguration

1. Stellen Sie eine Verbindung zu Ihrer Amazon-EC2-Client-Instance her. Dies ist die Instance, in der Sie Oracle Database installiert haben.
2. Erstellen Sie eine Sicherungskopie der Datei `sqlnet.ora`. Informationen zum Speicherort dieser Datei finden Sie in der Oracle-Dokumentation.
3. Verwenden Sie einen Texteditor, um die Daten namens `sqlnet.ora` zu bearbeiten. Fügen Sie die folgende Zeile zu. Wenn eine Zeile in der Datei mit `encryption_wallet_location` beginnt, ersetzen Sie diese durch die folgende.

   ```
   encryption_wallet_location=(source=(method=hsm))
   ```

   Speichern Sie die Datei.
4. Führen Sie den folgenden Befehl aus, um das Verzeichnis einzurichten, in dem Oracle Database die Bibliotheksdatei für die AWS CloudHSM-Softwarebibliothek für PKCS #11 erwartet.

   ```
   sudo mkdir -p /opt/oracle/extapi/64/hsm
   ```

5. Führen Sie den folgenden Befehl zum Kopieren der AWS CloudHSM-Softwarebibliothek der PKCS #11-Datei in das Verzeichnis, das Sie im vorherigen Schritt erstellt haben aus.

```
sudo cp /opt/cloudhsm/lib/libcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
```

> **Note**
>
> Das Verzeichnis `/opt/oracle/extapi/64/hsm` darf nur eine Bibliotheksdatei enthalten. Entfernen Sie alle anderen Dateien, die in diesem Verzeichnis vorhanden sind.

6. Führen Sie den folgenden Befehl aus, um den Eigentümer des Verzeichnisses `/opt/oracle` und alle darin enthaltenen Daten zu ändern.

```
sudo chown -R oracle:dba /opt/oracle
```

7. Starten Sie Oracle Database.

Generieren des Oracle TDE-Master-Verschlüsselungsschlüssels

Führen Sie zum Generieren des Oracle TDE-Masterschlüssels auf den HSMs Ihres Clusters die folgenden Schritte aus.

So generieren Sie den Masterschlüssel

1. Verwenden Sie den folgenden Befehl, um Oracle SQL*Plus zu öffnen. Geben Sie, wenn Sie dazu aufgefordert werden, das Systempasswort ein, das Sie beim Installieren von Oracle Database festgelegt haben.

```
sqlplus / as sysdba
```

> **Note**
>
> Für Client-SDK 3 müssen Sie die `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE`-Umgebungsvariable jedes Mal setzen, wenn Sie einen Hauptschlüssel erzeugen. Diese Variable wird nur für die Generierung des Masterschlüssels benötigt. Weitere Informationen finden Sie unter „Problem: Oracle legt das PCS #11-Attribut `CKA_MODIFIABLE` bei der Generierung des Masterschlüssels fest, aber

das HSM unterstützt es nicht“ unter Bekannte Probleme für die Integration von Drittanbieteranwendungen.

2. Führen Sie die SQL-Anweisung aus, die den Master-Verschlüsselungsschlüssel erstellt, wie in den folgenden Beispielen gezeigt. Verwenden Sie die Anweisung, die Ihrer Version von Oracle Database entspricht. Ersetzen Sie *<CU user name>* durch den Benutzernamen des Crypto-Benutzers (CU). Ersetzen Sie *<password>* durch das CU-Passwort.

   Important

   Führen Sie einmalig den folgenden Befehl aus. Jedes Mal, wenn der Befehl ausgeführt wird, wird ein neuer Master-Verschlüsselungsschlüssel erstellt.

   - Führen Sie für Oracle Database, Version 11, die folgende SQL-Anweisung aus.

     ```
     SQL> alter system set encryption key identified by "<CU user name>:<password>";
     ```

   - Für Oracle Database Version 12 und Version 19c führen Sie die folgende SQL-Anweisung aus.

     ```
     SQL> administer key management set key identified by "<CU user
      name>:<password>";
     ```

   Wenn die Antwort `System altered` oder `keystore altered` ist, haben Sie den Masterschlüssel für Oracle TDE erfolgreich generiert und festgelegt.

3. (Optional) Führen Sie den folgenden Befehl aus, um den Status von Oracle Wallet zu verifizieren.

   ```
   SQL> select * from v$encryption_wallet;
   ```

   Ist das Wallet nicht geöffnet, öffnen Sie es mit einem der folgenden Befehle. Ersetzen Sie *<CU user name>* durch den Namen des Crypto-Benutzers (CU). Ersetzen Sie *<password>* durch das CU-Passwort.

   - Führen Sie für Oracle 11 zum Öffnen des Wallet den folgenden Befehl aus.

```
SQL> alter system set encryption wallet open identified by "<CU user
 name>:<password>";
```

Führen Sie zum manuellen Schließen des Wallet den folgenden Befehl aus.

```
SQL> alter system set encryption wallet close identified by "<CU user
 name>:<password>";
```

- Bei Oracle 12 und Oracle 19c führen Sie den folgenden Befehl aus, um die Brieftasche zu öffnen.

```
SQL> administer key management set keystore open identified by "<CU user
 name>:<password>";
```

Führen Sie zum manuellen Schließen des Wallet den folgenden Befehl aus.

```
SQL> administer key management set keystore close identified by "<CU user
 name>:<password>";
```

Verwenden von Microsoft SignTool mit AWS CloudHSM zum Signieren von Dateien

In der Kryptographie und bei PKIs (Public Key Infrastructure) werden digitale Signaturen verwendet, um zu bestätigen, dass Daten von einer vertrauenswürdigen Einheit gesendet wurden. Signaturen zeigen außerdem, dass die Daten während der Übertragung nicht manipuliert wurden. Eine Signatur ist ein verschlüsselter Hash, der mit dem privaten Schlüssel des Absenders erzeugt wird. Der Empfänger kann die Integrität der Daten überprüfen, indem er ihre Hash-Signatur mit dem öffentlichen Schlüssel des Absenders entschlüsselt. Im Gegenzug liegt es in der Verantwortung des Absenders, ein digitales Zertifikat zu pflegen. Das digitale Zertifikat belegt den Besitz des privaten Schlüssels durch den Absender und stellt dem Empfänger den öffentlichen Schlüssel zur Verfügung, der für die Entschlüsselung benötigt wird. Solange sich der private Schlüssel im Besitz des Absenders befindet, kann der Signatur vertraut werden. AWS CloudHSM bietet eine sichere, nach FIPS 140-2 Level 3 validierte Hardware, mit der Sie diese Schlüssel mit exklusivem Einzelmandantenzugriff sichern können.

Viele Unternehmen verwenden das Microsoft SignTool. Es handelt sich um ein Befehlszeilentool, das Dateien signieren, überprüfen und mit Zeitstempeln versehen kann, um den Code-Signaturprozess zu vereinfachen. Sie können AWS CloudHSM verwenden, um Ihre Schlüsselpaare sicher zu speichern, bis sie vom SignTool benötigt werden. So können Sie einen einfach zu automatisierenden Workflow für das Signieren von Daten einrichten.

Die folgenden Themen geben einen Überblick über die Verwendung von SignTool mit AWS CloudHSM:

Themen

Microsoft SignTool mit AWS CloudHSM, Schritt 1: Einrichten der Voraussetzungen

Um das Microsoft SignTool mit AWS CloudHSM zu verwenden, benötigen Sie Folgendes:

- Eine Amazon-EC2-Client-Instance, auf der ein Windows-Betriebssystem läuft.
- Eine Zertifizierungsstelle (CA), die entweder selbst unterhalten oder von einem Drittanbieter eingerichtet wurde.
- Ein aktiver AWS CloudHSM-Cluster in derselben Virtual Public Cloud (VPC) wie Ihre EC2-Instance. Der Cluster muss mindestens ein HSM enthalten.
- Ein Crypto-Benutzer (CU), der Schlüssel im AWS CloudHSM-Cluster besitzt und verwaltet.
- Eine unsignierte Datei oder ausführbare Datei.
- Das Microsoft Windows Software Development Kit (SDK).

So richten Sie die Voraussetzungen für die Verwendung von AWS CloudHSM mit dem Windows SignTool ein:

1. Befolgen Sie die Anweisungen im Abschnitt Erste Schritte dieses Handbuchs, um eine Windows EC2-Instance und einen AWS CloudHSM-Cluster zu starten.

2. Wenn Sie Ihre eigene Windows Server CA hosten möchten, führen Sie die Schritte 1 und 2 unter Windows Server mit AWS CloudHSM als Zertifizierungsstelle konfigurieren aus. Andernfalls verwenden Sie weiterhin Ihre öffentliche, vertrauenswürdige Drittanbieter-CA.
3. Laden Sie eine der folgenden Versionen des Microsoft Windows-SDKs herunter und installieren Sie es in Ihrer Windows EC2-Instance:

   - Microsoft Windows SDK 10
   - Microsoft Windows SDK 8.1
   - Microsoft Windows SDK 7

   Die ausführbare Datei `SignTool` ist Teil der Installationsfeatures von Windows SDK Signing Tools für Desktop Apps. Sie können die anderen zu installierenden Features weglassen, wenn Sie sie nicht benötigen. Der Standardinstallationspfad ist:

   ```
   C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU
    architecture>\signtool.exe
   ```

Sie können nun das Microsoft Windows-SDK, Ihren AWS CloudHSM-Cluster und Ihre CA zum Erstellen eines Signierungszertifikats verwenden.

Microsoft SignTool mit AWS CloudHSM, Schritt 2: Erstellen eines Signaturzertifikats

Nachdem Sie das Windows-SDK auf Ihre EC2-Instance heruntergeladen haben, können Sie damit eine Certificate Signing Request (CSR) generieren. Die CSR ist ein unsigniertes Zertifikat, das möglicherweise zur Signatur an Ihre CA weitergeleitet wird. In diesem Beispiel verwenden wir die ausführbare Datei `certreq`, die im Windows-SDK enthalten ist, um die CSR zu generieren.

So erstellen Sie eine CSR mit der ausführbaren Datei **`certreq`**:

1. Wenn Sie dies noch nicht getan haben, verbinden Sie sich mit Ihrer Windows EC2-Instance. Weitere Informationen finden Sie unter Herstellen einer Verbindung zu Ihrer Instance im Amazon-EC2-Benutzerhandbuch für Windows-Instances.
2. Erstellen Sie eine Datei namens `request.inf`, die die folgenden Zeilen enthält. Ersetzen Sie die `Subject`-Informationen durch die Ihrer Organisation. Eine Erklärung der einzelnen Parameter finden Sie in der Dokumentation von Microsoft.

```
[Version]
Signature= $Windows NT$
[NewRequest]
Subject = "C=<Country>,CN=<www.website.com>,O=<Organization>,OU=<Organizational-
Unit>,L=<City>,S=<State>"
RequestType=PKCS10
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = Cavium Key Storage Provider
KeyUsage = "CERT_DIGITAL_SIGNATURE_KEY_USAGE"
MachineKeySet = True
Exportable = False
```

3. Führen Sie `certreq.exe`. In diesem Beispiel speichern wir die CSR als `request.csr`.

```
certreq.exe -new request.inf request.csr
```

Intern wird ein neues Schlüsselpaar auf Ihrem AWS CloudHSM-Cluster generiert, und der private Schlüssel des Paares wird zur Erstellung der CSR verwendet.

4. Übermitteln Sie die CSR an Ihre CA. Wenn Sie eine Windows Server-CA verwenden, führen Sie diese Schritte aus:

   a. Geben Sie den folgenden Befehl ein, um das CA-Tool zu öffnen:

   ```
   certsrv.msc
   ```

   b. Klicken Sie im neuen Fenster mit der rechten Maustaste auf den Namen des CA-Servers. Wählen Sie Alle Aufgaben aus, und wählen Sie dann Neue Anfrage senden aus.

   c. Navigieren Sie zum Speicherort von `request.csr` und wählen Sie Öffnen aus.

   d. Navigieren Sie zum Ordner Pending Requests (Ausstehende Anforderungen), indem Sie das Menü Server CA erweitern. Klicken Sie mit der rechten Maustaste auf die gerade erstellte Anforderung, und wählen Sie unter All tasks (Alle Aufgaben) Issue (Ausstellen) aus.

   e. Navigieren Sie nun zum Ordner Issued Certificates (Ausgestellte Zertifikate) (über dem Ordner Pending Requests (Ausstehende Anforderungen)).

   f. Wählen Sie Öffnen aus, um das Zertifikat anzuzeigen, und wählen Sie dann die Registerkarte Details aus.

g. Wählen Sie In Datei kopieren aus, um den Assistenten für den Zertifikatsexport zu starten. Speichern Sie die DER-kodierte X.509-Datei an einem sicheren Ort als `signedCertificate.cer`.

h. Verlassen Sie das CA-Tool und verwenden Sie den folgenden Befehl, der die Zertifikatsdatei in den persönlichen Zertifikatsspeicher unter Windows verschiebt. Sie kann dann von anderen Anwendungen verwendet werden.

```
certreq.exe -accept signedCertificate.cer
```

Sie können zum Signieren einer Datei nun Ihr importiertes Zertifikat in verwenden.

Microsoft SignTool mit AWS CloudHSM Schritt 3: Signieren einer Datei

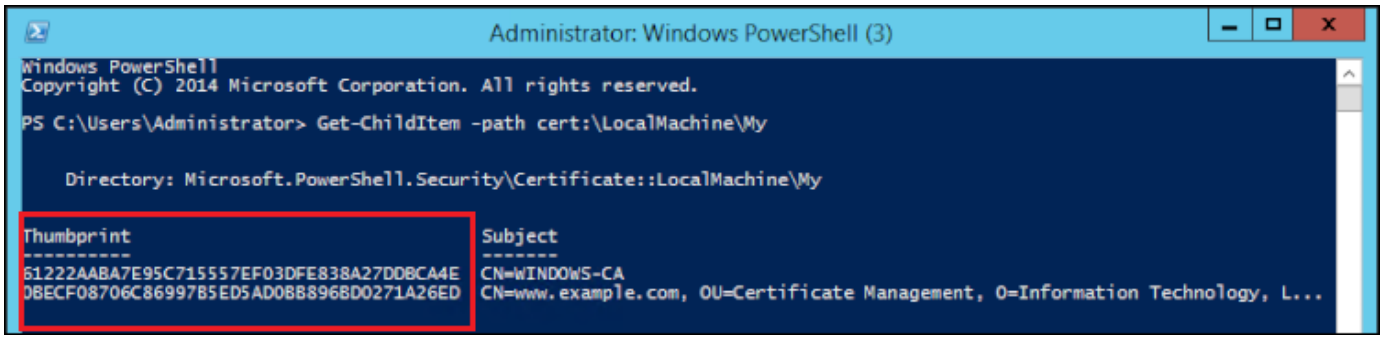
Sie können jetzt SignTool und Ihr importiertes Zertifikat verwenden, um Ihre Beispieldatei zu signieren. Um dies zu tun, müssen Sie den SHA-1-Hash oder Fingerabdruck (Thumbprint) des Zertifikats kennen. Der Thumbprint wird verwendet, um sicherzustellen, dass SignTool nur Zertifikate verwendet, die von verifiziert wurdenAWS CloudHSM. In diesem Beispiel verwenden wir , PowerShell um den Hash des Zertifikats abzurufen. Sie können auch die GUI der CA oder die ausführbare Datei `certutil` des Windows-SDKs verwenden.

So erhalten Sie den Fingerabdruck eines Zertifikats und signieren damit eine Datei:

1. Öffnen Sie PowerShell als Administrator und führen Sie den folgenden Befehl aus:

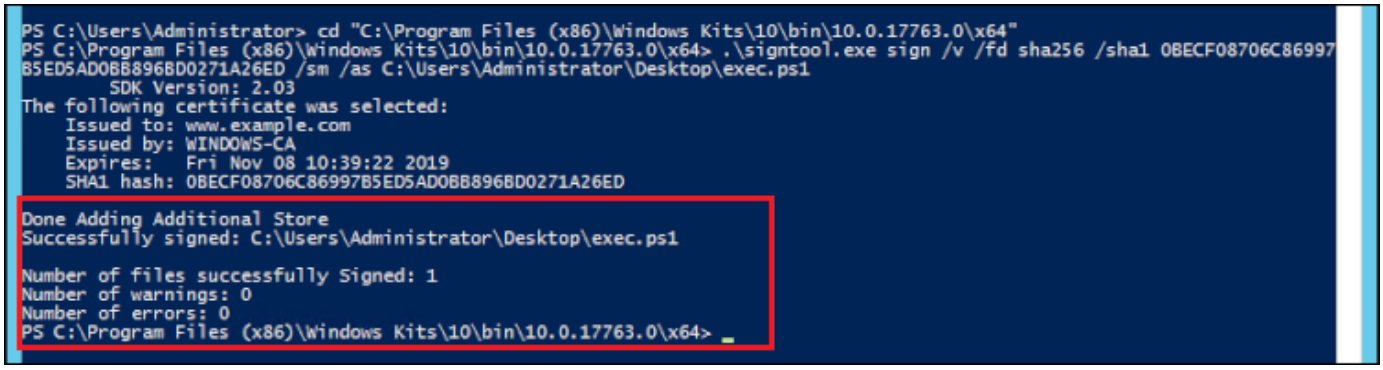
```
Get-ChildItem -path cert:\LocalMachine\My
```

Kopieren Sie den `Thumbprint`, der zurückgegeben wird.



2. Navigieren Sie zu dem Verzeichnis in PowerShell , das enthält`SignTool.exe`. Der Standardspeicherort ist `C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64`.
3. Signieren Sie dann Ihre Datei, indem Sie den folgenden Befehl ausführen. Wenn der Befehl erfolgreich ist, PowerShell gibt eine Erfolgsmeldung zurück.

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator
\Desktop\<test>.ps1
```



4. (Optional) Um die Signatur der Datei zu überprüfen, verwenden Sie den folgenden Befehl:

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Java Keytool und Jarsigner

AWS CloudHSM bietet die Integration mit den Hilfsprogrammen Java Keytool und Jarsigner über Client-SDK 3 und Client-SDK 5. Die Schritte zur Verwendung dieser Tools hängen von der Version des Client-SDK ab, die Sie derzeit heruntergeladen haben:

- Verwendung von Client-SDK 5 zur Integration mit Java Keytool und Jarsigner
- Verwendung von Client-SDK 3 zur Integration mit Java Keytool und Jarsigner

Verwendung von Client-SDK 5 zur Integration mit Java Keytool und Jarsigner

AWS CloudHSM Key Store ist ein spezieller JCE-Schlüsselspeicher, der Zertifikate verwendet, die mit Schlüsseln auf Ihrem HSM über Tools von Drittanbietern wie `keytool` und `jarsigner` verknüpft sind. AWS CloudHSM speichert keine Zertifikate auf dem HSM, da Zertifikate öffentliche, nicht

vertrauliche Daten sind. Der AWS CloudHSM-Schlüsselspeicher speichert die Zertifikate in einer lokalen Datei und ordnet sie den entsprechenden Schlüsseln auf Ihrem HSM zu.

Wenn Sie den AWS CloudHSM-Schlüsselspeicher verwenden, um neue Schlüssel zu generieren, werden keine Einträge in der lokalen Schlüsselspeicherdatei generiert - die Schlüssel werden auf dem HSM erstellt. Wenn Sie den AWS CloudHSM-Schlüsselspeicher verwenden, um nach Schlüsseln zu suchen, wird die Suche an das HSM übergeben. Wenn Sie Zertifikate im AWS CloudHSM-Schlüsselspeicher speichern, überprüft der Anbieter, ob ein Schlüsselpaar mit dem entsprechenden Alias auf dem HSM vorhanden ist, und ordnet dann das bereitgestellte Zertifikat dem entsprechenden Schlüsselpaar zu.

Themen

Voraussetzungen

Um den AWS CloudHSM-Schlüsselspeicher zu verwenden, müssen Sie zuerst das AWS CloudHSM JCE SDK initialisieren und konfigurieren.

Schritt 1: Installieren der JCE

Um das JCE, einschließlich der AWS CloudHSM-Client-Voraussetzungen, zu installieren, folgen Sie den Schritten zur Installation der Java-Bibliothek.

Schritt 2: Hinzufügen von HSM-Anmeldeinformationen zu Umgebungsvariablen

Richten Sie Umgebungsvariablen so ein, dass sie Ihre HSM-Anmeldeinformationen enthalten.

Linux

```
$ export HSM_USER=<HSM user name>
```

```
$ export HSM_PASSWORD=<HSM password>
```

Windows

```
PS C:\> $Env:HSM_USER=<HSM user name>
```

```
PS C:\> $Env:HSM_PASSWORD=<HSM password>
```

> **Note**
>
> Die AWS CloudHSM JCE bietet verschiedene Anmeldeoptionen. Um den AWS CloudHSM-Schlüsselspeicher mit Anwendungen von Drittanbietern zu verwenden, müssen Sie die implizite Anmeldung mit Umgebungsvariablen verwenden. Wenn Sie die explizite Anmeldung per Anwendungscode verwenden möchten, müssen Sie Ihre eigene Anwendung mithilfe des AWS CloudHSM-Schlüsselspeichers erstellen. Weitere Informationen finden Sie im Artikel zum Verwenden von AWS CloudHSM Key Store.

Schritt 3: Registrieren des JCE-Providers

Gehen Sie wie folgt vor, um den JCE-Anbieter in der Java- CloudProvider Konfiguration zu registrieren:

1. Öffnen Sie die `java.security`-Konfigurationsdatei in Ihrer Java-Installation zur Bearbeitung.
2. Fügen Sie in der `java.security`-Konfigurationsdatei `com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider` als letzten Anbieter hinzu. Wenn sich beispielsweise neun Anbieter in der `java.security`-Datei befinden, fügen Sie den folgenden Anbieter als letzten Anbieter in dem Abschnitt hinzu:

   ```
   security.provider.10=com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider
   ```

> **Note**
>
> Wenn Sie den AWS CloudHSM-Anbieter mit höherer Priorität hinzufügen, kann sich dies negativ auf die Leistung Ihres Systems auswirken, da dem AWS CloudHSM-Anbieter bei Vorgängen, die sicher in Software ausgelagert werden können, Priorität eingeräumt wird. Es hat sich bewährt, immer den Anbieter anzugeben, den Sie für einen Vorgang

verwenden möchten, unabhängig davon, ob es sich um das AWS CloudHSM oder einen softwarebasierten Anbieter handelt.

> **Note**
>
> Die Angabe von `-providerName-`, `-providerclass-` und `-providerpath-` Befehlszeilenoptionen beim Generieren von Schlüsseln mithilfe von Keytool mit dem AWS-CloudHSM-Schlüsselspeicher kann zu Fehlern führen.

Verwendung von AWS CloudHSM Key Store mit Keytool

Keytool ist ein beliebtes Befehlszeilenprogramm für allgemeine Schlüssel- und Zertifikataufgaben. Ein vollständiges Tutorial zu keytool liegt außerhalb des Umfangs der AWS CloudHSM-Dokumentation. In diesem Artikel werden die spezifischen Parameter erläutert, die Sie mit verschiedenen Keytool-Funktionen verwenden sollten, wenn Sie AWS CloudHSM als Stammverzeichnis des Vertrauens über den AWS CloudHSM-Schlüsselspeicher verwenden.

Wenn Sie keytool mit dem AWS CloudHSM-Schlüsselspeicher verwenden, geben Sie für jeden keytool-Befehl die folgenden Argumente an:

Linux

```
-storetype CLOUDHSM -J-classpath< '-J/opt/cloudhsm/java/*'>
```

Windows

```
-storetype CLOUDHSM -J-classpath<'-J"C:\Program Files\Amazon\CloudHSM\java\*"'>
```

Informationen zum Erstellen einer neuen Schlüsselspeicherdatei mithilfe des AWS CloudHSM-Schlüsselspeichers finden Sie unter Verwenden von AWS CloudHSM KeyStore. Um einen vorhandenen Schlüsselspeicher zu verwenden, geben Sie seinen Namen (einschließlich Pfad) mit dem Argument -keystore zu keytool an. Wenn Sie eine nicht vorhandene Schlüsselspeicherdatei in einem keytool-Befehl angeben, erstellt der AWS CloudHSM-Schlüsselspeicher eine neue Schlüsselspeicherdatei.

Erstellen neuer Schlüssel mit Keytool

Sie können keytool verwenden, um RSA-, AES- und DESede-Schlüssel zu erzeugen, die von JCE SDK des AWS CloudHSM unterstützt werden.

> **Important**
>
> Ein Schlüssel, der über keytool generiert wird, wird in Software generiert und dann in AWS CloudHSM als extrahierbarer, persistenter Schlüssel importiert.

Wir empfehlen nachdrücklich, nicht exportierbare Schlüssel außerhalb von keytool zu generieren und dann entsprechende Zertifikate in den Schlüsselspeicher zu importieren. Wenn Sie extrahierbare RSA- oder EC-Schlüssel über keytool und jarsigner verwenden, exportieren die Anbieter Schlüssel aus dem AWS CloudHSM und verwenden dann den Schlüssel lokal für Signaturvorgänge.

Wenn mehrere Client-Instances mit Ihrem AWS CloudHSM-Cluster verbunden sind, beachten Sie, dass das Importieren eines Zertifikats im Schlüsselspeicher einer Client-Instance die Zertifikate nicht automatisch auf anderen Client-Instances verfügbar macht. Um den Schlüssel und die zugehörigen Zertifikate auf jeder Client-Instance zu registrieren, müssen Sie eine Java-Anwendung ausführen, wie in the section called “Erstellen eines CSR mit Keytool” beschrieben. Alternativ können Sie die erforderlichen Änderungen auf einem Client vornehmen und die resultierende Schlüsselspeicherdatei auf jede andere Client-Instanz kopieren.

Beispiel 1: So generieren Sie einen symmetrischen AES-256-Schlüssel und speichern ihn in einer Schlüsselspeicherdatei namens „my_keystore.store“ im Arbeitsverzeichnis. Ersetzen Sie *<secret label>* durch ein eindeutiges Label.

Linux

```
$ keytool -genseckey -alias <secret label> -keyalg aes \
 -keysize 256 -keystore my_keystore.store \
 -storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
```

Windows

```
PS C:\> keytool -genseckey -alias <secret label> -keyalg aes `
 -keysize 256 -keystore my_keystore.store `
 -storetype CloudHSM -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Beispiel 2: So generieren Sie ein RSA 2048-Schlüsselpaar und speichern es in einer Schlüsselspeicherdatei namens „my_keystore.store“ im Arbeitsverzeichnis. Ersetzen Sie *<RSA key pair label>* durch ein eindeutiges Label.

Linux

```
$ keytool -genkeypair -alias <RSA key pair label> \
 -keyalg rsa -keysize 2048 \
 -sigalg sha512withrsa \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -genkeypair -alias <RSA key pair label> `
 -keyalg rsa -keysize 2048 `
 -sigalg sha512withrsa `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Sie finden eine Liste der unterstützten Signaturalgorithmen in der Java-Bibliothek.

Löschen eines Schlüssels mit Keytool

Der AWS CloudHSM-Schlüsselspeicher unterstützt das Löschen von Schlüsseln nicht. Sie können Schlüssel mit der Destroyable-Methode der Destroyable-Schnittstelle löschen.

```
((Destroyable) key).destroy();
```

Erstellen eines CSR mit Keytool

Sie erhalten die größte Flexibilität beim Generieren einer Zertifikatsignieranforderung (Certificate Signing Request, CSR), wenn Sie die OpenSSL Dynamic Engine verwenden. Der folgende Befehl verwendet keytool, um eine CSR für ein Schlüsselpaar mit dem Alias, `my-key-pair`, zu generieren.

Linux

```
$ keytool -certreq -alias <key pair label> \
 -file my_csr.csr \
```

```
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -certreq -alias <key pair label> `
 -file my_csr.csr `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

> **Note**
>
> Um ein Schlüsselpaar von keytool zu verwenden, muss dieses Schlüsselpaar einen Eintrag in der angegebenen Schlüsselspeicherdatei haben. Wenn Sie ein Schlüsselpaar verwenden möchten, das außerhalb von keytool generiert wurde, müssen Sie die Schlüssel- und Zertifikatmetadaten in den Schlüsselspeicher importieren. Anweisungen zum Importieren der Keystore-Daten finden Sie unter the section called “Verwenden von Keytool zum Importieren von Zwischen- und Stammzertifikaten in den AWS CloudHSM Key Store ”.

Verwenden von Keytool zum Importieren von Zwischen- und Stammzertifikaten in den AWS CloudHSM Key Store

Um ein CA-Zertifikat zu importieren, müssen Sie die Überprüfung einer vollständigen Zertifikatkette für ein neu importiertes Zertifikat aktivieren. Im Folgenden wird ein Beispielbefehl gezeigt.

Linux

```
$ keytool -import -trustcacerts -alias rootCAcert \
 -file rootCAcert.cert -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -import -trustcacerts -alias rootCAcert `
 -file rootCAcert.cert -keystore my_keystore.store `
 -storetype CLOUDHSM `
```

```
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Wenn Sie mehrere Client-Instanzen mit Ihrem AWS CloudHSM-Cluster verbinden, wird das Zertifikat beim Importieren eines Zertifikats im Schlüsselspeicher einer Client-Instanz nicht automatisch auf anderen Client-Instanzen verfügbar gemacht. Sie müssen das Zertifikat auf jeder Client-Instanz importieren.

Verwenden von Keytool zum Löschen von Zertifikaten aus dem AWS CloudHSM Key Store

Der folgende Befehl zeigt ein Beispiel für das Löschen eines Zertifikats aus einem Java-Keytool-Schlüsselspeicher.

Linux

```
$ keytool -delete -alias mydomain \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -delete -alias mydomain `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Wenn Sie mehrere Client-Instanzen mit Ihrem AWS CloudHSM-Cluster verbinden, wird das Zertifikat beim Löschen eines Zertifikats im Schlüsselspeicher einer Client-Instanz nicht automatisch von anderen Client-Instanzen entfernt. Sie müssen das Zertifikat auf jeder Client-Instanz löschen.

Importieren eines Arbeitszertifikats in den AWS CloudHSM Key Store mit Keytool

Sobald eine Zertifikatsignieranforderung (CSR) signiert ist, können Sie sie in den AWS CloudHSM-Schlüsselspeicher importieren und mit dem entsprechenden Schlüsselpaar verknüpfen. Der folgende Befehl bietet ein Beispiel.

Linux

```
$ keytool -importcert -noprompt -alias <key pair label> \
 -file my_certificate.crt \
```

```
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -importcert -noprompt -alias <key pair label> `
 -file my_certificate.crt `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Der Alias sollte ein Schlüsselpaar mit einem zugeordneten Zertifikat im Schlüsselspeicher sein. Wenn der Schlüssel außerhalb von keytool oder auf einer anderen Client-Instanz generiert wird, müssen Sie zuerst die Schlüssel- und Zertifikatmetadaten in den Schlüsselspeicher importieren.

Die Zertifikatkette muss überprüfbar sein. Wenn Sie das Zertifikat nicht überprüfen können, müssen Sie möglicherweise das Signaturzertifikat (Zertifizierungsstelle) in den Schlüsselspeicher importieren, damit die Kette überprüft werden kann.

Exportieren eines Zertifikats mit Keytool

Im folgenden Beispiel wird ein Zertifikat im Binärformat X.509 generiert. Um ein menschlich lesbares Zertifikat zu exportieren, fügen Sie dem `-exportcert`-Befehl `-rfc` hinzu.

Linux

```
$ keytool -exportcert -alias <key pair label> \
 -file my_exported_certificate.crt \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -exportcert -alias <key pair label> `
 -file my_exported_certificate.crt `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Verwenden von AWS CloudHSM Key Store mit Jarsigner

Jarsigner ist ein beliebtes Befehlszeilendienstprogramm zum Signieren von JAR-Dateien mit einem Schlüssel, der sicher auf einem HSM gespeichert ist. Ein komplettes Tutorial zu Jarsigner liegt außerhalb der AWS CloudHSM-Dokumentation. In diesem Abschnitt werden die Jarsigner-Parameter erläutert, mit denen Sie Signaturen AWS CloudHSM als Vertrauenswurzel über den AWS CloudHSM-Schlüsselspeicher signieren und überprüfen sollten.

Einrichten von Schlüsseln und Zertifikaten

Bevor Sie JAR-Dateien mit Jarsigner signieren können, stellen Sie sicher, dass Sie die folgenden Schritte eingerichtet oder ausgeführt haben:

1. Folgen Sie den Anweisungen unter AWS CloudHSM-Schlüsselspeichervoraussetzungen.
2. Richten Sie Ihre Signaturschlüssel und die zugehörigen Zertifikate und Zertifikatkette ein, die im AWS CloudHSM-Schlüsselspeicher des aktuellen Servers oder der aktuellen Client-Instanz gespeichert werden sollen. Erstellen Sie die Schlüssel auf dem AWS CloudHSM, und importieren Sie dann die zugehörigen Metadaten in Ihren AWS CloudHSM-Schlüsselspeicher. Informationen zum Einrichten der Schlüssel und Zertifikate mit keytool finden Sie unter the section called “Erstellen neuer Schlüssel mit Keytool”. Wenn Sie mehrere Client-Instanzen verwenden, um Ihre JARs zu signieren, erstellen Sie den Schlüssel, und importieren Sie die Zertifikatkette. Kopieren Sie dann die resultierende Schlüsselspeicherdatei auf jede Client-Instanz. Wenn Sie häufig neue Schlüssel generieren, ist es möglicherweise einfacher, Zertifikate einzeln in jede Client-Instanz zu importieren.
3. Die gesamte Zertifikatskette sollte überprüfbar sein. Damit die Zertifikatkette überprüft werden kann, müssen Sie möglicherweise das CA-Zertifikat und Zwischenzertifikate dem AWS CloudHSM-Schlüsselspeicher hinzufügen. Im Codeausschnitt in the section called “Signieren einer JAR-Datei mit AWS CloudHSM und Jarsigner” finden Sie eine Anleitung zur Verwendung von Java-Code zur Überprüfung der Zertifikatskette. Wenn Sie möchten, können Sie keytool verwenden, um Zertifikate zu importieren. Eine Anleitung zur Verwendung von keytool finden Sie unter the section called “Verwenden von Keytool zum Importieren von Zwischen- und Stammzertifikaten in den AWS CloudHSM Key Store ”.

Signieren einer JAR-Datei mit AWS CloudHSM und Jarsigner

Verwenden Sie den folgenden Befehl, um eine JAR-Datei zu signieren:

Linux;

Für OpenJDK 8

```
jarsigner -keystore my_keystore.store \
 -signedjar signthisclass_signed.jar \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass.jar <key pair label>
```

Für OpenJDK 11, OpenJDK 17 und OpenJDK 21

```
jarsigner -keystore my_keystore.store \
 -signedjar signthisclass_signed.jar \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass.jar <key pair label>
```

Windows

Für OpenJDK8

```
jarsigner -keystore my_keystore.store `
 -signedjar signthisclass_signed.jar `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*;C:\Program Files\Java
\jdk1.8.0_331\lib\tools.jar' `
  "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass.jar <key pair label>
```

Für OpenJDK 11, OpenJDK 17 und OpenJDK 21

```
jarsigner -keystore my_keystore.store `
 -signedjar signthisclass_signed.jar `
```

```
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*'`
  "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass.jar <key pair label>
```

Verwenden Sie den folgenden Befehl, um eine signierte JAR zu überprüfen:

Linux

Für OpenJDK8

```
jarsigner -verify \
 -keystore my_keystore.store \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass_signed.jar <key pair label>
```

Für OpenJDK 11, OpenJDK 17 und OpenJDK 21

```
jarsigner -verify \
 -keystore my_keystore.store \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass_signed.jar <key pair label>
```

Windows

Für OpenJDK 8

```
jarsigner -verify `
 -keystore my_keystore.store `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
```

```
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*;C:\Program Files\Java
\jdk1.8.0_331\lib\tools.jar' `
 "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass_signed.jar <key pair label>
```

Für OpenJDK 11, OpenJDK 17 und OpenJDK 21

```
jarsigner -verify `
 -keystore my_keystore.store `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*`
 "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass_signed.jar <key pair label>
```

Bekannte Probleme

1. Wir unterstützen keine EC-Schlüssel mit Keytool und Jarsigner.

Verwendung von Client-SDK 3 zur Integration mit Java Keytool und Jarsigner

AWS CloudHSM Key Store ist ein spezieller JCE-Schlüsselspeicher, der Zertifikate verwendet, die mit Schlüsseln auf Ihrem HSM über Tools von Drittanbietern wie `keytool` und `jarsigner` verknüpft sind. AWS CloudHSM speichert keine Zertifikate auf dem HSM, da Zertifikate öffentliche, nicht vertrauliche Daten sind. Der AWS CloudHSM-Schlüsselspeicher speichert die Zertifikate in einer lokalen Datei und ordnet sie den entsprechenden Schlüsseln auf Ihrem HSM zu.

Wenn Sie den AWS CloudHSM-Schlüsselspeicher verwenden, um neue Schlüssel zu generieren, werden keine Einträge in der lokalen Schlüsselspeicherdatei generiert - die Schlüssel werden auf dem HSM erstellt. Wenn Sie den AWS CloudHSM-Schlüsselspeicher verwenden, um nach Schlüsseln zu suchen, wird die Suche an das HSM übergeben. Wenn Sie Zertifikate im AWS CloudHSM-Schlüsselspeicher speichern, überprüft der Anbieter, ob ein Schlüsselpaar mit dem entsprechenden Alias auf dem HSM vorhanden ist, und ordnet dann das bereitgestellte Zertifikat dem entsprechenden Schlüsselpaar zu.

Themen

Voraussetzungen

Um den AWS CloudHSM-Schlüsselspeicher zu verwenden, müssen Sie zuerst das AWS CloudHSM JCE SDK initialisieren und konfigurieren.

Schritt 1: Installieren der JCE

Um das JCE, einschließlich der AWS CloudHSM-Client-Voraussetzungen, zu installieren, folgen Sie den Schritten zur Installation der Java-Bibliothek.

Schritt 2: Hinzufügen von HSM-Anmeldeinformationen zu Umgebungsvariablen

Richten Sie Umgebungsvariablen so ein, dass sie Ihre HSM-Anmeldeinformationen enthalten.

```
export HSM_PARTITION=PARTITION_1
export HSM_USER=<HSM user name>
export HSM_PASSWORD=<HSM password>
```

> **Note**
>
> Die CloudHSM JCE bietet verschiedene Anmeldeoptionen. Um den AWS CloudHSM-Schlüsselspeicher mit Anwendungen von Drittanbietern zu verwenden, müssen Sie die implizite Anmeldung mit Umgebungsvariablen verwenden. Wenn Sie die explizite Anmeldung per Anwendungscode verwenden möchten, müssen Sie Ihre eigene Anwendung mithilfe des AWS CloudHSM-Schlüsselspeichers erstellen. Weitere Informationen finden Sie im Artikel zum Verwenden von AWS CloudHSM Key Store.

Schritt 3: Registrieren des JCE-Providers

Um den JCE-Anbieter zu registrieren, verwenden Sie in der Java- CloudProvider Konfiguration.

1. Öffnen Sie die java.security-Konfigurationsdatei in Ihrer Java-Installation zur Bearbeitung.
2. Fügen Sie in der Konfigurationsdatei java.security `com.cavium.provider.CaviumProvider` als letzten Provider hinzu. Wenn sich beispielsweise neun Provider in der Datei java.security befinden, fügen Sie den folgenden Provider als letzten Provider in dem Abschnitt hinzu. Das Hinzufügen des Cavium-Providers mit höherer Priorität kann sich negativ auf die Leistung Ihres Systems auswirken.

   ```
   security.provider.10=com.cavium.provider.CaviumProvider
   ```

   > **Note**
   >
   > Intensive Nutzer können daran gewöhnt sein, `-providerName-`, `-providerclass-` und `-providerpath`-Befehlszeilenoptionen bei Verwendung von keytool anzugeben, anstatt die Sicherheitskonfigurationsdatei zu aktualisieren. Wenn Sie versuchen, beim Generieren von Schlüsseln mit dem AWS CloudHSM-Schlüsselspeicher Befehlszeilenoptionen anzugeben, führt dies zu Fehlern.

Verwendung von AWS CloudHSM Key Store mit Keytool

Keytool ist ein beliebtes Befehlszeilenprogramm für allgemeine Schlüssel- und Zertifikataufgaben auf Linux-Systemen. Ein vollständiges Tutorial zu keytool liegt außerhalb des Umfangs der AWS CloudHSM-Dokumentation. In diesem Artikel werden die spezifischen Parameter erläutert, die Sie mit verschiedenen Keytool-Funktionen verwenden sollten, wenn Sie AWS CloudHSM als Stammverzeichnis des Vertrauens über den AWS CloudHSM-Schlüsselspeicher verwenden.

Wenn Sie keytool mit dem AWS CloudHSM-Schlüsselspeicher verwenden, geben Sie für jeden keytool-Befehl die folgenden Argumente an:

```
-storetype CLOUDHSM \
  -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib
```

Informationen zum Erstellen einer neuen Schlüsselspeicherdatei mithilfe des AWS CloudHSM-Schlüsselspeichers finden Sie unter Verwenden von AWS CloudHSM KeyStore. Um einen vorhandenen Schlüsselspeicher zu verwenden, geben Sie seinen Namen (einschließlich Pfad) mit dem Argument -keystore zu keytool an. Wenn Sie eine nicht vorhandene Schlüsselspeicherdatei

in einem keytool-Befehl angeben, erstellt der AWS CloudHSM-Schlüsselspeicher eine neue Schlüsselspeicherdatei.

Erstellen neuer Schlüssel mit Keytool

Sie können keytool verwenden, um jede Art von Schlüssel zu generieren, die vom AWS CloudHSM-JCE SDK unterstützt wird. Eine vollständige Liste der Schlüssel und Längen finden Sie im Artikel Unterstützte Schlüssel in der Java-Bibliothek.

> **Important**
>
> Ein Schlüssel, der über keytool generiert wird, wird in Software generiert und dann in AWS CloudHSM als extrahierbarer, persistenter Schlüssel importiert.

Anweisungen zum Erstellen nicht extrahierbarer Schlüssel direkt auf dem HSM und ihrer Verwendung mit keytool oder Jarsigner werden im Codebeispiel unter Registrieren vorhandener Schlüssel im AWS CloudHSM Key Store gegeben. Wir empfehlen nachdrücklich, nicht exportierbare Schlüssel außerhalb von keytool zu generieren und dann entsprechende Zertifikate in den Schlüsselspeicher zu importieren. Wenn Sie extrahierbare RSA- oder EC-Schlüssel über keytool und jarsigner verwenden, exportieren die Provider Schlüssel aus dem AWS CloudHSM und verwenden dann den Schlüssel lokal für Signaturvorgänge.

Wenn mehrere Client-Instanzen mit Ihrem CloudHSM-Cluster verbunden sind, beachten Sie, dass das Importieren eines Zertifikats im Schlüsselspeicher einer Clientinstanz die Zertifikate nicht automatisch auf anderen Client-Instanzen verfügbar macht. Um den Schlüssel und die zugehörigen Zertifikate auf jeder Client-Instanz zu registrieren, müssen Sie eine Java-Anwendung ausführen, wie unter Generieren eines CSR mit Keytool beschrieben. Alternativ können Sie die erforderlichen Änderungen auf einem Client vornehmen und die resultierende Schlüsselspeicherdatei auf jede andere Client-Instanz kopieren.

Beispiel 1: So generieren Sie einen symmetrischen AES-256-Schlüssel und speichern ihn in einer Schlüsselspeicherdatei namens „my_keystore.store“ im Arbeitsverzeichnis. Ersetzen Sie *<secret label>* durch ein eindeutiges Label.

```
keytool -genseckey -alias <secret label> -keyalg aes \
  -keysize 256 -keystore my_keystore.store \
  -storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib/
```

Beispiel 2: So generieren Sie ein RSA 2048-Schlüsselpaar und speichern es in einer Schlüsselspeicherdatei namens „my_keystore.store“ im Arbeitsverzeichnis. Ersetzen Sie *<RSA key pair label>* durch ein eindeutiges Label.

```
keytool -genkeypair -alias <RSA key pair label> \
        -keyalg rsa -keysize 2048 \
        -sigalg sha512withrsa \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Beispiel 3: So generieren Sie einen p256 ED-Schlüssel und speichern ihn in einer Schlüsselspeicherdatei namens „my_keystore.store“ im Arbeitsverzeichnis. Ersetzen Sie *<ec key pair label>* durch ein eindeutiges Label.

```
keytool -genkeypair -alias <ec key pair label> \
        -keyalg ec -keysize 256 \
        -sigalg SHA512withECDSA \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Sie finden eine Liste der unterstützten Signaturalgorithmen in der Java-Bibliothek.

Löschen eines Schlüssels mit Keytool

Der AWS CloudHSM-Schlüsselspeicher unterstützt das Löschen von Schlüsseln nicht. Um Schlüssel zu löschen, müssen Sie die `deleteKey`-Funktion des AWS CloudHSM-Befehlszeilen-Tools, deleteKey, verwenden.

Erstellen eines CSR mit Keytool

Sie erhalten die größte Flexibilität beim Generieren einer Zertifikatsignieranforderung (Certificate Signing Request, CSR), wenn Sie die OpenSSL Dynamic Engine verwenden. Der folgende Befehl verwendet keytool, um eine CSR für ein Schlüsselpaar mit dem Alias, `my-key-pair`, zu generieren.

```
keytool -certreq -alias <key pair label> \
        -file my_csr.csr \
        -keystore my_keystore.store \
```

```
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

> **Note**
>
> Um ein Schlüsselpaar von keytool zu verwenden, muss dieses Schlüsselpaar einen Eintrag in der angegebenen Schlüsselspeicherdatei haben. Wenn Sie ein Schlüsselpaar verwenden möchten, das außerhalb von keytool generiert wurde, müssen Sie die Schlüssel- und Zertifikatmetadaten in den Schlüsselspeicher importieren. Anweisungen zum Importieren der Schlüsselspeicherdaten finden Sie unter Importieren von Zwischen- und Stammzertifikaten in AWS CloudHSM Key Store mit Keytool.

Verwenden von Keytool zum Importieren von Zwischen- und Stammzertifikaten in den AWS CloudHSM Key Store

Um ein CA-Zertifikat zu importieren, müssen Sie die Überprüfung einer vollständigen Zertifikatkette für ein neu importiertes Zertifikat aktivieren. Im Folgenden wird ein Beispielbefehl gezeigt.

```
keytool -import -trustcacerts -alias rootCAcert \
        -file rootCAcert.cert -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Wenn Sie mehrere Client-Instanzen mit Ihrem AWS CloudHSM-Cluster verbinden, wird das Zertifikat beim Importieren eines Zertifikats im Schlüsselspeicher einer Client-Instanz nicht automatisch auf anderen Client-Instanzen verfügbar gemacht. Sie müssen das Zertifikat auf jeder Client-Instanz importieren.

Verwenden von Keytool zum Löschen von Zertifikaten aus dem AWS CloudHSM Key Store

Der folgende Befehl zeigt ein Beispiel für das Löschen eines Zertifikats aus einem Java-Keytool-Schlüsselspeicher.

```
keytool -delete -alias mydomain -keystore \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
```

```
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Wenn Sie mehrere Client-Instanzen mit Ihrem AWS CloudHSM-Cluster verbinden, wird das Zertifikat beim Löschen eines Zertifikats im Schlüsselspeicher einer Client-Instanz nicht automatisch von anderen Client-Instanzen entfernt. Sie müssen das Zertifikat auf jeder Client-Instanz löschen.

Importieren eines Arbeitszertifikats in den AWS CloudHSM Key Store mit Keytool

Sobald eine Zertifikatsignieranforderung (CSR) signiert ist, können Sie sie in den AWS CloudHSM-Schlüsselspeicher importieren und mit dem entsprechenden Schlüsselpaar verknüpfen. Der folgende Befehl bietet ein Beispiel.

```
keytool -importcert -noprompt -alias <key pair label> \
        -file my_certificate.crt \
        -keystore my_keystore.store
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Der Alias sollte ein Schlüsselpaar mit einem zugeordneten Zertifikat im Schlüsselspeicher sein. Wenn der Schlüssel außerhalb von keytool oder auf einer anderen Client-Instanz generiert wird, müssen Sie zuerst die Schlüssel- und Zertifikatmetadaten in den Schlüsselspeicher importieren. Anweisungen zum Importieren der Zertifikatmetadaten finden Sie im Codebeispiel unter Registrieren vorhandener Schlüssel im AWS CloudHSM Key Store.

Die Zertifikatkette muss überprüfbar sein. Wenn Sie das Zertifikat nicht überprüfen können, müssen Sie möglicherweise das Signaturzertifikat (Zertifizierungsstelle) in den Schlüsselspeicher importieren, damit die Kette überprüft werden kann.

Exportieren eines Zertifikats mit Keytool

Im folgenden Beispiel wird ein Zertifikat im Binärformat X.509 generiert. Um ein menschlich lesbares Zertifikat zu exportieren, fügen Sie dem `-exportcert`-Befehl `-rfc` hinzu.

```
keytool -exportcert -alias <key pair label> \
        -file my_exported_certificate.crt \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Verwenden von AWS CloudHSM Key Store mit Jarsigner

Jarsigner ist ein beliebtes Befehlszeilendienstprogramm zum Signieren von JAR-Dateien mit einem Schlüssel, der sicher auf einem HSM gespeichert ist. Ein komplettes Tutorial zu Jarsigner liegt außerhalb der AWS CloudHSM-Dokumentation. In diesem Abschnitt werden die Jarsigner-Parameter erläutert, mit denen Sie Signaturen AWS CloudHSM als Vertrauenswurzel über den AWS CloudHSM-Schlüsselspeicher signieren und überprüfen sollten.

Einrichten von Schlüsseln und Zertifikaten

Bevor Sie JAR-Dateien mit Jarsigner signieren können, stellen Sie sicher, dass Sie die folgenden Schritte eingerichtet oder ausgeführt haben:

1. Folgen Sie den Anweisungen unter AWS CloudHSM-Schlüsselspeichervoraussetzungen .
2. Richten Sie Ihre Signaturschlüssel und die zugehörigen Zertifikate und Zertifikatkette ein, die im AWS CloudHSM-Schlüsselspeicher des aktuellen Servers oder der aktuellen Client-Instanz gespeichert werden sollen. Erstellen Sie die Schlüssel auf dem AWS CloudHSM, und importieren Sie dann die zugehörigen Metadaten in Ihren AWS CloudHSM-Schlüsselspeicher. Verwenden Sie das Codebeispiel unter Registrieren vorhandener Schlüssel beim AWS CloudHSM Key Store, um Metadaten in den Schlüsselspeicher zu importieren. Informationen zum Einrichten der Schlüssel und Zertifikate mit keytool finden Sie unter Erstellen neuer Schlüssel mit Keytool. Wenn Sie mehrere Client-Instanzen verwenden, um Ihre JARs zu signieren, erstellen Sie den Schlüssel, und importieren Sie die Zertifikatkette. Kopieren Sie dann die resultierende Schlüsselspeicherdatei auf jede Client-Instanz. Wenn Sie häufig neue Schlüssel generieren, ist es möglicherweise einfacher, Zertifikate einzeln in jede Client-Instanz zu importieren.
3. Die gesamte Zertifikatskette sollte überprüfbar sein. Damit die Zertifikatkette überprüft werden kann, müssen Sie möglicherweise das CA-Zertifikat und Zwischenzertifikate dem AWS CloudHSM-Schlüsselspeicher hinzufügen. Vgl. das Codebeispiel in Signieren einer JAR-Datei mit AWS CloudHSM und Jarsigner für eine Anleitung zur Verwendung von Java-Code zur Überprüfung der Zertifikatkette. Wenn Sie möchten, können Sie keytool verwenden, um Zertifikate zu importieren. Anweisungen zur Verwendung von keytool finden Sie unter Verwenden von Keytool zum Importieren von Zwischen- und Stammzertifikaten in den AWS CloudHSM-Schlüsselspeicher.

Signieren einer JAR-Datei mit AWS CloudHSM und Jarsigner

Verwenden Sie den folgenden Befehl, um eine JAR-Datei zu signieren:

```
jarsigner -keystore my_keystore.store \
```

```
        -signedjar signthisclass_signed.jar \
        -sigalg sha512withrsa \
        -storetype CloudHSM \
        -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
        -J-Djava.library.path=/opt/cloudhsm/lib \
        signthisclass.jar <key pair label>
```

Verwenden Sie den folgenden Befehl, um eine signierte JAR zu überprüfen:

```
jarsigner -verify \
        -keystore my_keystore.store \
        -sigalg sha512withrsa \
        -storetype CloudHSM \
        -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
        -J-Djava.library.path=/opt/cloudhsm/lib \
        signthisclass_signed.jar <key pair label>
```

Bekannte Probleme

Die folgende Liste enthält die aktuelle Liste bekannter Probleme.

- Beim Generieren von Schlüsseln mit keytool kann der erste Anbieter in der Anbieterkonfiguration nicht sein CaviumProvider.
- Beim Generieren von Schlüsseln mit keytool wird der erste (unterstützte) Provider in der Sicherheitskonfigurationsdatei verwendet, um den Schlüssel zu generieren. Dies ist in der Regel ein Software-Provider. Der generierte Schlüssel erhält dann einen Alias und wird während der Hinzufügung des Schlüssels als persistenter (Token-) Schlüssel in das AWS CloudHSM-HSM importiert.
- Wenn Sie keytool mit dem AWS CloudHSM-Schlüsselspeicher verwenden geben Sie nicht die Option `-providerName`, `-providerclass` oder `-providerpath` Optionen in der Befehlszeile an. Geben Sie diese Optionen in der Sicherheitsanbieterdatei an, wie in den Key Store-Voraussetzungen beschrieben.
- Wenn nicht extrahierbare EC-Schlüssel über keytool und Jarsigner verwendet werden, muss der SunEC-Provider aus der Liste der Provider in der java.security-Datei entfernt/deaktiviert werden. Wenn Sie extrahierbare EC-Schlüssel über keytool und Jarsigner verwenden, exportieren die Provider Schlüsselbits aus dem AWS CloudHSM-HSM und verwenden den Schlüssel lokal für Signaturvorgänge. Wir empfehlen nicht, exportierbare Schlüssel mit keytool oder Jarsigner zu verwenden.

Registrieren von bereits vorhandenen Schlüsseln beim AWS CloudHSM Key Store

Für maximale Sicherheit und Flexibilität bei Attributen und Benennungen empfehlen wir, die Signaturschlüssel mit key_mgmt_util zu generieren. Sie können auch eine Java-Anwendung verwenden, um den Schlüssel in AWS CloudHSM zu generieren.

Der folgende Abschnitt enthält ein Codebeispiel, das veranschaulicht, wie ein neues Schlüsselpaar auf dem HSM generiert und mit vorhandenen Schlüsseln registriert wird, die in den AWS CloudHSM-Schlüsselspeicher importiert wurden. Die importierten Schlüssel stehen für die Verwendung mit Tools von Drittanbietern wie keytool und Jarsigner zur Verfügung.

Wenn Sie einen bereits vorhandenen Schlüssel verwenden möchten, ändern Sie das Codebeispiel so, dass es nach einem Schlüssel nach Benennung sucht, anstatt einen neuen Schlüssel zu generieren. Beispielcode für die Suche nach einem Schlüssel nach Label ist im KeyUtilitiesRunnerJava-Beispiel auf verfügbar GitHub.

> **Important**
>
> Beim Registrieren eines Schlüssels, der auf AWS CloudHSM mit einem lokalen Schlüssel gespeichert ist, wird der Schlüssel nicht exportiert. Wenn der Schlüssel registriert ist, registriert der Schlüsselspeicher den Alias (oder die Benennung) des Schlüssels und korreliert lokal Speicherzertifikatobjekte mit einem Schlüsselpaar auf dem AWS CloudHSM. Solange das Schlüsselpaar als nicht exportierbar erstellt wird, verlassen die Schlüsselbits das HSM nicht.

```
                    //
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a copy of
this
// software and associated documentation files (the "Software"), to deal in the
Software
// without restriction, including without limitation the rights to use, copy, modify,
// merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
// permit persons to whom the Software is furnished to do so.
//
```

```
 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
 // INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 // PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
 // HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
 // OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 // SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 //

package com.amazonaws.cloudhsm.examples;

import com.cavium.key.CaviumKey;
import com.cavium.key.parameter.CaviumAESKeyGenParameterSpec;
import com.cavium.key.parameter.CaviumRSAKeyGenParameterSpec;
import com.cavium.asn1.Encoder;
import com.cavium.cfm2.Util;

import javax.crypto.KeyGenerator;

import java.io.ByteArrayInputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;

import java.math.BigInteger;

import java.security.*;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.security.KeyStore.PasswordProtection;
import java.security.KeyStore.PrivateKeyEntry;
import java.security.KeyStore.Entry;

import java.util.Calendar;
import java.util.Date;
import java.util.Enumeration;

//
// KeyStoreExampleRunner demonstrates how to load a keystore, and associate a
 certificate with a
// key in that keystore.
```

```
//
// This example relies on implicit credentials, so you must setup your environment
 correctly.
//
// https://docs.aws.amazon.com/cloudhsm/latest/userguide/java-library-
install.html#java-library-credentials
//

public class KeyStoreExampleRunner {

    private static byte[] COMMON_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x03 };
    private static byte[] COUNTRY_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x06 };
    private static byte[] LOCALITY_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x07 };
    private static byte[] STATE_OR_PROVINCE_NAME_OID = new byte[] { (byte) 0x55,
 (byte) 0x04, (byte) 0x08 };
    private static byte[] ORGANIZATION_NAME_OID = new byte[] { (byte) 0x55, (byte)
 0x04, (byte) 0x0A };
    private static byte[] ORGANIZATION_UNIT_OID = new byte[] { (byte) 0x55, (byte)
 0x04, (byte) 0x0B };

    private static String helpString = "KeyStoreExampleRunner%n" +
           "This sample demonstrates how to load and store keys using a keystore.%n%n"
 +
           "Options%n" +
           "\t--help\t\t\tDisplay this message.%n" +
           "\t--store <filename>\t\tPath of the keystore.%n" +
           "\t--password <password>\t\tPassword for the keystore (not your CU
 password).%n" +
           "\t--label <label>\t\t\tLabel to store the key and certificate under.%n" +
           "\t--list\t\t\tList all the keys in the keystore.%n%n";

   public static void main(String[] args) throws Exception {
       Security.addProvider(new com.cavium.provider.CaviumProvider());
       KeyStore keyStore = KeyStore.getInstance("CloudHSM");

       String keystoreFile = null;
       String password = null;
       String label = null;
       boolean list = false;
       for (int i = 0; i < args.length; i++) {
           String arg = args[i];
```

```
            switch (args[i]) {
                case "--store":
                    keystoreFile = args[++i];
                    break;
                case "--password":
                    password = args[++i];
                    break;
                case "--label":
                    label = args[++i];
                    break;
                case "--list":
                    list = true;
                    break;
                case "--help":
                    help();
                    return;
            }
        }

        if (null == keystoreFile || null == password) {
            help();
            return;
        }

        if (list) {
            listKeys(keystoreFile, password);
            return;
        }

        if (null == label) {
            label = "Keystore Example Keypair";
        }

        //
        // This call to keyStore.load() will open the pkcs12 keystore with the supplied
        // password and connect to the HSM. The CU credentials must be specified using
        // standard CloudHSM login methods.
        //
        try {
            FileInputStream instream = new FileInputStream(keystoreFile);
            keyStore.load(instream, password.toCharArray());
        } catch (FileNotFoundException ex) {
            System.err.println("Keystore not found, loading an empty store");
            keyStore.load(null, null);
```

```
        }

        PasswordProtection passwd = new PasswordProtection(password.toCharArray());
        System.out.println("Searching for example key and certificate...");

        PrivateKeyEntry keyEntry = (PrivateKeyEntry) keyStore.getEntry(label, passwd);
        if (null == keyEntry) {
            //
            // No entry was found, so we need to create a key pair and associate a
certificate.
            // The private key will get the label passed on the command line. The
keystore alias
            // needs to be the same as the private key label. The public key will have
":public"
            // appended to it. The alias used in the keystore will We associate the
certificate
            // with the private key.
            //
            System.out.println("No entry found, creating...");
            KeyPair kp = generateRSAKeyPair(2048, label + ":public", label);
            System.out.printf("Created a key pair with the handles %d/%d%n",
((CaviumKey) kp.getPrivate()).getHandle(), ((CaviumKey) kp.getPublic()).getHandle());

            //
            // Generate a certificate and associate the chain with the private key.
            //
            Certificate self_signed_cert = generateCert(kp);
            Certificate[] chain = new Certificate[1];
            chain[0] = self_signed_cert;
            PrivateKeyEntry entry = new PrivateKeyEntry(kp.getPrivate(), chain);

            //
            // Set the entry using the label as the alias and save the store.
            // The alias must match the private key label.
            //
            keyStore.setEntry(label, entry, passwd);

            FileOutputStream outstream = new FileOutputStream(keystoreFile);
            keyStore.store(outstream, password.toCharArray());
            outstream.close();

            keyEntry = (PrivateKeyEntry) keyStore.getEntry(label, passwd);
        }
```

```
        long handle = ((CaviumKey) keyEntry.getPrivateKey()).getHandle();
        String name = keyEntry.getCertificate().toString();
        System.out.printf("Found private key %d with certificate %s%n", handle, name);
    }

    private static void help() {
        System.out.println(helpString);
    }

    //
    // Generate a non-extractable / non-persistent RSA keypair.
    // This method allows us to specify the public and private labels, which
    // will make KeyStore alises easier to understand.
    //
    public static KeyPair generateRSAKeyPair(int keySizeInBits, String publicLabel,
String privateLabel)
            throws InvalidAlgorithmParameterException, NoSuchAlgorithmException,
NoSuchProviderException {

        boolean isExtractable = false;
        boolean isPersistent = false;
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("rsa", "Cavium");
        CaviumRSAKeyGenParameterSpec spec = new
CaviumRSAKeyGenParameterSpec(keySizeInBits, new BigInteger("65537"), publicLabel,
privateLabel, isExtractable, isPersistent);

        keyPairGen.initialize(spec);

        return keyPairGen.generateKeyPair();
    }

    //
    // Generate a certificate signed by a given keypair.
    //
    private static Certificate generateCert(KeyPair kp) throws CertificateException {
        CertificateFactory cf = CertificateFactory.getInstance("X509");
        PublicKey publicKey = kp.getPublic();
        PrivateKey privateKey = kp.getPrivate();
        byte[] version = Encoder.encodeConstructed((byte) 0,
Encoder.encodePositiveBigInteger(new BigInteger("2"))); // version 1
        byte[] serialNo = Encoder.encodePositiveBigInteger(new BigInteger(1,
Util.computeKCV(publicKey.getEncoded())));

        // Use the SHA512 OID and algorithm.
```

```
        byte[] signatureOid = new byte[] {
            (byte) 0x2A, (byte) 0x86, (byte) 0x48, (byte) 0x86, (byte) 0xF7, (byte)
0x0D, (byte) 0x01, (byte) 0x01, (byte) 0x0D };
        String sigAlgoName = "SHA512WithRSA";

         byte[] signatureId = Encoder.encodeSequence(
                                        Encoder.encodeOid(signatureOid),
                                        Encoder.encodeNull());

         byte[] issuer = Encoder.encodeSequence(
                                    encodeName(COUNTRY_NAME_OID, "<Country>"),
                                    encodeName(STATE_OR_PROVINCE_NAME_OID, "<State>"),
                                    encodeName(LOCALITY_NAME_OID, "<City>"),
                                    encodeName(ORGANIZATION_NAME_OID,
"<Organization>"),
                                    encodeName(ORGANIZATION_UNIT_OID, "<Unit>"),
                                    encodeName(COMMON_NAME_OID, "<CN>")
                                );

         Calendar c = Calendar.getInstance();
         c.add(Calendar.DAY_OF_YEAR, -1);
         Date notBefore = c.getTime();
         c.add(Calendar.YEAR, 1);
         Date notAfter = c.getTime();
         byte[] validity = Encoder.encodeSequence(
                                        Encoder.encodeUTCTime(notBefore),
                                        Encoder.encodeUTCTime(notAfter)
                                    );
         byte[] key = publicKey.getEncoded();

         byte[] certificate = Encoder.encodeSequence(
                                        version,
                                        serialNo,
                                        signatureId,
                                        issuer,
                                        validity,
                                        issuer,
                                        key);
         Signature sig;
         byte[] signature = null;
         try {
             sig = Signature.getInstance(sigAlgoName, "Cavium");
             sig.initSign(privateKey);
             sig.update(certificate);
```

```
            signature = Encoder.encodeBitstring(sig.sign());

        } catch (Exception e) {
            System.err.println(e.getMessage());
            return null;
        }

        byte [] x509 = Encoder.encodeSequence(
                        certificate,
                        signatureId,
                        signature
                        );
        return cf.generateCertificate(new ByteArrayInputStream(x509));
    }

     //
     // Simple OID encoder.
     // Encode a value with OID in ASN.1 format
     //
     private static byte[] encodeName(byte[] nameOid, String value) {
         byte[] name = null;
         name = Encoder.encodeSet(
                     Encoder.encodeSequence(
                             Encoder.encodeOid(nameOid),
                             Encoder.encodePrintableString(value)
                     )
                 );
         return name;
     }

    //
    // List all the keys in the keystore.
    //
    private static void listKeys(String keystoreFile, String password) throws Exception
{
        KeyStore keyStore = KeyStore.getInstance("CloudHSM");

        try {
            FileInputStream instream = new FileInputStream(keystoreFile);
            keyStore.load(instream, password.toCharArray());
        } catch (FileNotFoundException ex) {
            System.err.println("Keystore not found, loading an empty store");
            keyStore.load(null, null);
        }
```

```
        for(Enumeration<String> entry = keyStore.aliases(); entry.hasMoreElements();) {
            System.out.println(entry.nextElement());
        }
    }

}
```

Weitere Integrationen von Drittanbietern

Mehrere Drittanbieter unterstützen AWS CloudHSM als Vertrauensanker. Dies bedeutet, dass Sie beim Erstellen und Speichern der zugrunde liegenden Schlüssel in Ihrem CloudHSM-Cluster eine Software-Lösung Ihrer Wahl nutzen können. Dadurch kann sich Ihr Workload in AWS auf die Vorteile von CloudHSM in Bezug auf Latenz, Verfügbarkeit, Zuverlässigkeit und Elastizität verlassen. Die folgende Liste enthält Drittanbieter, die CloudHSM unterstützen.

> **Note**
>
> AWS empfiehlt oder bürgt nicht (für) alle Drittanbieter.

- Hashicorp Vault ist ein Verschlüsselungsverwaltungs-Tool, das dafür entwickelt wurde, Zusammenarbeit und Governance organisationsübergreifend zu ermöglichen. Es unterstützt AWS Key Management Service und AWS CloudHSM als Vertrauensanker für zusätzlichen Schutz.
- Thycotic Secrets Server hilft Kunden, vertrauliche Anmeldeinformationen über privilegierte Konten hinweg zu verwalten. Es unterstützt AWS CloudHSM als Vertrauensanker.
- Der KMIP-Adapter von P6R ermöglicht die Nutzung Ihrer AWS CloudHSM-Instances über eine standardmäßige KMIP-Schnittstelle.
- PrimeKey EJBCA ist eine beliebte Open-Source-Lösung für PKI. Sie können dadurch auf sichere Weise Schlüsselpaare mit AWS CloudHSM erstellen und speichern.
- Box KeySafe bietet vielen Organisationen Verschlüsselungsschlüsselverwaltung für Cloud-Inhalte mit strengen Anforderungen bezüglich Sicherheit, Datenschutz und den gesetzlichen Compliance-Bestimmungen. Kunden können KeySafe-Schlüssel direkt in AWS Key Management Service oder indirekt in AWS CloudHSM über AWS KMS Custom Key Store weiter sichern.

- Insyde Software unterstützt AWS CloudHSM als Vertrauensanker für das Signieren von Firmware.
- F5 BIG-IP LTM unterstützt AWS CloudHSM als Vertrauensanker.
- Cloudera Navigator Key HSM gibt Ihnen die Möglichkeit, Ihren CloudHSM-Cluster zum Erstellen und Speichern von Schlüsseln für Cloudera Navigator Key Trustee Server zu verwenden.
- Die Venafi Trust Protection Platform bietet umfassendes Maschinenidentitätsmanagement für TLS, SSH und Codesignatur mit AWS-CloudHSM-Schlüsselgenerierung und -schutz.

Überwachung von AWS CloudHSM

Zusätzlich zu den im Client-SDK integrierten Protokollierungsfunktionen können Sie auch AWS CloudTrail, Amazon CloudWatch Logs und Amazon CloudWatch zur Überwachung von AWS CloudHSM verwenden.

Client-SDK-Protokolle

Verwenden Sie die Client-SDK-Protokollierung, um die Diagnose- und Fehlerbehebungsinformationen der von Ihnen erstellten Anwendungen zu überwachen.

CloudTrail

Verwenden Sie CloudTrail, um alle API-Aufrufe in Ihrem AWS-Konto zu überwachen, einschließlich der Aufrufe, die Sie zum Erstellen und Löschen von Clustern, Hardware-Sicherheitsmodulen (HSM) und Ressourcen-Tags tätigen.

CloudWatch Logs

Verwenden Sie CloudWatch Logs, um die Protokolle Ihrer HSM-Instances zu überwachen. Dazu gehören Ereignisse zum Erstellen und Löschen von HSM-Benutzern, zum Ändern von Benutzerpasswörtern, zum Erstellen und Löschen von Schlüsseln und mehr.

CloudWatch

Verwenden Sie CloudWatch, um den Zustand Ihres Clusters in Echtzeit zu überwachen.

Themen

Arbeiten mit Client-SDK-Protokollen

Sie können vom Client-SDK generierte Protokolle abrufen. AWS CloudHSM bietet eine Implementierung der Protokollierung mit Client-SDK 3 und Client-SDK 5.

Themen

Protokollierung im Client-SDK 5

Die Protokolle des Client-SDK 5 enthalten Informationen für jede Komponente in einer nach der Komponente benannten Datei. Sie können das Configure-Tool für Client-SDK 5 verwenden, um die Protokollierung für jede Komponente zu konfigurieren.

Wenn Sie keinen Speicherort für die Datei angeben, schreibt das System Protokolle an den Standardspeicherort:

PKCS #11 library

- Linux

  ```
  /opt/cloudhsm/run/cloudhsm-pkcs11.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
  ```

OpenSSL Dynamic Engine

- Linux

  ```
  stderr
  ```

JCE provider

- Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
```

Informationen zur Konfiguration der Protokollierung für das Client-SDK 5 finden Sie im Client-SDK-5-Configure-Tool

Protokollierung im Client-SDK 3

Client-SDK-3-Protokolle enthalten detaillierte Informationen vom AWS CloudHSM-Client-Daemon. Der Speicherort der Protokolle hängt vom Betriebssystem der Amazon EC2-Client-Instance ab, auf der Sie den Client-Daemon ausführen.

Amazon Linux

In Amazon Linux werden die AWS CloudHSM-Client-Protokolle in die Datei mit dem Namen `/opt/cloudhsm/run/cloudhsm_client.log` geschrieben. Sie können logrotate oder ein ähnliches Tool verwenden, um diese Protokolle zu rotieren und zu verwalten.

Amazon Linux 2

In Amazon Linux 2 werden die AWS CloudHSM-Client-Protokolle gesammelt und im Journal gespeichert. Sie können journalctl verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie z. B. den folgenden Befehl, um die AWS CloudHSM-Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

CentOS 7

In CentOS 7 werden die AWS CloudHSM-Client-Protokolle gesammelt und im Journal gespeichert. Sie können journalctl verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie z. B. den folgenden Befehl, um die AWS CloudHSM-Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

CentOS 8

In CentOS 8 werden die AWS CloudHSM-Client-Protokolle gesammelt und im Journal gespeichert. Sie können journalctl verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie z. B. den folgenden Befehl, um die AWS CloudHSM-Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

RHEL 7

In Red Hat Enterprise Linux 7 werden die AWS CloudHSM-Client-Protokolle gesammelt und im Journal gespeichert. Sie können journalctl verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie z. B. den folgenden Befehl, um die AWS CloudHSM-Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

RHEL 8

In Red Hat Enterprise Linux 8 werden die AWS CloudHSM-Client-Protokolle gesammelt und im Journal gespeichert. Sie können journalctl verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie z. B. den folgenden Befehl, um die AWS CloudHSM-Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

In Ubuntu 16.04 werden die AWS CloudHSM-Client-Protokolle gesammelt und im Journal gespeichert. Sie können journalctl verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie z. B. den folgenden Befehl, um die AWS CloudHSM-Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

In Ubuntu 18.04 werden die AWS CloudHSM-Client-Protokolle gesammelt und im Journal gespeichert. Sie können journalctl verwenden, um diese Protokolle anzuzeigen und zu verwalten. Verwenden Sie z. B. den folgenden Befehl, um die AWS CloudHSM-Client-Protokolle anzuzeigen.

```
journalctl -f -u cloudhsm-client
```

Windows

- Für Windows-Client 1.1.2 und höher:

AWS CloudHSM Client-Protokolle werden in eine `cloudhsm.log`-Datei im Ordner AWS CloudHSM-Programmdateien (`C:\Program Files\Amazon\CloudHSM\`) geschrieben. Dem Namen einer Protokolldatei wird ein Zeitstempel angehängt, der angibt, wann der AWS CloudHSM-Client gestartet wurde.

- Für Windows-Client 1.1.1 und früher:

  Die Client-Protokolle werden nicht in eine Datei geschrieben. Die Protokolle werden in der Eingabeaufforderung oder im PowerShell-Fenster angezeigt, in der bzw. dem Sie den AWS CloudHSM-Client gestartet haben.

Arbeiten mit AWS CloudTrail und AWS CloudHSM

AWS CloudHSM ist in AWS CloudTrail integriert, einen Service, der die Aktionen eines Benutzers, einer Rolle oder eines AWS-Service in AWS CloudHSM protokolliert. CloudTrail erfasst alle API-Aufrufe für AWS CloudHSM als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der AWS CloudHSM-Konsole und Code-Aufrufe der AWS CloudHSM-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail-Ereignissen an einen Amazon S3-Bucket, einschließlich Ereignisse für AWS CloudHSM aktivieren. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse in der CloudTrail-Konsole trotzdem in Event history (Ereignisverlauf) anzeigen. Mit den von CloudTrail erfassten Informationen können Sie die an AWS CloudHSM gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen zu CloudTrail finden Sie im AWS CloudTrail-Benutzerhandbuch. Eine vollständige Liste der AWS CloudHSM-API-Operationen finden Sie unter Aktionen in der AWS CloudHSM-API-Referenz.

AWS CloudHSM-Informationen in CloudTrail

CloudTrail wird beim Erstellen Ihres AWS-Kontos für Sie aktiviert. Die in AWS CloudHSM auftretenden Aktivitäten werden als CloudTrail-Ereignis zusammen mit anderen AWS-Serviceereignissen in Ereignisverlauf aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS-Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail-Ereignisverlauf.

Zur kontinuierlichen Aufzeichnung von Ereignissen in Ihrem AWS-Konto, einschließlich Ereignissen für AWS CloudHSM, erstellen Sie einen Trail. Ein Trail ermöglicht es CloudTrail, Protokolldateien

in einem Amazon-S3-Bucket bereitzustellen. Wenn Sie einen Pfad in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon-S3-Bucket bereit. Darüber hinaus können Sie andere AWS-Services konfigurieren, um die in den CloudTrail-Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie unter:

- Übersicht zum Erstellen eines Trails
- Von CloudTrail unterstützte Dienste und Integrationen
- Konfigurieren von Amazon-SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail-Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail-Protokolldateien aus mehreren Konten

CloudTrail protokolliert alle AWS CloudHSM-Vorgänge, einschließlich schreibgeschützter Vorgänge wie `DescribeClusters` und `ListTags`, und Verwaltungsvorgänge wie `InitializeCluster`, `CreatHsm` und `DeleteBackup`.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Anhand der Identitätsinformationen zur Benutzeridentität können Sie Folgendes bestimmen:

- Ob die Anfrage mit Stammbenutzer- oder AWS Identity and Access Management (IAM)-Benutzeranmeldeinformationen ausgeführt wurde.
- Ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer ausgeführt wurde.
- Gibt an, ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter dem CloudTrail userIdentity-Element.

Grundlagen zu AWS CloudHSM-Protokolldateieinträgen

Ein Trail ist eine Konfiguration, durch die Ereignisse als Protokolldateien an den von Ihnen angegebenen Amazon-S3-Bucket übermittelt werden. CloudTrail-Protokolldateien können einen oder mehrere Einträge enthalten. Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält unter anderem Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion sowie über die Anfrageparameter. CloudTrail-Protokolleinträge sind kein geordnetes Stack-Trace der öffentlichen API-Aufrufe und erscheinen daher in keiner bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen CloudTrail-Protokolleintrag, der die Aktion AWS CloudHSM `CreateHsm` demonstriert.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAJZVM5NEGZSTCITAMM:ExampleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/AdminRole/ExampleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIY22AX6VRYNBGJSA",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2017-07-11T03:48:44Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAJZVM5NEGZSTCITAMM",
                "arn": "arn:aws:iam::111122223333:role/AdminRole",
                "accountId": "111122223333",
                "userName": "AdminRole"
            }
        }
    },
    "eventTime": "2017-07-11T03:50:45Z",
    "eventSource": "cloudhsm.amazonaws.com",
    "eventName": "CreateHsm",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.179",
    "userAgent": "aws-internal/3",
    "requestParameters": {
        "availabilityZone": "us-west-2b",
        "clusterId": "cluster-fw7mh6mayb5"
    },
    "responseElements": {
        "hsm": {
            "eniId": "eni-65338b5a",
            "clusterId": "cluster-fw7mh6mayb5",
            "state": "CREATE_IN_PROGRESS",
            "eniIp": "10.0.2.7",
            "hsmId": "hsm-6lz2hfmnzbx",
            "subnetId": "subnet-02c28c4b",
```

```
            "availabilityZone": "us-west-2b"
        }
    },
    "requestID": "1dae0370-65ec-11e7-a770-6578d63de907",
    "eventID": "b73a5617-8508-4c3d-900d-aa8ac9b31d08",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Arbeiten mit Amazon CloudWatch Logs und AWS CloudHSM Audit Logs

Wenn ein HSM in Ihrem Konto einen Befehl von den AWS CloudHSM Befehlszeilentools oder Softwarebibliotheken empfängt, zeichnet es die Ausführung des Befehls in Form eines Auditprotokolls auf. Die HSM-Audit-Protokolle enthalten alle Client-initiierten Verwaltungsbefehle, einschließlich derjenigen, die das HSM erstellen und löschen, die in das HSM ein- und ausloggen sowie die Benutzer und Schlüssel verwalten. Diese Protokolle bieten eine zuverlässige Aufzeichnung von Aktionen, die den Zustand des HSM verändert haben.

AWS CloudHSM sammelt Ihre HSM-Prüfprotokolle und sendet sie in Ihrem Namen an Amazon CloudWatch Logs. Sie können die Funktionen von CloudWatch Logs verwenden, um Ihre AWS CloudHSM Audit-Logs zu verwalten, einschließlich der Suche und Filterung der Logs und des Exports von Protokolldaten nach Amazon S3. Sie können mit Ihren HSM-Prüfprotokollen in der CloudWatch Amazon-Konsole arbeiten oder die CloudWatch Logs-Befehle in der CLI und den CloudWatch Logs-SDKs verwenden.

Themen

- So funktioniert die HSM-Auditprotokollierung
- HSM-Prüfprotokolle in CloudWatch Logs anzeigen
- Interpretieren von HSM-Audit-Prüfprotokollen
- HSM-Auditprotokoll-Referenz

So funktioniert die HSM-Auditprotokollierung

Die Audit-Protokollierung ist in allen AWS CloudHSM-Clustern automatisch aktiviert. Sie kann nicht deaktiviert oder ausgeschaltet werden, und es gibt keine Einstellungen, die den Export der Protokolle zu CloudWatch Logs durch AWS CloudHSM verhindern. Jedes Protokollereignis verfügt über eine

Sequenznummer, die Zeitstempel und die Reihenfolge der Ereignisse angibt und Ihnen hilft, etwaige Manipulationsversuche zu erkennen.

Jede HSM-Instance generiert ihre eigenen Protokolle. Die Audit-Protokolle von verschiedenen HSMs, auch von jenen im selben Cluster, werden sich wahrscheinlich unterscheiden. Beispiel: nur das erste HSM in jedem Cluster zeichnet die Initialisierung des HSM auf. Initialisierungsereignisse erscheinen nicht in den Protokollen von HSMs, die von Sicherungen geklont werden. Ebenso zeichnet das HSM, das die Schlüssel generiert, beim Anlegen eines Schlüssels ein Ereignis zur Schlüsselerzeugung auf. Die anderen HSMs im Cluster zeichnen ein Ereignis auf, wenn sie den Schlüssel über die Synchronisation erhalten.

AWS CloudHSM sammelt die Protokolle und sendet sie an CloudWatch Logs in Ihrem Konto. Um in Ihrem Namen mit dem CloudWatch-Logs-Dienst zu kommunizieren, verwendet AWS CloudHSM eine mit dem Dienst verknüpfte Rolle. Die IAM-Richtlinie, die der Rolle zugeordnet ist, erlaubt es AWS CloudHSM nur die erforderlichen Aufgaben zum Senden der Audit-Protokolle an CloudWatch Logs auszuführen.

> **Important**
>
> Wenn Sie einen Cluster vor dem 20. Januar 2018 angelegt haben und noch keine angehängte, servicegebundene Rolle angelegt haben, müssen Sie diese manuell erstellen. Dies ist für CloudWatch erforderlich, um Audit-Protokolle von Ihrem AWS CloudHSM-Cluster empfangen zu können. Weitere Informationen über die Erstellung von dienstverknüpften Rollen finden Sie unter Grundlegendes zu dienstverknüpften Rollen sowie unter Erstellen einer dienstverknüpften Rolle im IAM-Benutzerhandbuch.

HSM-Prüfprotokolle in CloudWatch Logs anzeigen

Amazon CloudWatch Logs organisiert die Audit-Logs in Protokollgruppen und innerhalb einer Protokollgruppe in Log-Streams. Jeder Protokolleintrag ist ein Ereignis. AWS CloudHSM erstellt eine Protokollgruppe für jeden Cluster und einen Protokollstream für jedes HSM im Cluster. Sie müssen keine CloudWatch Protokollkomponenten erstellen oder Einstellungen ändern.

- Der Name der Protokollgruppe ist `/aws/cloudhsm/<cluster ID>`; zum Beispiel `/aws/cloudhsm/cluster-likphkxygsn`. Wenn Sie den Namen der Protokollgruppe in einer CLI oder einem PowerShell Befehl verwenden, achten Sie darauf, ihn in doppelte Anführungszeichen zu setzen.

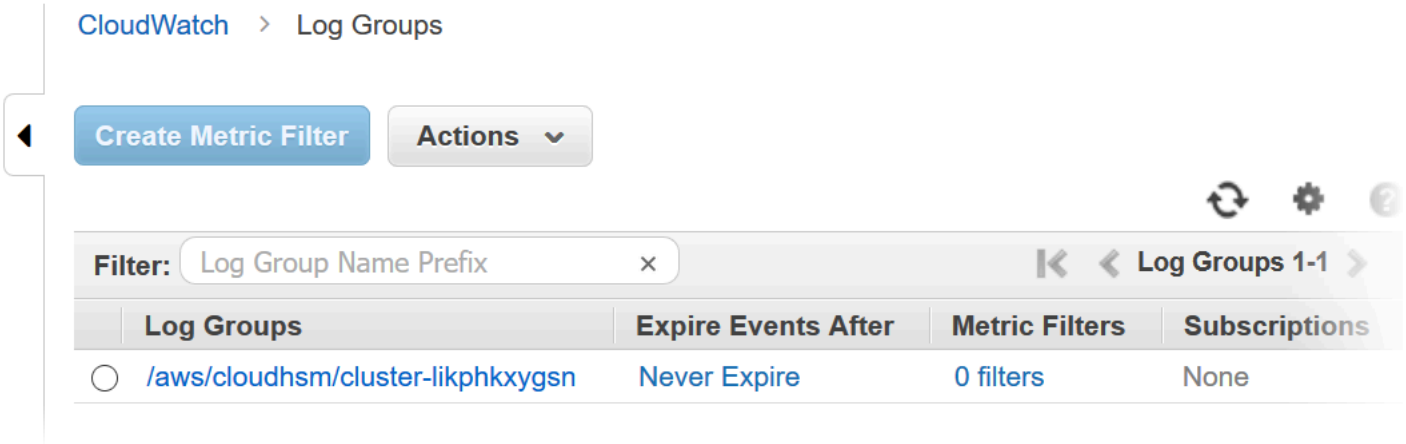
- Der Name des Protokollstreams ist die HSM-ID; zum Beispiel `hsm-nwbbiqbj4jk`.

  Im Allgemeinen gibt es für jedes HSM einen Protokollstream. Jede Aktion, die die HSM-ID ändert, z.B. wenn ein HSM ausfällt und ersetzt wird, erzeugt jedoch einen neuen Protokollstream.

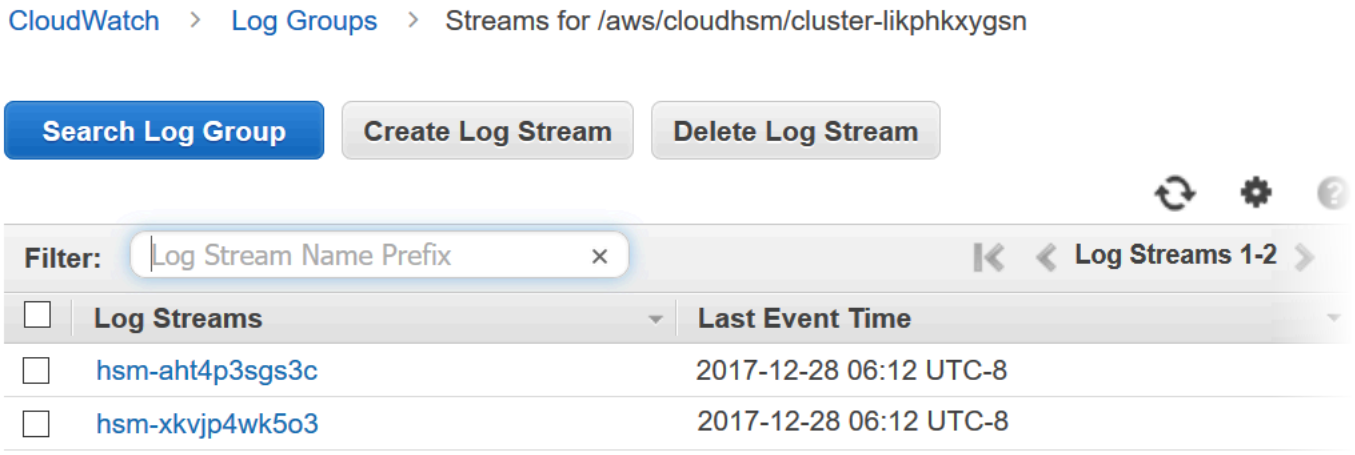
Weitere Informationen zu CloudWatch Logs-Konzepten finden Sie unter Konzepte im Amazon CloudWatch Logs-Benutzerhandbuch.

Sie können die Überwachungsprotokolle für ein HSM auf der Seite CloudWatch Logs in der AWS Management Console, den Logs-Befehlen in der CLI, den CloudWatch CloudWatch PowerShell Logs-Cmdletsoder den CloudWatch Logs-SDKs anzeigen. Anweisungen finden Sie unter Protokolldaten anzeigen im Amazon CloudWatch Logs-Benutzerhandbuch.

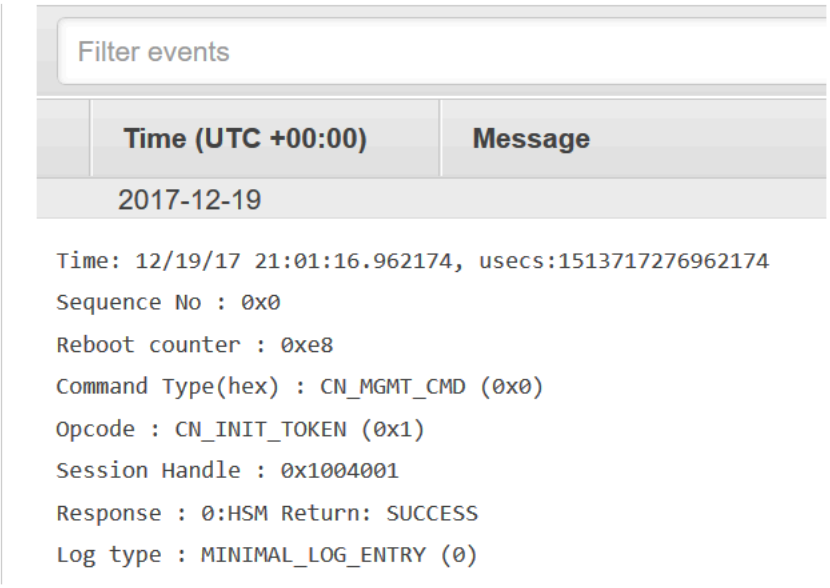
In der folgenden Abbildung wird beispielsweise die Protokollgruppe für den `cluster-likphkxygsn`-Cluster in der AWS Management Console angezeigt.



Wenn Sie den Namen der Cluster-Protokollgruppe wählen, können Sie den Protokollstream für jeden der HSMs im Cluster anzeigen. Das folgende Bild zeigt die Protokollströme für die HSMs im Cluster `cluster-likphkxygsn`.

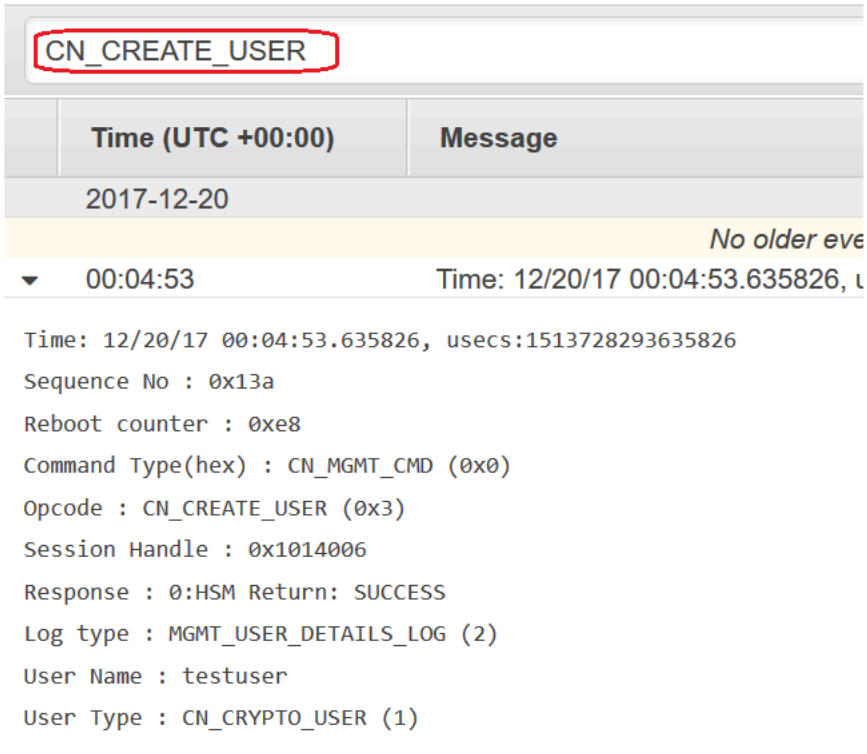


Wenn Sie einen HSM-Protokollstreamnamen auswählen, können Sie die Ereignisse im Auditprotokoll anzeigen. So ist beispielsweise dieses Ereignis, das eine Sequenznummer von 0x0 und einen `Opcode` von `CN_INIT_TOKEN` hat, typischerweise das erste Ereignis für das erste HSM in jedem Cluster. Es zeichnet die Initialisierung des HSM im Cluster auf.



Sie können all die vielen Funktionen in CloudWatch Logs verwenden, um Ihre Audit-Logs zu verwalten. Beispiel: Sie können die Funktion Ereignisse filtern nutzen, um einen speziellen Text in einem Ereignis zu suchen, wie `CN_CREATE_USER Opcode`.

Um alle Ereignisse zu finden, die den angegebenen Text nicht enthalten, fügen Sie ein Minuszeichen (-) vor dem Text ein. Wenn Sie beispielsweise Ereignisse suchen möchten, die `CN_CREATE_USER` nicht enthalten, geben Sie -CN_CREATE_USER ein.



Interpretieren von HSM-Audit-Prüfprotokollen

Die Ereignisse in den HSM-Audit-Protokollen haben standardisierte Felder. Einige Ereignistypen umfassen weitere Felder, die nützliche Informationen über das Ereignis aufzeichnen. Beispiel: Benutzeranmeldungs- und Benutzerverwaltungsereignisse enthalten den Benutzernamen und Typ des Benutzers. Key-Management-Befehle enthalten das Schlüssel-Handle.

Einige der Felder liefern besonders wichtige Informationen. `Opcode` identifiziert den Verwaltungsbefehl, der aufgezeichnet wird. `Sequence No` identifiziert ein Ereignis im Protokollstream und gibt die Reihenfolge an, in der es aufgenommen wurde.

Beispiel: Das folgende Beispielereignis ist das zweite Ereignis (`Sequence No: 0x1`) im Protokollstream für ein HSM. Es zeigt, wie das HSM einen Passwort-Verschlüsselungsschlüssel erzeugt, der Teil seiner Startroutine ist.

```
Time: 12/19/17 21:01:17.140812, usecs:1513717277140812
Sequence No : 0x1
Reboot counter : 0xe8
```

```
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GEN_PSWD_ENC_KEY (0x1d)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

Die folgenden Felder gelten für jedes AWS CloudHSM-Ereignis im Überwachungsprotokoll.

Zeit

Die Zeit, zu der das Ereignis in der UTC-Zeitzone aufgetreten ist. Die Zeit wird als menschenlesbare Zeit und Unix-Zeit in Mikrosekunden angezeigt.

Neustart des Zählers

Ein 32-Bit persistenter Ordinalzähler, der beim Neustart der HSM-Hardware erhöht wird.

Alle Ereignisse in einem Protokollstream haben den gleichen Neustartzählerwert. Der Neustartzähler ist jedoch möglicherweise nicht eindeutig für einen Protokollstream, da er sich zwischen verschiedenen HSM-Instances im selben Cluster unterscheiden kann.

Sequenz Nr.

Ein 64-Bit-Ordinalzähler, der bei jedem Protokollereignis erhöht wird. Das erste Ereignis in jedem Protokollstream hat eine Sequenznummer von 0x0. Es dürfen keine Lücken in den `Sequence No`-Werten vorhanden sein. Die Sequenznummer ist nur in einem Protokollstream eindeutig.

Befehlstyp

Ein hexadezimaler Wert, der die Kategorie des Befehls repräsentiert. Befehle im AWS CloudHSM-Protokollstream haben einen Befehlstyp `CN_MGMT_CMD` (0x0) oder `CN_CERT_AUTH_CMD` (0x9).

Opcode

Identifiziert den ausgeführten Verwaltungsbefehl. Eine Liste der `Opcode`-Werte in den AWS CloudHSM-Audit-Protokollen finden Sie unter HSM-Auditprotokoll-Referenz.

Session-Handle

Identifiziert die Sitzung, in der der Befehl ausgeführt und das Ereignis protokolliert wurde.

Antwort

Zeichnet die Antwort auf den Verwaltungsbefehl auf. Sie können das `Response`-Feld nach den Werten `SUCCESS` und `ERROR` durchsuchen.

Protokolltyp

Gibt den Protokolltyp des AWS CloudHSM-Protokolls an, das den Befehl aufgezeichnet hat.

- `MINIMAL_LOG_ENTRY (0)`
- `MGMT_KEY_DETAILS_LOG (1)`
- `MGMT_USER_DETAILS_LOG (2)`
- `GENERIC_LOG`

Beispiele für Audit-Protokollereignisse

Die Ereignisse in einem Protokollstream zeichnen die Historie des HSM von der Erstellung bis zum Löschen auf. Mit dem Protokoll können Sie den Lebenszyklus Ihrer HSMs überprüfen und sich einen Überblick über deren Betrieb verschaffen. Wenn Sie die Ereignisse interpretieren, beachten Sie den `Opcode`, der den Befehl oder die Aktion zur Verwaltung anzeigt, und die `Sequence No`, die die Reihenfolge der Ereignisse angibt.

Themen

Beispiel: Initialisieren des ersten HSM in einem Cluster

Der Audit-Protokollstream für das erste HSM in jedem Cluster unterscheidet sich erheblich von den Protokollströmen anderer HSMs im Cluster. Das Auditprotokoll für das erste HSM in jedem Cluster zeichnet seine Erstellung und Initialisierung auf. Die Protokolle weiterer HSMs im Cluster, die aus Backups generiert werden, beginnen mit einem Anmeldeereignis.

> **Important**
>
> Die folgenden Initialisierungseinträge erscheinen nicht in den CloudWatch-Protokollen der Cluster, die vor der Veröffentlichung der CloudHSM-Auditprotokollierung (30. August 2018) initialisiert wurden. Weitere Informationen finden Sie unter Dokumentenhistorie.

Die folgenden Beispielereignisse erscheinen im Protokollstream für den ersten HSM in einem Cluster. Das erste Ereignis im Protokoll – das mit `Sequence No 0x0` – entspricht dem Befehl für die Initialisierung des HSM (`CN_INIT_TOKEN`). Die Antwort (`Response : 0: HSM Return: SUCCESS`) zeigt an, dass der Befehl erfolgreich war.

```
Time: 12/19/17 21:01:16.962174, usecs:1513717276962174
Sequence No : 0x0
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INIT_TOKEN (0x1)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

Das zweite Ereignis in diesem exemplarischen Protokollstream (`Sequence No 0x1`) zeichnet den Befehl zum Erstellen des Kennwortverschlüsselungsschlüssels auf, den das HSM verwendet (`CN_GEN_PSWD_ENC_KEY`).

Dies ist eine typische Startsequenz für das erste HSM in jedem Cluster. Da nachfolgende HSMs im gleichen Cluster Klone des ersten sind, verwenden sie den gleichen Passwort-Verschlüsselungsschlüssel.

```
Time: 12/19/17 21:01:17.140812, usecs:1513717277140812
Sequence No : 0x1
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GEN_PSWD_ENC_KEY (0x1d)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

Das dritte Ereignis in diesem beispielhaften Protokollstream (`Sequence No 0x2`) ist die Erstellung des Benutzers Appliance-Benutzer (AU). Dabei handelt es sich um den AWS CloudHSM-

Service. Ereignisse, an denen HSM-Benutzer beteiligt sind, enthalten zusätzliche Felder für den Benutzernamen und den Benutzertyp.

```
Time: 12/19/17 21:01:17.174902, usecs:1513717277174902
Sequence No : 0x2
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_APPLIANCE_USER (0xfc)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : app_user
User Type : CN_APPLIANCE_USER (5)
```

Das vierte Ereignis in diesem Beispiel eines Protokollstreams (`Sequence No 0x3`) zeichnet das `CN_INIT_DONE`-Ereignis, das die Initialisierung des HSM abschließt.

```
Time: 12/19/17 21:01:17.298914, usecs:1513717277298914
Sequence No : 0x3
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INIT_DONE (0x95)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

Sie können den verbleibenden Ereignissen in der Startsequenz folgen. Diese Ereignisse können mehrere An- und Abmeldeereignisse und die Generierung des Schlüsselverschlüsselungscodes (Key Encryption Key, KEK) umfassen. Das folgende Ereignis zeichnet den Befehl auf, der das Passwort des Precrypto Officer (PRECO) ändert. Dieser Befehl aktiviert den Cluster.

```
Time: 12/13/17 23:04:33.846554, usecs:1513206273846554
Sequence No: 0x1d
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_CHANGE_PSWD (0x9)
Session Handle: 0x2010003
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: admin
User Type: CN_CRYPTO_PRE_OFFICER (6)
```

An- und Abmeldeereignisse

Beachten Sie die Ereignisse bei der Interpretation Ihres Audit-Protokolls, die das Ein- und Ausloggen von Benutzern in das HSM repräsentieren. Diese Ereignisse helfen Ihnen zu bestimmen, welcher Benutzer für die Verwaltungsbefehle verantwortlich ist, die in der Reihenfolge zwischen dem Anmelde- und dem Abmeldebefehl erscheinen.

Dieser Protokolleintrag zeichnet beispielsweise eine Anmeldung durch einen Verschlüsselungsverantwortlichen mit dem Namen `admin` auf. Die Sequenznummer, `0x0`, gibt an, dass dies das erste Ereignis in diesem Protokollstream ist.

Wenn sich ein Benutzer bei einem HSM anmeldet, zeichnen die anderen HSMs im Cluster auch ein Anmeldeereignis für den Benutzer auf. Die entsprechenden Anmeldeereignisse finden Sie in den Protokollströmen anderer HSMs im Cluster, kurz nach dem ersten Anmeldeereignis.

```
Time: 01/16/18 01:48:49.824999, usecs:1516067329824999
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPTO_OFFICER (2)
```

Das folgende Beispielereignis zeichnet den `admin`-Verschlüsselungsverantwortlichen auf, der sich abmeldet. Die Sequenznummer, `0x2`, gibt an, dass dies das dritte Ereignis im Protokollstream ist.

Wenn der angemeldete Benutzer die Sitzung schließt, ohne sich abzumelden, enthält der Protokollstream ein `CN_APP_FINALIZE`-Ereignis oder ein Ereignis für das Schließen der Sitzung (`CN_SESSION_CLOSE`), anstelle eines `CN_LOGOUT`-Ereignisses. Im Gegensatz zum Anmeldeereignis wird dieses Abmeldeereignis typischerweise nur von jenem HSM aufgezeichnet, das den Befehl ausführt.

```
Time: 01/16/18 01:49:55.993404, usecs:1516067395993404
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGOUT (0xe)
```

```
Session Handle : 0x7014000
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPTO_OFFICER (2)
```

Wenn ein Anmeldeversuch fehlschlägt, weil der Benutzername ungültig ist, zeichnet das HSM ein `CN_LOGIN`-Ereignis mit dem Benutzernamen und -typ im Login-Befehl auf. Die Antwort zeigt die Fehlermeldung 157, die erläutert, dass der Benutzername nicht vorhanden ist.

```
Time: 01/24/18 17:41:39.037255, usecs:1516815699037255
Sequence No : 0x4
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 157:HSM Error: user isn't initialized or user with this name doesn't exist
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : ExampleUser
User Type : CN_CRYPTO_USER (1)
```

Wenn ein Anmeldeversuch fehlschlägt, weil das Passwort ungültig ist, zeichnet das HSM ein `CN_LOGIN`-Ereignis mit dem Benutzernamen und -typ im Login-Befehl auf. Die Antwort zeigt die Fehlermeldung mit dem `RET_USER_LOGIN_FAILURE`-Fehlercode.

```
Time: 01/24/18 17:44:25.013218, usecs:1516815865013218
Sequence No : 0x5
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 163:HSM Error: RET_USER_LOGIN_FAILURE
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

Beispiel: Erstellen und Löschen von Benutzern

Dieses Beispiel zeigt die Protokollereignisse, die aufgezeichnet werden, wenn ein Verschlüsselungsverantwortlicher (CO) Benutzer anlegt und löscht.

Das erste Ereignis zeichnet einen CO, `admin`, bei der Anmeldung im HSM auf. Die Sequenznummer, `0x0`, gibt an, dass dies das erste Ereignis im Protokollstream ist. Der Name und Typ des Benutzers, der sich anmeldet, sind im Ereignis enthalten.

```
Time: 01/16/18 01:48:49.824999, usecs:1516067329824999
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPTO_OFFICER (2)
```

Das nächste Ereignis im Protokollstream (Sequenz `0x1`) zeichnet die Erstellung eines neuen Verschlüsselungsbenutzers (CU, Crypto-User) durch den CO auf. Der Name und Typ des neuen Benutzers sind im Ereignis enthalten.

```
Time: 01/16/18 01:49:39.437708, usecs:1516067379437708
Sequence No : 0x1
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_USER (0x3)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : bob
User Type : CN_CRYPTO_USER (1)
```

Dann erstellt der CO einen anderen Verschlüsselungsverantwortlichen, `alice`. Die Sequenznummer zeigt an, dass diese Aktion der vorherigen ohne dazwischenliegende Aktionen folgte.

```
Time: 01/16/18 01:49:55.993404, usecs:1516067395993404
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_CO (0x4)
Session Handle : 0x7014007
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : alice
```

```
User Type : CN_CRYPTO_OFFICER (2)
```

Später meldet sich der CO mit Namen `admin` an und löscht den Verschlüsselungsverantwortlichen mit Namen `alice`. Das HSM zeichnet ein `CN_DELETE_USER`-Ereignis auf. Der Name und Typ des gelöschten Benutzers sind im Ereignis enthalten.

```
Time: 01/23/18 19:58:23.451420, usecs:1516737503451420
Sequence No : 0xb
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_DELETE_USER (0xa1)
Session Handle : 0x7014007
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : alice
User Type : CN_CRYPTO_OFFICER (2)
```

Beispiel: Erstellen und Löschen eines Schlüsselpaares

Dieses Beispiel zeigt die Ereignisse, die in einem HSM-Auditprotokoll aufgezeichnet werden, wenn Sie ein Schlüsselpaar erstellen und löschen.

Das folgende Ereignis zeichnet die Anmeldung des Verschlüsselungsbenutzers (CU, Crypto User) mit Namen `crypto_user` am HSM auf.

```
Time: 12/13/17 23:09:04.648952, usecs:1513206544648952
Sequence No: 0x28
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_LOGIN (0xd)
Session Handle: 0x2014005
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: crypto_user
User Type: CN_CRYPTO_USER (1)
```

Anschließend erzeugt der CU ein Schlüsselpaar (`CN_GENERATE_KEY_PAIR`). Das Schlüssel-Handle des privaten Schlüssels ist `131079`. Das Schlüssel-Handle des öffentlichen Schlüssels ist `131078`.

```
Time: 12/13/17 23:09:04.761594, usecs:1513206544761594
Sequence No: 0x29
Reboot counter: 0xe8
```

```
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_GENERATE_KEY_PAIR (0x19)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131079
Public Key Handle: 131078
```

Der CU löscht das Schlüsselpaar sofort wieder. Ein CN_DESTROY_OBJECT-Ereignis zeichnet die Löschung des öffentlichen Schlüssels (131078) auf.

```
Time: 12/13/17 23:09:04.813977, usecs:1513206544813977
Sequence No: 0x2a
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_DESTROY_OBJECT (0x11)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131078
Public Key Handle: 0
```

Anschließend zeichnet ein zweites CN_DESTROY_OBJECT-Ereignis die Löschung des privaten Schlüssels (131079) auf.

```
Time: 12/13/17 23:09:04.815530, usecs:1513206544815530
Sequence No: 0x2b
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_DESTROY_OBJECT (0x11)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131079
Public Key Handle: 0
```

Und schließlich meldet sich der CU ab.

```
Time: 12/13/17 23:09:04.817222, usecs:1513206544817222
Sequence No: 0x2c
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
```

```
Opcode: CN_LOGOUT (0xe)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: crypto_user
User Type: CN_CRYPTO_USER (1)
```

Beispiel: Generieren und Synchronisieren eines Schlüssels

Dieses Beispiel zeigt den Effekt der Schlüsselerstellung in einem Cluster mit mehreren HSMs. Der Schlüssel wird auf einem HSM erzeugt, aus dem HSM als maskiertes Objekt extrahiert und in den anderen HSMs als maskiertes Objekt eingefügt.

> **Note**
>
> Die Client-Tools können den Schlüssel möglicherweise nicht synchronisieren. Der Befehl kann außerdem den Parameter min_srv umfassen, der den Schlüssel nur mit der angegebenen Anzahl von HSMs synchronisiert. In beiden Fällen synchronisiert der AWS CloudHSM-Service den Schlüssel mit den anderen HSMs im Cluster. Da die HSMs nur Client-seitige Verwaltungsbefehle in ihren Protokollen aufzeichnen, wird die serverseitige Synchronisation nicht im HSM-Protokoll aufgezeichnet.

Betrachten Sie zunächst den Protokollstream des HSM, das die Befehle empfängt und ausführt. Der Protokollstream ist nach der HSM-ID, `hsm-abcde123456`, benannt, aber die HSM-ID erscheint nicht in den Protokollereignissen.

Zunächst meldet sich der `testuser`-Verschlüsselungsbenutzer (CU, Crypto User) am `hsm-abcde123456`-HSM an.

```
Time: 01/24/18 00:39:23.172777, usecs:1516754363172777
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

Der CU führt den Befehl exSymKey aus, um einen symmetrischen Schlüssel zu generieren. Das hsm-abcde123456-HSM generiert einen symmetrischen Schlüssel mit dem Schlüssel-Handle 262152. Das HSM vermerkt ein CN_GENERATE_KEY-Ereignis im Protokoll.

```
Time: 01/24/18 00:39:30.328334, usecs:1516754370328334
Sequence No : 0x1
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GENERATE_KEY (0x17)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

Das nächste Ereignis im Protokollstream für hsm-abcde123456 zeichnet den ersten Schritt im Schlüssel-Synchronisationsprozess auf. Der neue Schlüssel (Schlüssel-Handle 262152) wird aus dem HSM als maskiertes Objekt extrahiert.

```
Time: 01/24/18 00:39:30.330956, usecs:1516754370330956
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_EXTRACT_MASKED_OBJECT_USER (0xf0)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

Betrachten Sie jetzt den Protokollstream für das HSM hsm-zyxwv987654, ein anderes HSM im selben Cluster. Dieser Protokollstream enthält auch ein Anmelde-Ereignis für den CU testuser. Der Zeitwert zeigt an, dass es kurz nach der Anmeldung des Benutzers am HSM hsm-abcde123456 auftritt.

```
Time: 01/24/18 00:39:23.199740, usecs:1516754363199740
Sequence No : 0xd
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
```

```
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

Dieser Protokollstream für diese HSM verfügt nicht über ein CN_GENERATE_KEY-Ereignis. Aber es hat ein Ereignis, das die Synchronisation des Schlüssels zu diesem HSM aufzeichnet. Das CN_INSERT_MASKED_OBJECT_USER-Ereignis zeichnet den Empfang des Schlüssels 262152 als maskiertes Objekt auf. Jetzt existiert der Schlüssel 262152 auf beiden HSMs im Cluster.

```
Time: 01/24/18 00:39:30.408950, usecs:1516754370408950
Sequence No : 0xe
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INSERT_MASKED_OBJECT_USER (0xf1)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

Wenn der CU-Benutzer sich abmeldet, wird dieses CN_LOGOUT-Ereignis nur in den Protokollstream jenes HSM aufgenommen, das die Befehle erhalten hat.

Beispiel: Exportieren eines Schlüssels

Dieses Beispiel zeigt die Audit-Protokoll-Ereignisse, die aufgezeichnet werden, wenn ein Verschlüsselungsbenutzer (CU, Crypto User) Schlüssel aus einem Cluster mit mehreren HSMs exportiert.

Das folgende Ereignis zeichnet die CU (testuser)-Anmeldung bei key_mgmt_util auf.

```
Time: 01/24/18 19:42:22.695884, usecs:1516822942695884
Sequence No : 0x26
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

Der CU-Benutzer verwendet den Befehl exSymKey, um den Schlüssel 7, ein 256-Bit-AES-Schlüssel, zu exportieren. Der Befehl verwendet den Schlüssel 6, ein 256-Bit-AES-Schlüssel in den HSMs, als Verpackungsschlüssel.

Das HSM, das den Befehl empfängt, vermerkt ein CN_WRAP_KEY-Ereignis für den Schlüssel 7, den Schlüssel, der exportiert wird.

```
Time: 01/24/18 19:51:12.860123, usecs:1516823472860123
Sequence No : 0x27
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_WRAP_KEY (0x1a)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 7
Public Key Handle : 0
```

Anschließend verzeichnet das HSM ein CN_NIST_AES_WRAP-Ereignis für den Verpackungsschlüssel 6. Der Schlüssel wird verpackt und anschließend sofort wieder entpackt, aber der HSM zeichnet nur ein Ereignis auf.

```
Time: 01/24/18 19:51:12.905257, usecs:1516823472905257
Sequence No : 0x28
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_NIST_AES_WRAP (0x1e)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 6
Public Key Handle : 0
```

Der exSymKey-Befehl schreibt den exportierten Schlüssel in eine Datei, ändert aber nicht den Schlüssel auf dem HSM. Daher gibt es keine entsprechenden Ereignisse in den Protokollen der anderen HSMs im Cluster.

Beispiel: Importieren eines Schlüssels

Dieses Beispiel zeigt die Audit-Protokoll-Ereignisse, die aufgezeichnet werden, wenn Sie Schlüssel in die HSMs in einem Cluster importieren. In diesem Beispiel verwendet der Verschlüsselungsbenutzer

(CU, Crypto User) den Befehl imSymKey zum Importieren eines AES-Schlüssels in die HSMs. Der Befehl verwendet Schlüssel 6 als Verpackungsschlüssel.

Das HSM, das den Befehl empfängt, vermerkt ein CN_NIST_AES_WRAP-Ereignis für den Schlüssel 6, den Verpackungsschlüssel.

```
Time: 01/24/18 19:58:23.170518, usecs:1516823903170518
Sequence No : 0x29
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_NIST_AES_WRAP (0x1e)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 6
Public Key Handle : 0
```

Anschließend verzeichnet das HSM ein CN_UNWRAP_KEY-Ereignis für den Importvorgang. Der importierte Schlüssel ist dem Schlüssel-Handle 11 zugeordnet.

```
Time: 01/24/18 19:58:23.200711, usecs:1516823903200711
Sequence No : 0x2a
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_UNWRAP_KEY (0x1b)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

Wenn ein neuer Schlüssel generiert oder importiert wird, versuchen die Client-Tools automatisch, den neuen Schlüssel mit anderen HSMs im Cluster zu synchronisieren. In diesem Fall zeichnet das HSM ein CN_EXTRACT_MASKED_OBJECT_USER-Ereignis auf, wenn der Schlüssel 11 aus dem HSM als maskiertes Objekt extrahiert wird.

```
Time: 01/24/18 19:58:23.203350, usecs:1516823903203350
Sequence No : 0x2b
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_EXTRACT_MASKED_OBJECT_USER (0xf0)
Session Handle : 0x7004003
```

```
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

Die Protokollströme anderer HSMs im Cluster spiegeln die Ankunft des neu importierten Schlüssels wider.

Dieses Ereignis wurde beispielsweise im Protokollstream eines anderen HSM im gleichen Cluster aufgezeichnet. Dieses `CN_INSERT_MASKED_OBJECT_USER`-Ereignis zeichnet die Ankunft eines maskierten Objekts auf, das den Schlüssel 11 darstellt.

```
Time: 01/24/18 19:58:23.286793, usecs:1516823903286793
Sequence No : 0xb
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INSERT_MASKED_OBJECT_USER (0xf1)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

Beispiel: Freigeben eines Schlüssels und Aufheben der Freigabe

Dieses Beispiel zeigt das Prüfprotokollereignis, das aufgezeichnet wird, wenn ein CU den privaten ECC-Schlüssel für andere CUs freigibt oder die Freigabe aufhebt. Der CU verwendet den Befehl shareKey und stellt das Schlüssel-Handle, die Benutzer-ID und den Wert 1 zur Verfügung, um den Schlüssel freizugeben oder den Wert `0`, um die Freigabe aufzuheben.

Im folgenden Beispiel zeichnet der HSM, der den Befehl empfängt, ein `CM_SHARE_OBJECT`-Ereignis auf. Dieses Ereignis repräsentiert den Vorgang der Freigabe.

```
Time: 02/08/19 19:35:39.480168, usecs:1549654539480168
Sequence No : 0x3f
Reboot counter : 0x38
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_SHARE_OBJECT (0x12)
Session Handle : 0x3014007
Response : 0:HSM Return: SUCCESS
Log type : UNKNOWN_LOG_TYPE (5)
```

HSM-Auditprotokoll-Referenz

AWS CloudHSM zeichnet HSM-Verwaltungsbefehle in Audit-Protokollereignissen auf. Jedes Ereignis hat einen Operationscode-Wert (`0pcode`) der die aufgetretene Aktion und ihre Reaktion identifiziert. Sie können die `0pcode`-Werte zum Durchsuchen, Sortieren und Filtern der Protokolle verwenden.

Die folgende Tabelle definiert die `0pcode` Werte in einem AWS CloudHSM Audit-Protokoll.

Operationscode (Opcode)	Beschreibung
Benutzeranmeldung: Diese Ereignisse enthalten den Benutzernamen und Typ.	
`CN_LOGIN (0xd)`	Benutzerlogin
`CN_LOGOUT (0xe)`	Benutzerabmeldung
`CN_APP_FINALIZE`	Die Verbindung mit dem HSM wurde geschlossen. Alle Sitzungsschlüssel oder Quorum-Token aus dieser Verbindung wurden gelöscht.
`CN_CLOSE_SESSION`	Die Sitzung mit dem HSM wurde geschlossen. Alle Sitzungsschlüssel oder Quorum-Token aus dieser Sitzung wurden gelöscht.
Benutzerverwaltung: Diese Ereignisse enthalten den Benutzernamen und Typ.	
`CN_CREATE_USER (0x3)`	Erstellen eines Verwaltungsbenutzers (CU, Crypto User).
`CN_CREATE_CO`	Erstellen eines Verschlüsselungsverantwortlichen (CO)
`CN_DELETE_USER`	Löschen eines Benutzers
`CN_CHANGE_PSWD`	Ändern eines Benutzer-Passworts
`CN_SET_M_VALUE`	Set Quorum-Authentifizierung (M of N) for a user action
`CN_APPROVE_TOKEN`	Approve a Quorum-Authentifizierung token for a user action

Operationscode (Opcode)	Beschreibung
CN_DELETE_TOKEN	Delete one or more Quorum-Token
CN_GET_TOKEN	Request a signing token to initiate a Quorum-Operation
Schlüsselverwaltung: Diese Ereignisse enthalten das Schlüssel-Handle.	
CN_GENERATE_KEY	Generieren eines symmetrischen Schlüssels
CN_GENERATE_KEY_PAIR (0x19)	Generate an asymmetric key pair
CN_CREATE_OBJECT	Import a public key (without wrapping)
CN_MODIFY_OBJECT	Set a key attribute
CN_DESTROY_OBJECT (0x11)	Deletion of a Sitzungsschlüssel
CN_TOMBSTONE_OBJECT	Deletion of a Token-Schlüssel
CN_SHARE_OBJECT	Freigabe oder Aufheben der Freigabe eines Schlüssels
CN_WRAP_KEY	Export an encrypted copy of a key (wrapKey)
CN_UNWRAP_KEY	Import an encrypted copy of a key (unwrapKey)
CN_DERIVE_KEY	Derive a symmetric key from an existing key
CN_NIST_AES_WRAP	Verschlüsseln oder Entschlüsseln eines Schlüssels mit einem AES-Schlüssel
CN_INSERT_MASKED_OBJECT_USER	Insert an encrypted key with attributes from another HSM in the cluster.
CN_EXTRACT_MASKED_OBJECT_USER	Wraps/encrypts a key with attributes from the HSM to be sent to another HSM in the cluster.
Back up HSMs	
CN_BACKUP_BEGIN	Begin the backup process

Operationscode (Opcode)	Beschreibung
`CN_BACKUP_END`	Completed the backup process
`CN_RESTORE_BEGIN`	Begin restoring from a backup
`CN_RESTORE_END`	Completed the restoration process from a backup
Certificate-Based Authentication	
`CN_CERT_AUTH_STORE_CERT`	Stores the cluster certificate
HSM Instance Commands	
`CN_INIT_TOKEN (0x1)`	Start the HSM initialization process
`CN_INIT_DONE`	The HSM initialization process has finished
`CN_GEN_KEY_ENC_KEY`	Generate a key encryption key (KEK)
`CN_GEN_PSWD_ENC_KEY (0x1d)`	Generate a password encryption key (PEK)
HSM crypto commands	
`CN_FIPS_RAND`	Generate a FIPS-compliant random number

Abrufen von CloudWatch-Metriken für AWS CloudHSM

Verwenden Sie CloudWatch, um Ihren AWS CloudHSM-Cluster in Echtzeit zu überwachen. Die Metriken können nach Region, Cluster-ID oder Cluster-ID und HSM-ID gruppiert werden.

Der `AWS/CloudHSM`-Namespace enthält die folgenden Metriken:

Metrik	Beschreibung
HsmUnhealthy	Die HSM-Instance funktioniert nicht ordnungsgemäß. AWS CloudHSM ersetzt automatisch fehlerhafte Instances für Sie. Sie können die Cluster-Größe proaktiv erweitern, um die Auswirkungen auf die Leistung während des Austauschs des HSM zu reduzieren.

Metrik	Beschreibung
HsmTemperature	Die Junction-Temperatur des Hardware-Prozessors. Das System wird heruntergefahren, wenn die Temperatur 110 °C erreicht.
HsmKeysSessionOccupied	Die Anzahl der Sitzungsschlüssel, die von der HSM-Instance verwendet werden.
HsmKeysTokenOccupied	Die Anzahl der Token-Schlüssel, die von der HSM-Instance und dem Cluster verwendet werden.
HsmSslCtxsOccupied	Die Anzahl der End-to-End-verschlüsselten Kanäle, die derzeit für die HSM-Instance eingerichtet sind. Es sind bis zu 2.048 Kanäle zulässig.
HsmSessionCount	Die Anzahl der offenen Verbindungen zur HSM-Instance. Es sind bis zu 2.048 zulässig. Standardmäßig ist der Client-Daemon so konfiguriert, dass er zwei Sitzungen mit jeder HSM-Instance unter einem Ende-zu-Ende-verschlüsselten Kanal öffnet. AWS CloudHSM kann auch bis zu 2 Verbindungen mit dem HSM geöffnet haben, um den Zustand der HSMs zu überwachen.
HsmUsersAvailable	Die Anzahl der zusätzlichen Benutzer, die erstellt werden können. Dies entspricht der maximalen Anzahl von Benutzern (aufgeführt in HsmUsersMax) abzüglich der bisher erstellten Benutzer.
HsmUsersMax	Maximale Anzahl von Benutzern, die in der HSM-Instance erstellt werden können. Derzeit sind es 1.024.
InterfaceEth2OctetsInput	Die kumulierte Summe des bisher auf das HSM eingehenden Datenverkehrs.
InterfaceEth2OctetsOutput	Die kumulierte Summe des bisher auf das HSM ausgehenden Datenverkehrs.

AWS CloudHSM Leistung

Für Produktions-Cluster sollten Sie mindestens zwei HSM-Instances über verschiedene Availability Zones in einer Region verteilt haben. Wir empfehlen, Ihren Cluster einem Belastungstest zu unterziehen, um die zu erwartende Spitzenlast zu ermitteln, und dann ein weiteres HSM hinzuzufügen, um eine hohe Verfügbarkeit zu gewährleisten. Für Anwendungen, die auf zuverlässige, neu generierte Schlüssel angewiesen sind, empfehlen wir mindestens drei HSM-Instances, die über verschiedene Availability Zones in einer Region verteilt sind.

Leistungsdaten

Die Leistung von AWS CloudHSM Clustern hängt von der jeweiligen Arbeitslast ab. Die folgende Tabelle zeigt die ungefähre Leistung für gängige kryptografische Algorithmen, die auf einer EC2-Instance ausgeführt werden. Um die Leistung zu erhöhen, können Sie Ihren Clustern zusätzliche HSM-Instances hinzufügen. Die Leistung kann je nach Konfiguration, Datengröße und zusätzlicher Anwendungslast auf Ihren EC2-Instances variieren. Wir empfehlen Ihnen, Ihre Anwendung einem Belastungstest zu unterziehen, um den Skalierungsbedarf zu ermitteln.

Operation	Cluster mit zwei HSMs[1]	Cluster mit drei HSMs[2]	Cluster mit sechs HSMs[3]
RSA 2048-bit sign	2.000 Operationen/ Sekunde	3.000 Operationen/ Sekunde	5.000 Operationen/ Sekunde
EC P256 sign	500 Operationen/ Sekunde	750 Operationen/ Sekunde	1.500 Operationen/ Sekunde

- [1] Ein Cluster mit zwei HSMs, bei dem die Java-Multithread-Anwendung auf einer c4.large-EC2-Instance mit einem HSM in derselben AZ wie die EC2-Instance ausgeführt wird.
- [2] Ein Cluster mit drei HSMs, bei dem die Java-Multithread-Anwendung auf einer c4.large-EC2-Instance mit einem HSM in derselben AZ wie die EC2-Instance ausgeführt wird.
- [3] Ein Cluster mit sechs HSMs, bei dem die Java-Multithread-Anwendung auf einer c4.large-EC2-Instance mit zwei HSMs in derselben AZ wie die EC2-Instance ausgeführt wird.

HSM-Drosselung

Wenn Ihre Workload die HSM-Kapazität Ihres Clusters überschreitet, erhalten Sie Fehlermeldungen, die darauf hinweisen, dass HSMs ausgelastet oder gedrosselt sind. Einzelheiten dazu, was in einem solchen Fall zu tun ist, finden Sie unter HSM-Drosselung.

Sicherheit in AWS CloudHSM

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und 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 geteilten Verantwortung beschreibt dies als Sicherheit der Cloud selbst 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 geltenden Compliance-Programmen finden Sie unter AWS-Services in Umfang nach Compliance-Programm . AWS CloudHSM
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Nutzung anwenden können AWS CloudHSM. In den folgenden Themen erfahren Sie, wie Sie die Konfiguration vornehmen AWS CloudHSM , um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie lernen auch, wie Sie andere AWS-Services nutzen können, die Sie bei der Überwachung und Sicherung Ihrer AWS CloudHSM Ressourcen unterstützen.

Inhalt

Datenschutz in AWS CloudHSM

Das Modell der AWS gemeinsamen Verantwortung und geteilter Verantwortung gilt für den Datenschutz in AWS CloudHSM. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit der Konsole, der API, der CLI AWS CloudHSM oder den AWS SDKs arbeiten oder diese anderweitig AWS-Services verwenden. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine

Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Verschlüsselung im Ruhezustand

Wenn ein AWS CloudHSM Backup von einem HSM erstellt wird, verschlüsselt das HSM seine Daten, bevor es sie an sendet. AWS CloudHSM Die Daten werden mit einem spezifischen flüchtigen Verschlüsselungsschlüssel verschlüsselt. Weitere Informationen finden Sie unter Sicherungen von AWS CloudHSM-Clustern.

Verschlüsselung während der Übertragung

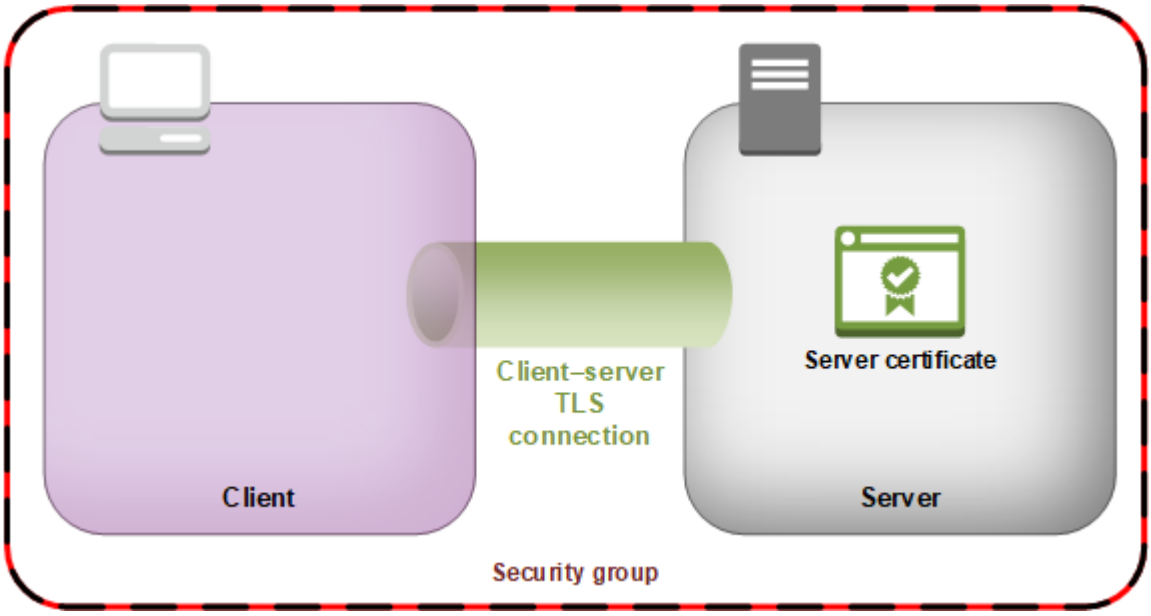
Die Kommunikation zwischen dem AWS CloudHSM Client und dem HSM in Ihrem Cluster ist von Ende zu Ende verschlüsselt. Diese Kommunikation kann nur vom Client und vom HSM entschlüsselt werden. Weitere Informationen finden Sie unter nd-to-end E-Verschlüsselung.

AWS CloudHSM Client-Verschlüsselung end-to-end

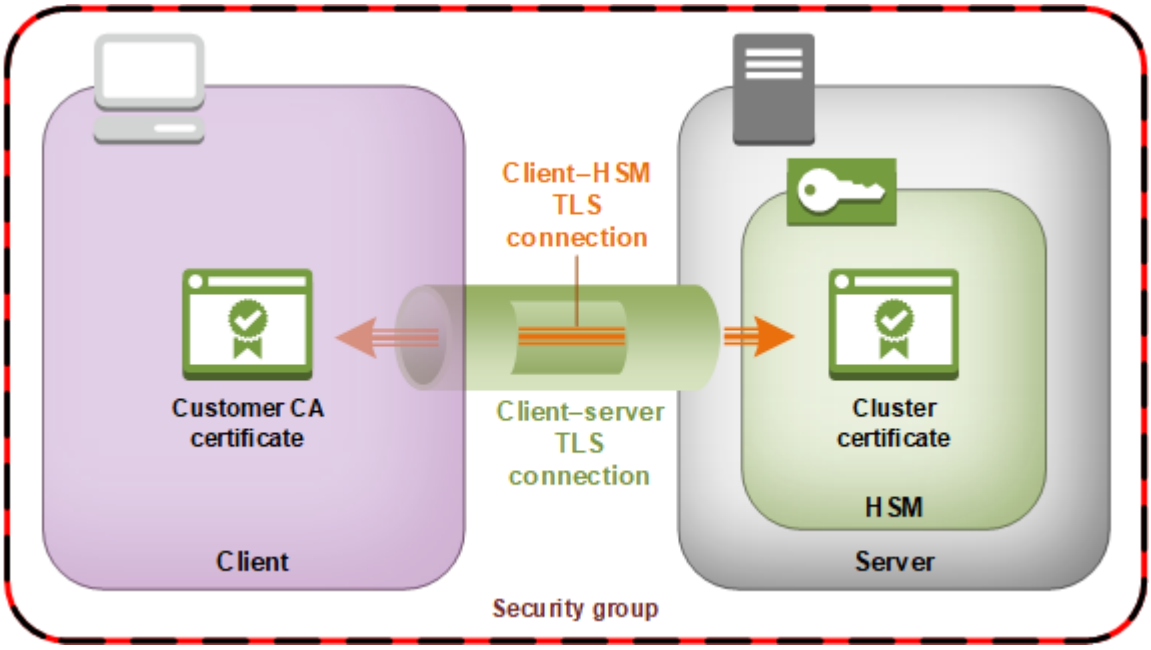
Die Kommunikation zwischen der Client-Instance und den HSMs in Ihrem Cluster ist End-to-End verschlüsselt. Nur Ihr Client und Ihre HSMs können die Kommunikation entschlüsseln.

Der folgende Prozess erklärt, wie der Client eine end-to-end verschlüsselte Kommunikation mit einem HSM herstellt.

1. Ihr Client stellt eine gegenseitig authentifizierte Transport Layer Security (TLS)-Verbindung zu dem Server her, der Ihre HSM-Hardware hostet. Die Sicherheitsgruppe Ihres Clusters erlaubt eingehenden Datenverkehr an den Server nur von Client-Instances in der Sicherheitsgruppe. Der Client überprüft auch das Zertifikat des Servers, um sicherzustellen, dass es sich um einen vertrauenswürdigen Server handelt.



2. Als nächstes richtet der Client eine verschlüsselte Verbindung mit der HSM-Hardware ein. Das HSM hat das Cluster-Zertifikat, das Sie mit Ihrer eigenen Zertifikatsstelle (CA) signiert haben, und der Client hat das Stammzertifikat der CA. Bevor die verschlüsselte Client-HSM-Verbindung eingerichtet wird, überprüft der Client das Cluster-Zertifikat des HSM anhand seines Stammzertifikats. Die Verbindung wird nur hergestellt, wenn der Client nur erfolgreich sichergestellt hat, dass der HSM vertrauenswürdig ist.



Sicherheit von Cluster-Sicherungen

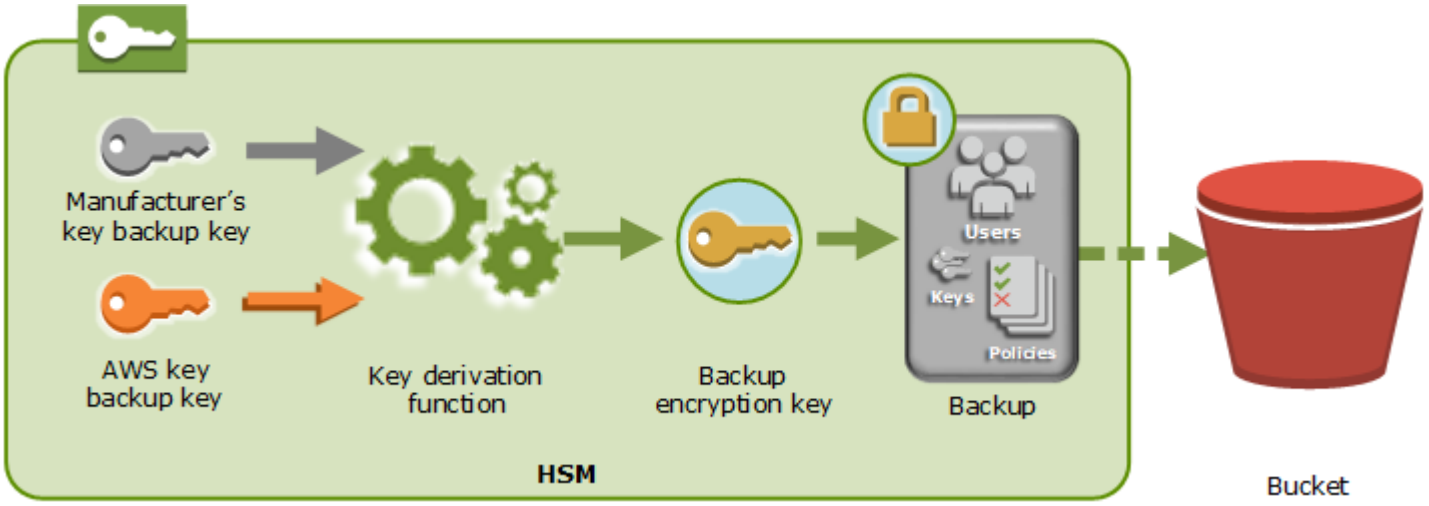
Wenn AWS CloudHSM ein Backup vom HSM erstellt wird, verschlüsselt das HSM alle Daten, bevor es sie an sendet. AWS CloudHSM Die Daten werden zu keinem Zeitpunkt unverschlüsselt aus dem HSM ausgegeben. Darüber hinaus können Backups nicht entschlüsselt werden, AWS da AWS es keinen Zugriff auf den Schlüssel hat, der zum Entschlüsseln der Backups verwendet wurde.

Um seine Daten zu verschlüsseln, verwendet das HSM einen eindeutigen, flüchtigen Verschlüsselungsschlüssel als vorübergehenden Schlüssel für die Sicherung (EBK, Ephemeral Backup Key). Das EBK ist ein AES-256-Bit-Verschlüsselungsschlüssel, der bei der Erstellung eines Backups im HSM generiert wird. AWS CloudHSM Das HSM generiert den EBK und verwendet ihn anschließend, um die Daten des HSM unter Verwendung einer FIPS-konformen Wrapping-Methode für AES-Schlüssel entsprechend der NIST Special Publication 800-38F zu verschlüsseln. Dann gibt das HSM die verschlüsselten Daten an weiter. AWS CloudHSM Die verschlüsselten Daten enthalten eine verschlüsselte Kopie des EBK.

Um den EBK zu verschlüsseln, verwendet HSM einen anderen Verschlüsselungsschlüssel, der als persistenter Sicherungsschlüssel (PBK, Persistent Backup Key) bezeichnet wird. Der PBK (Persistent Backup Key) ist ebenfalls ein AES 256-Bit-Verschlüsselungsschlüssel. Um den PBK zu erzeugen, verwendet das HSM eine FIPS-konforme Schlüsselableitungsfunktion (KDF, Key Derivation Function) in einem Counter-Modus entsprechend der NIST Special Publication 800-108. Diese KDF verwendet u. a. die folgenden Eingaben:

- Ein Herstellerschlüssel zur Schlüsselsicherung (MKBK, Manufacturer Key Backup Key), der vom Hardwarehersteller fest in die HSM-Hardware integriert ist
- Ein AWS Schlüssel-Backup-Key (AKBK), der bei der ersten Konfiguration von sicher im HSM installiert ist. AWS CloudHSM

Die Verschlüsselungsverfahren sind in der folgenden Abbildung zusammengefasst. Der Sicherungs-Verschlüsselungsschlüssel stellt den PBK und den EBK dar.



AWS CloudHSM kann Backups nur auf AWS eigenen HSMs desselben Herstellers wiederherstellen. Da jede Sicherung alle Benutzer, Schlüssel und Konfigurationen des ursprünglichen HSM enthält, enthält das wiederhergestellte HSM enthält dieselben Schutzmechanismen und Zugriffssteuerungen wie das ursprüngliche. Bei der Wiederherstellung werden alle anderen Daten, die sich möglicherweise vor der Wiederherstellung noch auf dem HSM befinden, vollständig überschrieben.

Eine Sicherung besteht aus nur verschlüsselten Daten. Bevor der Service ein Backup in Amazon S3 speichert, verschlüsselt der Service das Backup erneut mit AWS Key Management Service (AWS KMS).

Identitäts- und Zugriffsmanagement für AWS CloudHSM

AWS verwendet Sicherheitsanmeldeinformationen, um Sie zu identifizieren und Ihnen Zugriff auf Ihre AWS-Ressourcen zu gewähren. Sie können Funktionen von AWS Identity and Access Management (IAM) verwenden, um anderen Benutzern, Services und Anwendungen die vollständige oder eingeschränkte Nutzung Ihrer AWS-Ressourcen zu ermöglichen. Sie können dies tun, ohne Ihre Sicherheitsanmeldeinformationen zu teilen.

IAM-Benutzer sind standardmäßig nicht berechtigt, AWS-Ressourcen zu erstellen, anzuzeigen oder zu ändern. Um einem IAM-Benutzer den Zugriff auf Ressourcen wie einen Load Balancer und das Ausführen von Aufgaben zu ermöglichen, gehen Sie folgendermaßen vor:

1. Erstellen Sie eine IAM-Richtlinie, die dem IAM-Benutzer die Berechtigung zur Nutzung der jeweiligen Ressourcen und API-Aktionen erteilt, die er benötigt.
2. Weisen Sie die Richtlinie dem IAM-Benutzer oder der Gruppe zu, zu der der IAM-Benutzer gehört.

Wenn Sie einem Benutzer oder einer Benutzergruppe eine Richtlinie zuordnen, wird den Benutzern die Ausführung der angegebenen Aufgaben für die angegebenen Ressourcen gestattet oder verweigert.

Sie können IAM beispielsweise verwenden, um Benutzer und Gruppen unter Ihrem AWS-Konto zu erstellen. Ein IAM-Benutzer kann eine Person, ein System oder eine Anwendung sein. Anschließend gewähren Sie den Benutzern und Gruppen Berechtigungen, um bestimmte Aktionen für die angegebenen Ressourcen mithilfe einer IAM-Richtlinie durchzuführen.

Erteilen von Berechtigungen mithilfe von IAM-Richtlinien

Wenn Sie einem Benutzer oder einer Benutzergruppe eine Richtlinie zuordnen, wird den Benutzern die Ausführung der angegebenen Aufgaben für die angegebenen Ressourcen gestattet oder verweigert.

Eine IAM-Richtlinie ist ein JSON-Dokument, das eine oder mehrere Anweisungen enthält. Jedes Statement ist dem folgenden Beispiel entsprechend strukturiert.

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect": "effect",
    "Action": "action",
    "Resource": "resource-arn",
    "Condition": {
      "condition": {
        "key":"value"
      }
    }
  }]
}
```

- Effect: Der effect-Wert kann `Allow` oder `Deny` lauten. IAM-Benutzer verfügen standardmäßig nicht über die Berechtigung zur Verwendung von Ressourcen und API-Aktionen. Daher werden alle Anfragen abgelehnt. Dieser Standardwert kann durch eine explizite Zugriffserlaubnis überschrieben werden. Eine explizite Zugriffsverweigerung überschreibt jedwede Zugriffserlaubnis.
- Aktion: Mit Aktion wird die API-Aktion spezifiziert, für die Sie Berechtigungen erteilen oder verweigern. Für weitere Informationen zum Angeben von Aktionen siehe API-Aktionen für AWS CloudHSM.

- Ressource — Die Ressource, die von der Aktion betroffen ist. AWS CloudHSM unterstützt keine Berechtigungen auf Ressourcenebene. Sie müssen den Platzhalter „*“ verwenden, um alle Ressourcen anzugeben. AWS CloudHSM
- Condition: Sie können optional Bedingungen verwenden, um zu steuern, wann die Richtlinie wirksam ist. Weitere Informationen finden Sie unter Bedingungsschlüssel für AWS CloudHSM.

Weitere Informationen finden Sie im IAM-Benutzerhandbuch.

API-Aktionen für AWS CloudHSM

Im Action-Element Ihrer IAM-Richtlinienerklärung können Sie jede API-Aktion angeben, die AWS CloudHSM angeboten wird. Dem Namen der Aktion muss wie im folgenden Beispiel die Zeichenfolge `cloudhsm:` in Kleinbuchstaben vorangestellt werden.

```
"Action": "cloudhsm:DescribeClusters"
```

Wenn Sie mehrere Aktionen in einer einzigen Anweisung angeben möchten, setzen Sie sie in eckige Klammern und trennen Sie sie wie im folgenden Beispiel dargestellt durch Kommas.

```
"Action": [
    "cloudhsm:DescribeClusters",
    "cloudhsm:DescribeHsm"
]
```

Sie können auch mehrere Aktionen mithilfe des Platzhalters * angeben. Im folgenden Beispiel werden alle API-Aktionsnamen angegeben AWS CloudHSM , die mit `List` beginnen.

```
"Action": "cloudhsm:List*"
```

Um alle API-Aktionen für anzugeben AWS CloudHSM, verwenden Sie den Platzhalter *, wie im folgenden Beispiel gezeigt.

```
"Action": "cloudhsm:*"
```

Eine Liste der API-Aktionen für finden Sie AWS CloudHSM unter AWS CloudHSM Aktionen.

Bedingungsschlüssel für AWS CloudHSM

Beim Erstellen einer Richtlinie können Sie die Bedingungen angeben, die steuern, wann die Richtlinie wirksam wird. Jede Bedingung enthält ein oder mehrere Schlüssel-Wert-Paare. Es gibt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel.

AWS CloudHSM hat keine dienstspezifischen Kontextschlüssel.

Weitere Informationen über globale Bedingungsschlüssel finden Sie unter Globale Bedingungskontextschlüssel in AWS im IAM-Benutzer-Leitfaden.

Vordefinierte AWS-verwaltete Richtlinien für AWS CloudHSM

Die von AWS erstellten verwalteten Richtlinien erteilen die erforderlichen Berechtigungen für häufige Anwendungsfälle. Sie können diese Richtlinien basierend auf dem erforderlichen Zugriff auf AWS CloudHSM an IAM-Benutzer anfügen:

- AWSCloudHSMFullAccess— Gewährt vollen Zugriff, der für die Nutzung der AWS CloudHSM Funktionen erforderlich ist.
- AWSCloudHSMReadOnlyAccess— Gewährt schreibgeschützten Zugriff auf AWS CloudHSM Funktionen.

Vom Kunden verwaltete Richtlinien für AWS CloudHSM

Wir empfehlen Ihnen, dafür eine IAM-Administratorgruppe zu erstellen AWS CloudHSM , die nur die für die Ausführung AWS CloudHSM erforderlichen Berechtigungen enthält. Fügen Sie die Richtlinie mit den benötigten Berechtigungen an diese Gruppe an. Fügen Sie der Gruppe bei Bedarf IAM-Benutzer hinzu. Jeder hinzugefügte Benutzer erbt die Richtlinie von der Administratorgruppe.

Außerdem empfehlen wir, zusätzliche Benutzergruppen basierend auf den Berechtigungen zu erstellen, die die jeweiligen Benutzer benötigen. Dadurch wird sichergestellt, dass nur vertrauenswürdige Benutzer Zugriff auf kritische API-Aktionen erhalten. Sie können beispielsweise eine Benutzergruppe erstellen, mit der Sie schreibgeschützten Zugriff auf Cluster und HSMs gewähren. Da diese Gruppe einem Benutzer das Löschen von Clustern oder HSMs nicht erlaubt, kann ein nicht vertrauenswürdiger Benutzer die Verfügbarkeit einer Produktions-Workload nicht beeinträchtigen.

Da im Laufe der Zeit neue AWS CloudHSM Verwaltungsfunktionen hinzugefügt werden, können Sie sicherstellen, dass nur vertrauenswürdige Benutzer sofort Zugriff erhalten. Indem Sie den Richtlinien

bei der Erstellung begrenzte Berechtigungen zuweisen, können Sie diesen zu einem späteren Zeitpunkt neue Funktionsberechtigungen manuell zuweisen.

Im Folgenden finden Sie Beispielrichtlinien für AWS CloudHSM. Informationen zum Erstellen einer Richtlinie und zum Anfügen an eine IAM-Benutzergruppe finden Sie unter Erstellen von Richtlinien auf der Registerkarte „JSON“ im IAM-Benutzer-Leitfaden.

Beispiele

- Berechtigungen mit Schreibschutz
- Berechtigungen für Hauptbenutzer
- Berechtigungen für Administratoren

Example Beispiel: Berechtigungen mit Schreibschutz

Diese Richtlinie gewährt Zugriff auf die API-Aktionen `DescribeClusters` und `DescribeBackups`. Sie enthält außerdem zusätzliche Berechtigungen für bestimmte Amazon-EC2-API-Aktionen. Sie erlaubt jedoch nicht, dass der Benutzer Cluster oder HSMs löscht.

```
{
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:ListTags"
      ],
      "Resource": "*"
   }
}
```

Example Beispiel: Berechtigungen für Hauptbenutzer

Diese Richtlinie ermöglicht den Zugriff auf eine Teilmenge der AWS CloudHSM API-Aktionen. Sie enthält außerdem zusätzliche Berechtigungen für bestimmte Amazon-EC2-Aktionen. Sie erlaubt jedoch nicht, dass der Benutzer Cluster oder HSMs löscht. Sie müssen die `iam:CreateServiceLinkedRole` Aktion angeben, um die automatische Erstellung der AWSServiceRoleForCloudHSMdienstbezogenen Rolle in Ihrem Konto AWS CloudHSM zu

ermöglichen. Diese Rolle ermöglicht das AWS CloudHSM Protokollieren von Ereignissen. Weitere Informationen finden Sie unter Mit Diensten verknüpfte Rollen für AWS CloudHSM.

> **Note**
>
> Eine Liste der Aktionen in jedem Service finden Sie unter Aktionen, Ressourcen und Konditionsschlüssel für AWS CloudHSM in der Referenz zur Serviceberechtigung.

```
{
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:CreateCluster",
         "cloudhsm:CreateHsm",
         "cloudhsm:RestoreBackup",
         "cloudhsm:CopyBackupToRegion",
         "cloudhsm:InitializeCluster",
         "cloudhsm:ListTags",
         "cloudhsm:TagResource",
         "cloudhsm:UntagResource",
         "ec2:CreateNetworkInterface",
         "ec2:DescribeNetworkInterfaces",
         "ec2:DescribeNetworkInterfaceAttribute",
         "ec2:DetachNetworkInterface",
         "ec2:DeleteNetworkInterface",
         "ec2:CreateSecurityGroup",
         "ec2:AuthorizeSecurityGroupIngress",
         "ec2:AuthorizeSecurityGroupEgress",
         "ec2:RevokeSecurityGroupEgress",
         "ec2:DescribeSecurityGroups",
         "ec2:DeleteSecurityGroup",
         "ec2:CreateTags",
         "ec2:DescribeVpcs",
         "ec2:DescribeSubnets",
         "iam:CreateServiceLinkedRole"
      ],
      "Resource": "*"
   }
```

```
}
```

Example Beispiel: Berechtigungen für Administratoren

Diese Richtlinie ermöglicht den Zugriff auf alle AWS CloudHSM API-Aktionen, einschließlich der Aktionen zum Löschen von HSMs und Clustern. Sie enthält außerdem zusätzliche Berechtigungen für bestimmte Amazon-EC2-Aktionen. Sie müssen die `iam:CreateServiceLinkedRole` Aktion angeben, um die automatische Erstellung der AWSServiceRoleForCloudHSMdienstbezogenen Rolle in Ihrem Konto AWS CloudHSM zu ermöglichen. Diese Rolle ermöglicht das AWS CloudHSM Protokollieren von Ereignissen. Weitere Informationen finden Sie unter Mit Diensten verknüpfte Rollen für AWS CloudHSM.

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":[
         "cloudhsm:*",
         "ec2:CreateNetworkInterface",
         "ec2:DescribeNetworkInterfaces",
         "ec2:DescribeNetworkInterfaceAttribute",
         "ec2:DetachNetworkInterface",
         "ec2:DeleteNetworkInterface",
         "ec2:CreateSecurityGroup",
         "ec2:AuthorizeSecurityGroupIngress",
         "ec2:AuthorizeSecurityGroupEgress",
         "ec2:RevokeSecurityGroupEgress",
         "ec2:DescribeSecurityGroups",
         "ec2:DeleteSecurityGroup",
         "ec2:CreateTags",
         "ec2:DescribeVpcs",
         "ec2:DescribeSubnets",
         "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*"
   }
}
```

Mit Diensten verknüpfte Rollen für AWS CloudHSM

Die IAM-Richtlinie, die Sie zuvor erstellt haben, Vom Kunden verwaltete Richtlinien für AWS CloudHSM beinhaltet die `iam:CreateServiceLinkedRole` Aktion. AWS CloudHSM definiert eine serviceverknüpfte Rolle mit dem Namen. AWSServiceRoleForCloudHSM Die Rolle ist vordefiniert von AWS CloudHSM und umfasst Berechtigungen, die AWS CloudHSM erforderlich sind, um andere AWS Dienste in Ihrem Namen aufzurufen. Die Rolle vereinfacht die Einrichtung Ihres Service, da Sie die Rollenrichtlinie und Berechtigungen der Vertrauensrichtlinie nicht manuell hinzufügen müssen.

Die Rollenrichtlinie ermöglicht es AWS CloudHSM , Amazon CloudWatch Logs-Protokollgruppen und Log-Streams zu erstellen und Protokollereignisse in Ihrem Namen zu schreiben. Sie können sie nachstehend und in der IAM-Konsole einsehen.

```
{
    "Version": "2018-06-12",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

Die Vertrauensrichtlinie für die AWSServiceRoleForCloudHSMRolle ermöglicht es AWS CloudHSM , die Rolle zu übernehmen.

```
{
  "Version": "2018-06-12",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Erstellen einer serviceverknüpften Rolle (automatisch)

AWS CloudHSM erstellt die AWSServiceRoleForCloudHSMRolle, wenn Sie einen Cluster erstellen, wenn Sie die `iam:CreateServiceLinkedRole` Aktion in die Berechtigungen aufnehmen, die Sie bei der Erstellung der AWS CloudHSM Administratorgruppe definiert haben. Siehe Vom Kunden verwaltete Richtlinien für AWS CloudHSM.

Wenn Sie bereits über einen oder mehrere Cluster verfügen und nur die AWSServiceRoleForCloudHSMRolle hinzufügen möchten, können Sie die Konsole, den Befehl create-cluster oder die CreateClusterAPI-Operation verwenden, um einen Cluster zu erstellen. Verwenden Sie dann die Konsole, den Befehl delete-cluster oder den DeleteClusterAPI-Vorgang, um ihn zu löschen. Beim Erstellen des neuen Clusters wird die serviceverknüpfte Rolle erstellt und auf alle Cluster in Ihrem Konto angewandt. Alternativ können Sie die Rolle manuell erstellen. Weitere Informationen finden Sie im folgenden Abschnitt.

> **Note**
>
> Sie müssen nicht alle unter Erstellen eines Clusters beschriebenen Schritte ausführenErste Schritte mit AWS CloudHSM, wenn Sie ihn nur erstellen, um die Rolle hinzuzufügen. AWSServiceRoleForCloudHSM

Erstellen einer serviceverknüpften Rolle (manuell)

Sie können die IAM-Konsole oder API verwenden AWS CLI, um die AWSServiceRoleForCloudHSMRolle zu erstellen. Weitere Informationen finden Sie unter Erstellen einer serviceverknüpfte Rolle im IAM-Leitfaden.

Bearbeiten der serviceverknüpften Rolle

AWS CloudHSM erlaubt Ihnen nicht, die AWSServiceRoleForCloudHSMRolle zu bearbeiten. Nachdem die Rolle erstellt wurde, können Sie z. B. nicht ihren Namen ändern, da möglicherweise

verschiedene Entitäten unter dem Namen auf die Rolle verweisen. Die Rollenrichtlinie kann ebenfalls nicht geändert werden. Sie können mithilfe von IAM jedoch die Beschreibung der Rolle bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer serviceverknüpften Rolle im IAM-Benutzerhandbuch.

Löschen der serviceverknüpften -Rolle

Eine serviceverknüpfte Rolle kann nicht gelöscht werden, solange noch Cluster vorhanden sind, auf die sie angewendet wurde. Um die Rolle zu löschen, müssen Sie zuerst die einzelnen HSM im Cluster und dann den Cluster selbst löschen. Jeder Cluster in Ihrem Konto muss gelöscht werden. Sie können dann die IAM-Konsole oder API verwenden AWS CLI, um die Rolle zu löschen. Weitere Informationen zum Löschen eines Clusters finden Sie unter Löschen eines AWS CloudHSM Clusters. Weitere Informationen finden Sie unter Löschen einer serviceverknüpften Rolle im IAM-Leitfaden.

-Compliance

AWS CloudHSM bietet FIPS-anerkannte Sicherheitsrichtlinien für die HSMs. AWS CloudHSM Erfüllt außerdem die PCI-PIN-, PCI-3DS- und SOC2-Konformitätsanforderungen. Wenn Sie sich auf ein FIPS-validiertes HSM verlassen, können Sie die unternehmensinternen, vertraglichen und behördlichen Compliance-Anforderungen für die Datensicherheit in der Cloud erfüllen. AWS Weitere Informationen finden Sie unten.

Compliance mit FIPS 140-2

> Die Veröffentlichung 140-2 des Federal Information Processing Standard (FIPS) ist ein Sicherheitsstandard der US-Regierung, mit dem die Sicherheitsanforderungen für Verschlüsselungsmodule angegeben werden, die vertrauliche Informationen schützen. Die von bereitgestellten HSM AWS CloudHSM sind nach FIPS 140-2 Level 3 zertifiziert (Zertifikat #4218). Weitere Informationen dazu finden Sie unter FIPS-Validierung für Hardware.

PCI-DSS-Compliance

> Der Payment Card Industry Data Security Standard (PCI-DSS) ist ein vom Payment Card Industry Security Standards Council festgelegter proprietärer Standard für Informationssicherheit. Die von bereitgestellten HSMs entsprechen PCI DSS AWS CloudHSM .

PCI-PIN-Compliance

> PCI PIN enthält Sicherheitsanforderungen und Bewertungsstandards für die Übertragung, Verarbeitung und Verwaltung von PIN-Daten (Personal Identification Number), d. h.

Informationen, die für Transaktionen an Geldautomaten und point-of-sale (POS-) Terminals verwendet werden. AWS CloudHSM ist seit Januar 2023 PCI-PIN-konform. Weitere Informationen finden Sie im Artikel AWS CloudHSM ist jetzt PCI-PIN-zertifiziert.

Konformität mit PCI-3DS

PCI 3DS (oder Three Domain Secure, 3-D Secure) bietet Datensicherheit für sichere EMV-3D-E-Commerce-Zahlungen. PCI 3DS bietet eine weitere Sicherheitsebene für Online-Einkäufe. AWS CloudHSM ist PCI-3DS-konform.

SOC2

SOC2 ist ein Framework, das Serviceunternehmen dabei unterstützt, ihre Sicherheitskontrollen für Cloud und Rechenzentren nachzuweisen. AWS CloudHSM hat SOC2-Kontrollen in kritischen Bereichen implementiert, um den Grundsätzen für vertrauenswürdige Dienste zu entsprechen. Weitere Informationen finden Sie auf der Seite mit häufig gestellten Fragen zu AWS SOC.

AWS CloudHSM Häufig gestellte Fragen zur PCI-PIN-Konformität

Die PCI-PIN enthält Sicherheitsanforderungen und Bewertungsstandards für die Übertragung, Verarbeitung und Verwaltung von PIN-Daten (Personal Identification Number), d. h. Informationen, die für Transaktionen an Geldautomaten und point-of-sale (POS-) Terminals verwendet werden.

Die PCI-PIN-Konformitätsbescheinigung (AOC) und die Zusammenfassung der Verantwortlichkeiten stehen Kunden über AWS Artifact zur Verfügung, ein Self-Service-Portal für den On-Demand-Zugriff auf AWS-Compliance-Berichte. Für weitere Informationen melden Sie sich in der AWS-Managementkonsole bei AWS Artifact an. Sie erfahren auch mehr unter Erste Schritte mit AWS Artifact.

Häufig gestellte Fragen

F: Was ist die Konformitätsbescheinigung und die Zusammenfassung der Verantwortlichkeiten?

Die Konformitätsbescheinigung (AOC) wird von einem Qualified PIN Assessor (QPA) ausgestellt, der bescheinigt, dass die geltenden Kontrollen des PCI-PIN-Standards AWS CloudHSM eingehalten werden. In der Übersicht über die Zuständigkeiten werden die Kontrollen beschrieben, für die die jeweiligen Kunden verantwortlich sind. AWS CloudHSM

F: Wie erhalte ich die AWS CloudHSM Konformitätsbescheinigung?

Die PCI-PIN-Konformitätsbescheinigung (AOC) steht Kunden über AWS Artifact zur Verfügung, ein Self-Service-Portal für den On-Demand-Zugriff auf AWS-Compliance-Berichte. Für weitere Informationen melden Sie sich in der AWS-Managementkonsole bei AWS Artifact an. Sie erfahren auch mehr unter Erste Schritte mit AWS Artifact.

F: Wie kann ich erfahren, für welche PCI-PIN-Kontrollen ich verantwortlich bin?

Detaillierte Informationen finden Sie unter „Zusammenfassung der AWS CloudHSM PCI-PIN-Verantwortung“ aus dem AWS PCI PIN Compliance Package, das Kunden über AWS Artifact, ein Self-Service-Portal für On-Demand-Zugriff auf AWS-Compliance-Berichte, zur Verfügung steht. Für weitere Informationen melden Sie sich in der AWS-Managementkonsole bei AWS Artifact an. Sie erfahren auch mehr unter Erste Schritte mit AWS Artifact.

F: Kann ich mich als AWS CloudHSM Kunde auf die PCI-PIN Attestion of Compliance (AOC) verlassen?

Kunden müssen ihre PCI-PIN-Konformität selbst verwalten. Sie müssen ein formelles PCI-PIN-Bescheinigungsverfahren durch einen qualifizierten PIN-Assessor (QPA) durchlaufen, um zu überprüfen, ob Ihre Zahlungs-Workload alle PCI-PIN-Kontrollen/-Anforderungen erfüllt. Für die Kontrollen, für die AWS verantwortlich ist, kann sich Ihre QPA jedoch ohne weitere Tests auf die Konformitätsbescheinigung (AWS CloudHSM Attestation of Compliance, AOC) verlassen.

F: Ist AWS CloudHSM verantwortlich für die PCI-PIN-Anforderungen im Zusammenhang mit dem Lebenszyklus der Schlüsselverwaltung?

AWS CloudHSM ist verantwortlich für den physischen Gerätelebenszyklus der HSMs. Die Kunden sind für die wichtigsten Anforderungen des PCI-PIN-Standards über den Lebenszyklus der Verwaltung verantwortlich.

F: Welche AWS CloudHSM Steuerungen sind PCI-PIN-konform?

Das AOC fasst die AWS CloudHSM Kontrollen zusammen, die von QPA bewertet werden. Die PCI-PIN-Zusammenfassung der Verantwortlichkeiten steht Kunden über AWS Artifact zur Verfügung, ein Self-Service-Portal für den On-Demand-Zugriff auf AWS-Compliance-Berichte.

F: AWS CloudHSM Unterstützt es Zahlungsfunktionen wie PIN-Übersetzung und DUKPT?

Nein, AWS CloudHSM bietet Allzweck-HSMs. Im Laufe der Zeit können wir ggf. Zahlungsfunktionen anbieten. Obwohl der Dienst Zahlungsfunktionen nicht direkt ausführt, ermöglicht die AWS CloudHSM PCI-PIN-Konformitätsbescheinigung den Kunden, ihre eigene PCI-Konformität für ihre

Dienste zu erreichen, auf denen sie ausgeführt werden. AWS CloudHSM Wenn Sie daran interessiert sind, die Services von AWS Payment Cryptography für Ihre Workload zu nutzen, lesen Sie bitte den Blog „Verlagerung der Zahlungsabwicklung in die Cloud mit AWS Payment Cryptography“.

Benachrichtigungen über veraltete Funktionen

AWS CloudHSM Kann von Zeit zu Zeit Funktionen als veraltet einstufen, um weiterhin die Anforderungen von FIPS 140, PCI-DSS, PCI-PIN, PCI-3DS und SOC2 zu erfüllen. Auf dieser Seite sind die derzeit geltenden Änderungen aufgeführt.

FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt

Das National Institute of Standards and Technology (NIST) [1] weist darauf hin, dass die Unterstützung von Triple-DES-Verschlüsselung (DESede, 3DES, DES3) und RSA-Schlüssel-Verpackung und -Entpackung mit PKCS-#1 v1.5 Padding nach dem 31. Dezember 2023 nicht mehr zulässig ist. Daher endet die Unterstützung für diese Geräte in unseren mit dem Federal Information Processing Standard (FIPS) konformen Instances am 1. Januar 2024.

Diese Anleitung gilt für die folgenden kryptografischen Operationen:

- Generierung von Triple-DES-Schlüsseln
  - `CKM_DES3_KEY_GEN` für die PKCS-#11-Bibliothek
  - `DESede`-Keygen für den JCE-Anbieter
  - `genSymKey` mit `-t=21` für die KMU
- Verschlüsselung mit Triple-DES-Schlüsseln (Hinweis: Entschlüsselungsvorgänge sind zulässig)
  - Für die PKCS-#11-Bibliothek: `CKM_DES3_CBC` verschlüsseln, `CKM_DES3_CBC_PAD` verschlüsseln und `CKM_DES3_ECB` verschlüsseln
  - Für den JCE-Anbieter: `DESede/CBC/PKCS5Padding` verschlüsseln, `DESede/CBC/NoPadding` verschlüsseln, `DESede/ECB/Padding` verschlüsseln und `DESede/ECB/NoPadding` verschlüsseln
- Verpacken, Entpacken, Verschlüsseln und Entschlüsseln von RSA-Schlüsseln mit PKCS #1 v1.5-Padding
  - `CKM_RSA_PKCS`Verpacken, Entpacken, Verschlüsseln und Entschlüsseln für das PKCS-#11-SDK
  - `RSA/ECB/PKCS1Padding` für das JCE SDK einpacken, entpacken, verschlüsseln und entschlüsseln

- `wrapKey` und `unWrapKey` mit `-m 12` für die KMU (Hinweis: `12` ist der Wert für den Mechanismus `RSA_PKCS`)

[1] Einzelheiten zu dieser Änderung finden Sie in Tabelle 1 und Tabelle 5 unter Umstellung auf den Einsatz kryptografischer Algorithmen und Schlüssellängen.

Belastbarkeit in AWS CloudHSM

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability 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 Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur. Für weitere Informationen zu AWS CloudHSM -Funktionen, die der Unterstützung der Resilienz dienen, siehe Hohe Verfügbarkeit und Load Balancing von Clustern.

Sicherheit der Infrastruktur in AWS CloudHSM

Als verwalteter Service AWS CloudHSM ist er durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben sind.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff AWS CloudHSM über das Netzwerk. 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.

Netzwerkisolierung

Eine Virtual Private Cloud (VPC) ist ein virtuelles Netzwerk in Ihrem eigenen logisch isolierten Bereich in der AWS Cloud. Sie können einen Cluster in einem privaten Subnetz der VPC erstellen. Sie können private Subnetze erstellen, wenn Sie eine VPC erstellen. Weitere Informationen finden Sie unter Erstellen einer Virtual Private Cloud (VPC).

Wenn Sie ein HSM erstellen, AWS CloudHSM fügen Sie ein elastic network interface (ENI) in Ihr Subnetz ein, damit Sie mit Ihren HSMs interagieren können. Weitere Informationen finden Sie unter Clusterarchitektur.

AWS CloudHSM erstellt eine Sicherheitsgruppe, die eingehende und ausgehende Kommunikation zwischen HSMs in Ihrem Cluster ermöglicht. Sie können diese Sicherheitsgruppe verwenden, um EC2-Instances die Kommunikation mit den HSMs im Cluster zu ermöglichen. Weitere Informationen finden Sie unter Konfiguration der Sicherheitsgruppen der Amazon-EC2-Client-Instance.

Autorisierung von Benutzern

Mit erfordern Operationen AWS CloudHSM, die auf dem HSM ausgeführt werden, die Anmeldeinformationen eines authentifizierten HSM-Benutzers. Weitere Informationen finden Sie unter the section called “Grundlegendes zu HSM-Benutzern”.

AWS CloudHSM und VPC-Endpunkte

Sie können eine private Verbindung zwischen Ihrer VPC herstellen und AWS CloudHSM einen VPC-Schnittstellen-Endpunkt erstellen. Schnittstellenendpunkte werden von einer Technologie unterstützt AWS PrivateLink, mit der Sie privat auf AWS CloudHSM APIs zugreifen können, ohne dass ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder eine AWS Direct Connect-Verbindung erforderlich ist. Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen, um mit AWS CloudHSM APIs zu kommunizieren. Datenverkehr zwischen Ihrer VPC und AWS CloudHSM verlässt das Amazon-Netzwerk nicht.

Jeder Schnittstellenendpunkt wird durch eine oder mehrere Elastic-Network-Schnittstellen in Ihren Subnetzen dargestellt.

Weitere Informationen finden Sie unter Interface VPC Endpoints (AWS PrivateLink) im Amazon VPC-Benutzerhandbuch.

Überlegungen zu AWS CloudHSM VPC-Endpunkten

Bevor Sie einen Schnittstellen-VPC-Endpunkt für einrichten AWS CloudHSM, sollten Sie die Eigenschaften und Einschränkungen der Schnittstellen-Endpunkte im Amazon VPC-Benutzerhandbuch lesen.

- AWS CloudHSM unterstützt Aufrufe aller API-Aktionen von Ihrer VPC aus.

Erstellen eines Schnittstellen-VPC-Endpunkts für AWS CloudHSM

Sie können einen VPC-Endpunkt für den AWS CloudHSM Service entweder mit der Amazon VPC-Konsole oder der AWS Command Line Interface (CLI) erstellen. Weitere Informationen finden Sie unter Erstellung eines Schnittstellenendpunkts im Benutzerhandbuch für Amazon VPC.

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

```
com.amazonaws.region.cloudhsmv2
```

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

```
com.amazonaws.us-west-2.cloudhsmv2
```

Um die Verwendung des VPC-Endpunkts zu vereinfachen, können Sie einen privaten DNS-Hostnamen für Ihren VPC-Endpunkt aktivieren. Wenn Sie die Option Privaten DNS-Namen aktivieren auswählen, wird der AWS CloudHSM Standard-DNS-Hostname (`https://cloudhsmv2.<region>.amazonaws.com`) zu Ihrem VPC-Endpunkt aufgelöst.

Diese Option vereinfacht die Verwendung des VPC-Endpunkts. Die AWS SDKs und die CLI verwenden standardmäßig den AWS CloudHSM 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 Benutzerhandbuch für Amazon VPC.

Erstellen einer VPC-Endpunktrichtlinie für AWS CloudHSM

Sie können eine Endpunktrichtlinie an Ihren VPC-Endpunkt anhängen, der den Zugriff auf AWS CloudHSM steuert. Die Richtlinie gibt die folgenden Informationen an:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, für die Aktionen ausgeführt werden können.

Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon-VPC-Benutzerhandbuch.

Beispiel: VPC-Endpunktrichtlinie für Aktionen AWS CloudHSM

Im Folgenden finden Sie ein Beispiel für eine Endpunktrichtlinie für AWS CloudHSM. Wenn diese Richtlinie an einen Endpunkt angehängt ist, gewährt sie allen Prinzipalen auf allen Ressourcen Zugriff auf die aufgelisteten AWS CloudHSM Aktionen. Identitäts- und Zugriffsmanagement für AWS CloudHSMWeitere AWS CloudHSM Aktionen und die entsprechenden IAM-Berechtigungen finden Sie unter.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "cloudhsm:DescribeBackups",
            "cloudhsm:DescribeClusters",
            "cloudhsm:ListTags",
         ],
         "Resource":"*"
      }
   ]
}
```

Aktualisieren Sie die Verwaltung in AWS CloudHSM

AWS verwaltet die Firmware. Die Pflege der Firmware erfolgt durch einen Drittanbieter. Die Firmware muss von NIST hinsichtlich der Compliance mit FIPS 140-2 Level 3 bewertet werden. Es kann nur mit dem FIPS-Schlüssel (auf den AWS keinen Zugriff hat) kryptografisch signierte Firmware installiert werden.

Fehlerbehebung für AWS CloudHSM

Wenn Sie Probleme mit AWS CloudHSM haben, finden Sie möglicherweise in den folgenden Themen Hinweise, wie diese Probleme behoben werden können.

Themen

Bekannte Probleme

AWS CloudHSM hat die folgenden bekannten Probleme. Wählen Sie ein Thema, um mehr zu erfahren.

Themen

Bekannte Probleme für alle HSM-Instances

Die folgenden Probleme betreffen alle AWS CloudHSM Benutzer, unabhängig davon, ob sie das Befehlszeilentool key_mgmt_util, das PKCS #11 -SDK, das JCE SDK oder das OpenSSL SDK verwenden.

Themen

Problem: Beim AES Key Wrapping wird anstelle der standardkonformen Implementierung der Schlüsselverpackung ohne Padding das PKCS #5 Padding verwendet.

Des Weiteren werden Schlüsselverpackungen ohne Padding und mit Zero Padding nicht unterstützt.

- Auswirkung: Das Ein- und Auspacken mit diesem Algorithmus hat keine Auswirkungen. AWS CloudHSM Schlüssel, die mit eingeschlossen sind, können jedoch AWS CloudHSM nicht aus anderen HSMs oder Software entfernt werden, die die Einhaltung der No-Padding-Spezifikation voraussetzt. Dies liegt daran, dass während des standardkonformen Entpackens am Ende Ihrer Schlüsseldaten acht Bytes Padding-Daten hinzugefügt werden können. Extern verpackte Schlüssel können nicht ordnungsgemäß in eine Instanz entpackt werden. AWS CloudHSM
- Problemumgehung: Wenn Sie einen Schlüssel extern entpacken, der mit AES Key Wrapping und PKCS #5 Padding auf einer AWS CloudHSM-Instance verpackt wurde, entfernen Sie vor Verwendung des Schlüssels das zusätzliche Padding. Zu diesem Zweck können Sie die zusätzlichen Bytes in einem Datei-Editor abtrimmen oder nur die Schlüsselbytes in einen neuen Puffer in Ihrem Code kopieren.
- Stand der Lösung: Mit der Client- und Software-Version 3.1.0 bietet AWS CloudHSM standardkonforme Optionen für AES Key Wrapping. Weitere Informationen finden Sie unter AES Key Wrapping.

Problem: Der Client-Daemon benötigt mindestens eine gültige IP-Adresse in seiner Konfigurationsdatei, um erfolgreich eine Verbindung mit dem Cluster herstellen zu können.

- Auswirkung: Wenn Sie jedes HSM in Ihrem Cluster löschen und dann ein neues HSM hinzufügen, das eine neue IP-Adresse erhält, sucht der Client-Daemon weiterhin unter den ursprünglichen IP-Adressen nach Ihren HSMs.
- Umgehung: Wenn Sie einen intermittierenden Workload ausführen, empfehlen wir, das `IpAddress` Argument in der CreateHsmFunktion zu verwenden, um das elastic network interface (ENI) auf seinen ursprünglichen Wert zu setzen. Beachten Sie, dass eine ENI spezifisch für eine Availability Zone (AZ) ist. Die Alternative ist, die Datei `/opt/cloudhsm/daemon/1/cluster.info` zu löschen und dann die Client-Konfiguration auf die IP-Adresse Ihres neuen HSM zurückzusetzen. Sie können den Befehl `client -a <IP address>` verwenden. Weitere Informationen finden Sie unter Installation und Konfiguration des AWS CloudHSM Clients (Linux) oder Installation und Konfiguration des AWS CloudHSM Clients (Windows).

Problem: Es gab eine Obergrenze von 16 KB für Daten, die gehasht und signiert werden können AWS CloudHSM

- Stand der Lösung: Daten mit weniger als 16 KB werden weiterhin zum Versehen mit einem Hash-Wert an HSM gesendet. Daten zwischen 16 KB und 64 KB können nun auch lokal in der Software mit einem Hash-Wert versehen werden. Der Client und die SDKs erzeugen explizit einen Fehler, wenn der Datenpuffer größer als 64 KB ist. Sie müssen Ihren Client und Ihr(e) SDK(s) auf Version 1.1.1 oder höher aktualisieren, um dieses Problem zu beheben.

Problem: Es konnte für importierte Schlüssel nicht angegeben werden, dass diese nicht exportierbar sein sollen.

- Stand der Lösung: Dieses Problem wurde behoben. Ihrerseits sind keine Maßnahmen erforderlich, um die Korrektur nutzen zu können.

Problem: Der Standardmechanismus für den WrapKey und die unWrapKey Befehle in key_mgmt_util wurde entfernt

- Lösung: Wenn Sie den WrapKey oder unWrapKey Befehle verwenden, müssen Sie die `-m` Option verwenden, um den Mechanismus anzugeben. Weitere Informationen finden Sie in den Beispielen im WrapKey oder in den unWrapKeyArtikeln.

Problem: Befindet sich ein einzelnes HSM in Ihrem Cluster, funktioniert HSM-Failover nicht ordnungsgemäß.

- Auswirkung: Wenn die einzelne HSM-Instance im Cluster an Konnektivität verliert, stellt der Client keine erneute Verbindung zu ihr her – auch dann nicht, wenn die HSM-Instance zu einem späteren Zeitpunkt wiederhergestellt wird.
- Problemumgehung: Wir empfehlen mindestens zwei HSM-Instances pro Produktions-Cluster. Wenn Sie diese Konfiguration verwenden, werden Sie nicht von diesem Problem betroffen sein. Deaktivieren Sie bei Clustern mit nur einem HSM den Client-Daemon, um die Konnektivität wiederherzustellen.
- Stand der Lösung: Dieses Problem wurde in AWS CloudHSM -Client Version 1.1.2 behoben. Sie müssen auf diesen Client upgraden, um das Problem zu beheben.

Problem: Wenn die Schlüsselkapazität der HSMs im Cluster innerhalb kurzer Zeit überschritten wird, geht der Client in den Status „Unbehandelter Fehler“ über.

- Auswirkung: Wenn der Client in den Status „Unbehandelter Fehler“ wechselt, hängt er sich auf und muss neu gestartet werden.
- Problemumgehung: (Problemumgehung) Testen Sie Ihren Durchsatz, um sicherzustellen, dass Sitzungsschlüssel nicht mit einer Rate erstellt werden, die der Client nicht verarbeiten kann. Sie können die Rate heruntersetzen, indem Sie dem Cluster ein HSM hinzufügen oder die Erstellung des Sitzungsschlüssels verzögern.
- Stand der Lösung: Dieses Problem wurde in AWS CloudHSM -Client Version 1.1.2 behoben. Sie müssen auf diesen Client upgraden, um das Problem zu beheben.

Problem: Digest-Operationen mit HMAC-Schlüsseln mit einer Größe von mehr als 800 Bytes werden nicht unterstützt.

- Auswirkung: HMAC-Schlüssel mit einer Größe von mehr als 800 Bytes können im HSM erstellt oder in das HSM importiert werden. Wenn Sie diesen größeren Schlüssel in einer Digest-Operation jedoch über die JCE oder key_mgmt_util verwenden, schlägt die Operation fehl. Beachten Sie, dass die Größe von HMAC-Schlüsseln beim Verwenden von PKCS11 auf 64 Bytes beschränkt wird.
- Problemumgehung: Wenn Sie HMAC-Schlüssel für Digest-Operationen im HSM verwenden, stellen Sie sicher, dass die Schlüsselgröße weniger als 800 Bytes beträgt.
- Stand der Lösung: Keine Angabe.

Problem: Das im Client-SDK 3 enthaltene Tool client_info löscht den Inhalt des durch das optionale Ausgabeargument angegebenen Pfads

- Auswirkung: Alle vorhandenen Dateien und Unterverzeichnisse unter dem angegebenen Ausgabepfad können dauerhaft verloren gehen.
- Problemumgehung: Verwenden Sie das optionale Argument `-output` *`path`* nicht, wenn Sie das `client_info`-Tool verwenden.
- Stand der Lösung: Dieses Problem wurde in Client SDK 3.3.2 Version behoben. Sie müssen auf diesen Client upgraden, um das Problem zu beheben.

Problem: Sie erhalten eine Fehlermeldung, wenn Sie das SDK-5-Configure Tool mit dem **`--cluster-id`**-Argument in containerisierten Umgebungen ausführen

Sie erhalten den folgenden Fehler, wenn Sie das Argument --cluster-id mit dem Configure Tool verwenden:

```
No credentials in the property bag
```

Dieser Fehler wird durch ein Update auf Version 2 des Instance Metadata Service (IMDSv2) verursacht. Weitere Informationen finden Sie in der IMDSv2-Dokumentation.

- Auswirkung: Dieses Problem betrifft Benutzer, die das Configure Tool auf SDK-Versionen 5.5.0 und höher in containerisierten Umgebungen ausführen und EC2-Instance-Metadaten zur Bereitstellung von Anmeldeinformationen verwenden.
- Umgehung: Legen Sie das PUT-Antwort-Hop-Limit auf mindestens zwei fest. Eine Anleitung dazu finden Sie unter Konfigurieren der Optionen für Instance-Metadaten.

Bekannte Probleme für den PKCS#11-Bibliothek

Themen

- Problem: Die Überprüfung von secp256k1-Signaturen schlägt auf EL6-Plattformen wie CentOS6 und RHEL 6 fehl
- Problem: Eine falsche Reihenfolge von Funktionsaufrufen führt zu undefinierten Ergebnissen, anstatt dass sie fehlschlagen
- Problem: Die schreibgeschützte Sitzung wird in SDK 5 nicht unterstützt
- Problem: Die cryptoki.h-Header-Datei ist nur für Windows verfügbar

Problem: Der AES-Schlüsselumbruch in Version 3.0.0 der PKCS #11-Bibliothek validiert IVs nicht vor der Verwendung

Wenn Sie einen IV angeben, der kürzer als 8 Byte ist, wird er vor der Verwendung mit unvorhersehbaren Bytes aufgefüllt.

> **Note**
>
> Dies wirkt sich nur `C_WrapKey` mit `CKM_AES_KEY_WRAP`-Mechanismus aus.

- Auswirkung: Wenn Sie in der Version 3.0.0 der PKCS #11-Bibliothek eine IV angeben, die kürzer als 8 Byte ist, können Sie den Schlüssel möglicherweise nicht entpacken.
- Problemumgehungen:
  - Wir empfehlen Ihnen dringend, auf die Version 3.0.1 oder höher der PKCS #11-Bibliothek zu aktualisieren, die die IV-Länge während des AES-Schlüsselumbruchs korrekt durchsetzt. Ändern Sie Ihren Umbruchcode, um einen NULL-IV zu übergeben, oder geben Sie den Standard-IV von `0xA6A6A6A6A6A6A6A6` an. Weitere Informationen finden Sie im Benutzerdefinierte IVs mit nicht-konformer Länge für AES Key Wrap.
  - Wenn Sie mit der Version 3.0.0 der PKCS #11-Bibliothek Schlüssel mit einer IV von weniger als 8 Byte umgewandelt haben, wenden Sie sich bitte an uns, um Unterstützung zu erhalten.
- Auflösungsstatus: Dieses Problem wurde in Version 3.0.1 der PKCS #11-Bibliothek behoben. Um Schlüssel mit AES-Schlüsselumbruch zu umschließen, geben Sie einen IV an, der NULL oder 8 Byte lang ist.

Problem: PKCS #11 SDK 2.0.4 und frühere Versionen verwenden immer den Standard-IV von **`0xA6A6A6A6A6A6A6A6`** für AES-Schlüsselumbruch und -entpacken.

Vom Benutzer bereitgestellte IVs wurden stillschweigend ignoriert.

> **Note**
>
> Dies wirkt sich nur `C_WrapKey` mit `CKM_AES_KEY_WRAP`-Mechanismus aus.

- Auswirkung:
  - Wenn Sie PKCS #11 SDK 2.0.4 oder eine frühere Version und ein vom Benutzer bereitgestelltes IV verwendet haben, werden Ihre Schlüssel mit dem Standard-IV von `0xA6A6A6A6A6A6A6A6` umgebrochen.
  - Wenn Sie PKCS #11 SDK 3.0.0 oder höher und einen vom Benutzer bereitgestellten IV verwendet haben, werden Ihre Schlüssel mit dem vom Benutzer bereitgestellten IV umschlossen.
- Problemumgehungen:
  - Um Schlüssel zu entpacken, die mit PKCS #11 SDK 2.0.4 oder früher umschlossen sind, verwenden Sie den Standard-IV von `0xA6A6A6A6A6A6A6A6`.
  - Um Schlüssel zu entpacken, die mit PKCS #11 SDK 3.0.0 oder höher umschlossen sind, verwenden Sie den vom Benutzer bereitgestellten IV.
- Auflösungsstatus: Wir empfehlen dringend, dass Sie Ihren Umbruch- und Entpackungscode so ändern, dass ein NULL-IV übergeben wird, oder geben Sie den Standard-IV von `0xA6A6A6A6A6A6A6A6` an.

Problem: Das Attribut **`CKA_DERIVE`** wurde nicht unterstützt und nicht verarbeitet.

- Stand der Lösung: Wir haben Fehlerbehebungen implementiert, um `CKA_DERIVE` zu akzeptieren, wenn es auf `FALSE` festgelegt ist. Wenn `CKA_DERIVE` auf `TRUE` eingestellt ist, wird es erst dann unterstützt, wenn wir die Schlüsselableitungsfunktion AWS CloudHSM hinzufügen. Sie müssen Ihren Client und Ihr(e) SDK(s) auf Version 1.1.1 oder höher aktualisieren, um dieses Problem zu beheben.

Problem: Das Attribut **CKA_SENSITIVE** wurde nicht unterstützt und nicht verarbeitet.

- Stand der Lösung: Wir haben Korrekturen implementiert, mit denen das Attribut `CKA_SENSITIVE` akzeptiert und berücksichtigt wird. Sie müssen Ihren Client und Ihr(e) SDK(s) auf Version 1.1.1 oder höher aktualisieren, um dieses Problem zu beheben.

Problem: Mehrteiliges Hashing und Signatur werden nicht unterstützt.

- Auswirkung: `C_DigestUpdate` und `C_DigestFinal` werden nicht implementiert. `C_SignFinal` ist ebenfalls nicht implementiert und schlägt mit `CKR_ARGUMENTS_BAD` für einen Nicht-`NULL`-Puffer fehl.
- Problemumgehung: Hashen Sie Ihre Daten innerhalb Ihrer Anwendung und verwenden Sie sie AWS CloudHSM nur zum Signieren des Hashs.
- Stand der Lösung: Wir korrigieren den Client und die SDKs, sodass mehrteiliges Hashings korrekt implementiert wird. Aktualisierungen werden im AWS CloudHSM -Forum und auf der Seite des Versionsverlaufs bekanntgegeben.

Problem: **C_GenerateKeyPair** verarbeitet **CKA_MODULUS_BITS** oder **CKA_PUBLIC_EXPONENT** in der privaten Vorlage nicht konform zu den Standards.

- Auswirkung: `C_GenerateKeyPair` sollte `CKA_TEMPLATE_INCONSISTENT` zurückgeben, wenn die private Vorlage `CKA_MODULUS_BITS` oder `CKA_PUBLIC_EXPONENT` enthält. Stattdessen generiert es einen privaten Schlüssel, für den alle Nutzungsfelder auf `FALSE` gesetzt sind. Der Schlüssel kann nicht verwendet werden.
- Problemumgehung: Wir empfehlen, dass Ihre Anwendung die Nutzungsfeldwerte zusätzlich zum Fehlercode auswertet.
- Stand der Lösung: Wir implementieren Korrekturen, um die richtige Fehlermeldung zurückzugeben, wenn eine fehlerhafte private Schlüsselvorlage verwendet wird. Die aktualisierte PKCS#11-Bibliothek wird auf der Seite des Versionsverlaufs bekanntgegeben.

Problem: Sie können nicht mehr als 16 KB Daten mit Hash-Wert versehen.

Bei größeren Puffern werden nur die ersten 16 KB mit einem Hash-Wert versehen und zurückgegeben. Die überschüssigen Daten wurden stillschweigend ignoriert.

- Stand der Lösung: Daten mit weniger als 16 KB werden weiterhin zum Versehen mit einem Hash-Wert an HSM gesendet. Daten zwischen 16 KB und 64 KB können nun auch lokal in der Software mit einem Hash-Wert versehen werden. Der Client und die SDKs erzeugen explizit einen Fehler, wenn der Datenpuffer größer als 64 KB ist. Sie müssen Ihren Client und Ihr(e) SDK(s) auf Version 1.1.1 oder höher aktualisieren, um dieses Problem zu beheben.

Problem: Puffer für die API-Operationen **`C_Encrypt`** und **`C_Decrypt`** dürfen bei Verwendung des **`CKM_AES_GCM`**-Mechanismus 16 KB nicht überschreiten.

AWS CloudHSM unterstützt keine mehrteilige AES-GCM-Verschlüsselung.

- Auswirkung: Sie können den `CKM_AES_GCM`-Mechanismus nicht zum Verschlüsseln von Daten verwenden, die größer als 16 KB sind.
- Problemumgehung: Sie können einen alternativen Mechanismus wie`CKM_AES_CBC`,`CKM_AES_CBC_PAD`, verwenden oder Sie können Ihre Daten in Teile aufteilen und jedes Teil einzeln verschlüsseln. `AES_GCM` Wenn Sie verwenden`AES_GCM`, müssen Sie die Aufteilung Ihrer Daten und die anschließende Verschlüsselung verwalten. AWS CloudHSM führt für Sie keine mehrteilige AES-GCM-Verschlüsselung durch. Beachten Sie, dass FIPS erfordert, dass der Initialisierungsvektor (IV) auf dem HSM generiert wird. `AES-GCM` Deshalb unterscheidet sich der IV für jeden Teil Ihrer mit AES-GCM verschlüsselten Daten.
- Stand der Lösung: Wir korrigieren das SDK, sodass es explizit fehlschlägt, wenn der Datenpuffer zur groß ist. Wir geben für die `C_EncryptUpdate`- und `C_DecryptUpdate`-API-Operationen `CKR_MECHANISM_INVALID` zurück. Wir werten Alternativen zur Unterstützung größerer Puffer aus, ohne dabei eine mehrteilige Verschlüsselung verwenden zu müssen. Updates werden im AWS CloudHSM Forum und auf der Seite mit dem Versionsverlauf bekannt gegeben.

Problem: Elliptic-Curve Diffie-Hellman- (ECDH) Schlüsselableitung wird teilweise innerhalb des HSM ausgeführt.

Der private EC-Schlüssel verbleibt zu jeder Zeit innerhalb des HSM, der Prozess zur Schlüsselableitung wird aber in mehreren Schritten durchgeführt. Dies hat zur Folge, dass auf dem Client Zwischenergebnisse eines jeden Schritts verfügbar sind.

- Auswirkung: In Client SDK 3 ist der mithilfe des `CKM_ECDH1_DERIVE` Mechanismus abgeleitete Schlüssel zunächst auf dem Client verfügbar und wird dann in das HSM importiert. Ein Schlüssel-Handle wird dann an Ihre Anwendung zurückgegeben.

- Problemumgehung: Wenn Sie die SSL/TLS-Auslagerung in AWS CloudHSM implementieren, stellt diese Einschränkung möglicherweise kein Problem dar. Wenn Ihre Anwendung vorschreibt, dass Ihr Schlüssel zu jeder Zeit innerhalb einer FIPS-Begrenzung verbleibt, sollten Sie mit einem alternativen Protokoll ohne erforderliche ECDH-Schlüsselableitung arbeiten.
- Stand der Lösung: Wir arbeiten derzeit an der Möglichkeit, die ECDH-Schlüsselableitung vollständig innerhalb des HSM auszuführen. Die aktualisierte Implementierung wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: Die Überprüfung von secp256k1-Signaturen schlägt auf EL6-Plattformen wie CentOS6 und RHEL 6 fehl

Der Grund hierfür ist, dass die PKCS #11-Bibliothek von CloudHSM während der Initialisierung des Überprüfungsvorgangs einen Netzwerkaufruf vermeidet, indem sie OpenSSL zur Überprüfung von EC-Kurvendaten verwendet. Da secp256k1 durch das Standard-OpenSSL-Paket auf EL6-Plattformen nicht unterstützt wird, schlägt die Initialisierung fehl.

- Auswirkung: Die secp256k1-Signaturüberprüfung schlägt auf EL6-Plattformen fehl. Der Überprüfungsaufruf schlägt fehl und es wird eine `CKR_HOST_MEMORY`-Fehlermeldung angezeigt.
- Problemumgehung: Wir empfehlen die Verwendung von Amazon Linux 1 oder einer beliebigen EL7-Plattform, wenn Ihre PKCS #11-Anwendung secp256k1-Signaturen überprüfen soll. Alternativ können Sie ein Upgrade auf eine Version des OpenSSL-Pakets durchführen, das die secp256k1-Kurve unterstützt.
- Stand der Lösung: Wir implementieren zurzeit Korrekturen, die es ermöglichen, auf das HSM zurückzugreifen, wenn die lokale Kurvenvalidierung nicht verfügbar ist. Die aktualisierte PKCS #11-Bibliothek wird auf der Seite des Versionsverlaufs bekanntgegeben.

Problem: Eine falsche Reihenfolge von Funktionsaufrufen führt zu undefinierten Ergebnissen, anstatt dass sie fehlschlagen

- Auswirkung: Wenn Sie eine falsche Reihenfolge von Funktionen aufrufen, ist das Endergebnis falsch, obwohl die einzelnen Funktionsaufrufen erfolgreich sind. Beispielsweise stimmen entschlüsselte Daten möglicherweise nicht mit dem ursprünglichen Klartext überein oder Signaturen lassen sich möglicherweise nicht verifizieren. Dieses Problem betrifft sowohl einteilige als auch mehrteilige Operationen.

  Beispiele für falsche Funktionssequenzen:

- `C_EncryptInit`/`C_EncryptUpdate` gefolgt von `C_Encrypt`
- `C_DecryptInit`/`C_DecryptUpdate` gefolgt von `C_Decrypt`
- `C_SignInit`/`C_SignUpdate` gefolgt von `C_Sign`
- `C_VerifyInit`/`C_VerifyUpdate` gefolgt von `C_Verify`
- `C_FindObjectsInit` gefolgt von `C_FindObjectsInit`

- Umgehung: Ihre Anwendung sollte gemäß der PKCS #11 -Spezifikation die richtige Reihenfolge von Funktionsaufrufen sowohl für einteilige als auch für mehrteilige Operationen verwenden. Ihre Anwendung sollte sich nicht darauf verlassen, dass die CloudHSM PKCS #11-Bibliothek unter diesen Umständen einen Fehler zurückgibt.

Problem: Die schreibgeschützte Sitzung wird in SDK 5 nicht unterstützt

- Problem: SDK 5 unterstützt nicht das Öffnen von Nur-Lesesitzungen mit `C_OpenSession`.
- Auswirkung: Wenn Sie versuchen, einen Aufruf an `C_OpenSession` ohne Angabe von `CKF_RW_SESSION` zu tätigen, schlägt der Aufruf mit der folgenden Fehlermeldung `CKR_FUNCTION_FAILED` fehl.
- Problemumgehung: Wenn Sie eine Sitzung öffnen, müssen Sie die `CKF_SERIAL_SESSION | CKF_RW_SESSION`-Flags an den `C_OpenSession`-Funktionsaufruf übergeben.

Problem: Die **`cryptoki.h`**-Header-Datei ist nur für Windows verfügbar

- Problem: Bei den AWS CloudHSM Client-SDK 5-Versionen 5.0.0 bis 5.4.0 unter Linux ist die Header-Datei nur mit Windows-Betriebssystemen kompatibel. `/opt/cloudhsm/include/pkcs11/cryptoki.h`
- Auswirkung: Unter Linux-basierten Betriebssystemen können Probleme auftreten, wenn Sie versuchen, diese Header-Datei in Ihre Anwendung aufzunehmen.
- Lösungsstatus: Führen Sie ein Upgrade auf AWS CloudHSM Client SDK 5 Version 5.4.1 oder höher durch, die eine Linux-kompatible Version dieser Header-Datei enthält.

Bekannte Probleme für das JCE-SDK

Themen

Problem: Bei der Arbeit mit asymmetrischen Schlüsselpaaren wird die belegte Schlüsselkapazität auch dann angezeigt, wenn Sie Schlüssel nicht explizit erstellen oder importieren.

- Auswirkung: Dieses Problem kann dazu führen, dass Ihre HSM unerwartet nicht mehr genügend Schlüsselspeicher haben. Dies tritt auf, wenn Ihre Anwendung ein JCE-Standardschlüsselobjekt für Kryptooperationen anstelle eines `CaviumKey`-Objekts verwendet. Wenn Sie ein JCE-Standardschlüsselobjekt verwenden, importiert `CaviumProvider` diesen Schlüssel implizit in das HSM, da ein Sitzungsschlüssel diesen Schlüssel erst löscht, wenn die Anwendung beendet wird. Dies hat zur Folge, dass Schlüssel erstellt werden, während die Anwendung ausgeführt wird. Dies kann dazu führen, dass Ihren HSM kein freier Schlüsselspeicher mehr zur Verfügung steht und Ihre Anwendung einfriert.
- Problemumgehung: Wenn Sie die Klasse `CaviumSignature`, `CaviumCipher`, `CaviumMac` oder `CaviumKeyAgreement` verwenden, sollten Sie den Schlüssel als `CaviumKey` und nicht als JCE-Standardschlüsselobjekt bereitstellen.

  Sie können mithilfe der `ImportKey`-Klasse einen normalen Schlüssel manuell in einen `CaviumKey` umwandeln und den Schlüssel dann nach Abschluss des Vorgangs manuell löschen.
- Stand der Lösung: Wir aktualisieren den `CaviumProvider`, um implizite Importe ordnungsgemäß zu verwalten. Diese Fehlerbehebung wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: Das JCE KeyStore ist schreibgeschützt

- Auswirkungen: Derzeit können im JCE KeyStore nur Objekttypen gespeichert werden, die von HSM unterstützt werden. Insbesondere ist es nicht möglich, Zertifikate im Schlüsselspeicher zu speichern. Dadurch wird die Interoperabilität mit Tools wie jarsigner, die im Schlüsselspeicher ein Zertifikat erwarten, verhindert.
- Problemumgehung: Sie können Änderungen am Code vornehmen, sodass Zertifikate anstatt aus dem Schlüsselspeicher aus lokalen Dateien oder aus einem S3-Bucket-Speicherort geladen werden.
- Stand der Lösung: Wir arbeiten daran, Unterstützung für die Zertifikatspeicherung im Schlüsselspeicher hinzuzufügen. Diese Funktion wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: Puffer für die AES-GCM-Verschlüsselung dürfen nicht größer sein als 16.000 Byte.

Es wird keine mehrteilige AES-GCM-Verschlüsselung unterstützt.

- Auswirkung: Sie können AES-GCM nicht zum Verschlüsseln von Daten verwenden, die größer als 16.000 Byte sind.
- Problemumgehung: Sie können einen alternativen Mechanismus verwenden, wie beispielsweise AES-CBC, oder Sie können Ihre Daten unterteilen und jeden Teil einzeln verschlüsseln. Wenn Sie die Daten unterteilen, müssen Sie den unterteilten Verschlüsselungstext und dessen Entschlüsselung verwalten. Da FIPS erfordert, dass der Initialisierungsvektor (IV) für AES-GCM im HSM erstellt wird, ist der IV für die einzelnen AES-GCM-verschlüsselte Datensegmente jeweils anders.
- Stand der Lösung: Wir korrigieren das SDK, sodass es explizit fehlschlägt, wenn der Datenpuffer zur groß ist. Wir werten Alternativen zur Unterstützung größerer Puffer aus, ohne eine mehrteilige Verschlüsselung verwenden zu müssen. Aktualisierungen werden im AWS CloudHSM -Forum und auf der Seite des Versionsverlaufs bekanntgegeben.

Problem: Elliptic-Curve Diffie-Hellman- (ECDH) Schlüsselableitung wird teilweise innerhalb des HSM ausgeführt.

Der private EC-Schlüssel verbleibt zu jeder Zeit innerhalb des HSM, der Prozess zur Schlüsselableitung wird aber in mehreren Schritten durchgeführt. Dies hat zur Folge,

dass auf dem Client Zwischenergebnisse eines jeden Schritts verfügbar sind. Ein ECDH-Schlüsselableitungsbeispiel ist in den Java-Codebeispielenverfügbar.

- Auswirkung: Das Client SDK 3 erweitert das JCE um ECDH-Funktionalität. Wenn Sie die `KeyAgreement` Klasse verwenden, um eine abzuleiten SecretKey, ist sie zunächst auf dem Client verfügbar und wird dann in das HSM importiert. Ein Schlüssel-Handle wird dann an Ihre Anwendung zurückgegeben.
- Problemumgehung: Wenn Sie SSL/TLS Offload in implementieren AWS CloudHSM, stellt diese Einschränkung möglicherweise kein Problem dar. Wenn Ihre Anwendung vorschreibt, dass Ihr Schlüssel zu jeder Zeit innerhalb einer FIPS-Begrenzung verbleibt, sollten Sie mit einem alternativen Protokoll ohne erforderliche ECDH-Schlüsselableitung arbeiten.
- Stand der Lösung: Wir arbeiten derzeit an der Möglichkeit, die ECDH-Schlüsselableitung vollständig innerhalb des HSM auszuführen. Wenn verfügbar, werden wir die aktualisierte Implementierung auf der Seite des Versionsverlaufs bekanntgeben.

Problem: KeyGenerator und interpretiert den Schlüsselgrößenparameter KeyAttribute fälschlicherweise als Anzahl von Bytes statt als Bits

Beim Generieren eines Schlüssels mithilfe der `init` Funktion der KeyGenerator Klasse oder des `SIZE` Attributs der AWS CloudHSM KeyAttribute Enumeration erwartet die API fälschlicherweise, dass das Argument die Anzahl der Schlüsselbytes ist, obwohl es stattdessen die Anzahl der Schlüsselbits sein sollte.

- Auswirkung: Die Client-SDK-Versionen 5.4.0 bis 5.4.2 erwarten fälschlicherweise, dass die Schlüsselgröße den angegebenen APIs in Byte zur Verfügung gestellt wird.
- Umgehung: Wenn Sie die Client-SDK-Versionen 5.4.0 bis 5.4.2 verwenden, konvertieren Sie die Schlüsselgröße von Bits in Byte, bevor Sie die KeyGenerator Klasse oder die KeyAttribute Enumeration verwenden, um Schlüssel mithilfe des AWS CloudHSM JCE-Anbieters zu generieren.
- Lösungsstatus: Führen Sie ein Upgrade Ihrer Client-SDK-Version auf Version 5.5.0 oder höher durch. Dies beinhaltet einen Fix, mit dem Schlüsselgrößen in Bits korrekt erwartet werden, wenn Sie die KeyGenerator Klasse oder die Enumeration zum Generieren von Schlüsseln verwenden. KeyAttribute

Problem: Das Client-SDK 5 gibt die Warnung „Es ist ein illegaler Reflective Access-Vorgang aufgetreten“ aus

Bei Verwendung von Client-SDK 5 mit Java 11 gibt CloudHSM die folgende Java-Warnung aus:

```
```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by
 com.amazonaws.cloudhsm.jce.provider.CloudHsmKeyStore (file:/opt/cloudhsm/java/
cloudhsm-jce-5.6.0.jar) to field java.security .KeyStore.keyStoreSpi
WARNING: Please consider reporting this to the maintainers of
 com.amazonaws.cloudhsm.jce.provider.CloudHsmKeyStore
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
 access operations
WARNING: All illegal access operations will be denied in a future release
```
```

Diese Warnungen haben keine Auswirkungen. Wir sind uns dieses Problems bewusst und arbeiten daran, es zu lösen. Es ist weder eine Lösung noch eine Problemumgehung erforderlich.

Problem: Der JCE-Sitzungspool ist erschöpft

Auswirkung: Sie können möglicherweise keine Operationen in JCE ausführen, wenn die folgende Meldung angezeigt wird:

```
com.amazonaws.cloudhsm.jce.jni.exception.InternalException: There are too many
 operations
happening at the same time: Reached max number of sessions in session pool: 1000
```

Problemumgehungen:

- Starten Sie Ihre JCE-Anwendung neu, falls Probleme auftreten.
- Wenn Sie einen Vorgang ausführen, müssen Sie den JCE-Vorgang möglicherweise abschließen, bevor Sie den Bezug zu dem Vorgang verlieren.

> **Note**
>
> Je nach Vorgang ist möglicherweise eine Abschlussmethode erforderlich.

Operation	Vervollständigungsmethode
Verschlüsselungsverfahren	`doFinal()` im Verschlüsselungs- oder Entschlüsselungsmodus `wrap()` im Wrap-Modus `unwrap()` im Unwrap-Modus
KeyAgreement	`generateSecret()` oder `generateSecret(String)`
KeyPairGenerator	`generateKeyPair()` , `genKeyPair()` oder `reset()`
KeyStore	Keine Methode erforderlich
MAC	`doFinal()` oder `reset()`
MessageDigest	`digest()` oder `reset()`
SecretKeyFactory	Keine Methode erforderlich
SecureRandom	Keine Methode erforderlich
Signatur	`sign()` im Signiermodus `verify()` im Überprüfungsmodus

Lösungsstatus: Wir haben dieses Problem in Client SDK 5.9.0 und höher behoben. Um dieses Problem zu beheben, aktualisieren Sie Ihr Client-SDK auf eine dieser Versionen.

Problem: Speicherverlust im Client-SDK 5 bei GetKey-Vorgängen

- Auswirkung: Bei der `getKey` API-Operation ist in JCE in den Client-SDK-Versionen 5.10.0 und früher ein Speicherverlust aufgetreten. Wenn Sie die `getKey` API in Ihrer Anwendung mehrfach verwenden, führt dies zu einem erhöhten Speicherwachstum und damit zu einer Erhöhung des

Speicherbedarfs in Ihrer Anwendung. Im Laufe der Zeit kann dies zu Drosselungsfehlern führen oder einen Neustart der Anwendung erfordern.

- Problemumgehung: Wir empfehlen ein Upgrade auf Client SDK 5.11.0. Wenn dies nicht möglich ist, empfehlen wir, die `getKey` API in Ihrer Anwendung nicht mehrmals aufzurufen. Verwenden Sie stattdessen den zuvor zurückgegebenen Schlüssel aus der vorherigen `getKey` Operation so oft wie möglich wieder.
- Lösungsstatus: Aktualisieren Sie Ihre Client-SDK-Version auf 5.11.0 oder höher, was eine Lösung für dieses Problem beinhaltet.

Bekannte Probleme für die OpenSSL Dynamic Engine

Dies sind die bekannten Probleme für die OpenSSL Dynamic Engine

Themen

Problem: Sie können AWS CloudHSM OpenSSL Dynamic Engine nicht auf RHEL 6 und CentOS6 installieren

- Auswirkung: Die OpenSSL Dynamic Engine unterstützt nur OpenSSL 1.0.2[f+]. Standardmäßig werden RHEL 6 und CentOS 6 mit OpenSSL 1.0.1 ausgeliefert.
- Problemumgehung: Aktualisieren Sie die OpenSSL-Bibliothek auf RHEL 6 und CentOS 6 auf Version 1.0.2[f+].

Problem: Standardmäßig wird nur die RSA-Auslagerung zum HSM unterstützt.

- Auswirkung: Zur Maximierung der Leistung ist der SDK nicht für die Auslagerung zusätzlicher Funktionen, wie z. B. Zufallszahlengenerierung oder EC-DH-Operationen, konfiguriert.
- Problemumgehung: Bitte kontaktieren Sie uns mittels eines Supportfalls, wenn Sie zusätzliche Operationen auslagern möchten.
- Stand der Lösung: Wir arbeiten daran, den SDK um Unterstützung für das Konfigurieren von Auslagerungsoptionen über eine Konfigurationsdatei zu erweitern. Die Aktualisierung wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: RSA-Ver- und Entschlüsselung mit OAEP-Padding unter Verwendung eines Schlüssels im HSM wird nicht unterstützt.

- Auswirkung: Jeder Aufruf zur RSA-Verschlüsselung und -Entschlüsselung mit OAEP-Padding schlägt mit einem Fehler fehl. divide-by-zero Der Grund besteht darin, dass die dynamische OpenSSL-Engine die Operation lokal unter Verwendung der Fake-PEM-Datei aufruft, statt die Operation in das HSM auszulagern.
- Problemumgehung: Sie können diesen Vorgang in der PKCS #11-Bibliothek oder in der JCE-Anbieter ausführen.
- Stand der Lösung: Wird fügen eine Unterstützung des SDK hinzu, damit diese Operation ordnungsgemäß ausgelagert wird. Die Aktualisierung wird auf der Seite des Versionsverlaufs bekanntgegeben, sobald sie verfügbar ist.

Problem: Nur die Generierung privaten Schlüssel der RSA- und ECC-Schlüssel wird in das HSM ausgelagert.

Für jeden anderen Schlüsseltyp wird die AWS CloudHSM OpenSSL-Engine nicht für die Anrufverarbeitung verwendet. Stattdessen wird die lokale OpenSSL-Engine verwendet. Dies generiert einen Schlüssel lokal in der Software.

- Auswirkungen: Da das Failover stillschweigend erfolgt, gibt es keinen Hinweis darauf, dass Sie keinen Schlüssel erhalten haben, der sicher im HSM generiert wurde. Sie sehen einen Ausgabeablauf, der die Zeichenkette "`.....++++++`" enthält, wenn der Schlüssel lokal von OpenSSL in der Software generiert wurde. Diese Abfolge ist nicht vorhanden, wenn die Operation in das HSM ausgelagert wurde. Da der Schlüssel nicht im HSM erstellt oder gespeichert, steht er für die zukünftige Nutzung nicht zur Verfügung.

- Problemumgehung: Verwenden Sie die OpenSSL-Engine nur für Schlüsseltypen, die sie unterstützt. Verwenden Sie für alle anderen Schlüsseltypen PKCS #11 oder JCE in Anwendungen oder `key_mgmt_util` in der CLI.

Problem: Sie können OpenSSL Dynamic Engine für Client-SDK 3 nicht auf RHEL 8, CentOS 8 oder Ubuntu 18.04 LTS installieren

- Auswirkung: RHEL 8, CentOS 8 und Ubuntu 18.04 LTS liefern standardmäßig eine Version von OpenSSL, die nicht mit OpenSSL Dynamic Engine für Client-SDK 3 kompatibel ist.
- Problemumgehung: Verwenden Sie eine Linux-Plattform, die OpenSSL Dynamic Engine unterstützt. Weitere Informationen zu unterstützten Plattformen finden Sie unter Unterstützte Plattformen.
- Auflösungsstatus: AWS CloudHSM unterstützt diese Plattformen mit OpenSSL Dynamic Engine für Client SDK 5. Weitere Informationen finden Sie unter Unterstützte Plattformen und OpenSSL Dynamic Engine.

Problem: SHA-1 Sign and Verify ist auf RHEL 9 (9.2+) als veraltet markiert

- Auswirkung: Die Verwendung des SHA-1 Message Digest für kryptografische Zwecke ist in RHEL 9 (9.2+) veraltet. Infolgedessen schlagen Signier- und Verifizierungsvorgänge mit SHA-1 unter Verwendung der OpenSSL Dynamic Engine fehl.
- Problemumgehung: Wenn Ihr Szenario die Verwendung von SHA-1 zum Signieren/Überprüfen vorhandener kryptografischer Signaturen oder von Drittanbietern erfordert, finden Sie weitere Informationen unter Enhancing RHEL Security: Understanding SHA-1 Deprecation in den Versionshinweisen zu RHEL 9 (9.2+) und RHEL 9 (9.2+).

Problem: AWS CloudHSM OpenSSL Dynamic Engine ist nicht mit dem FIPS-Anbieter für OpenSSL v3.x kompatibel

- Auswirkung: Sie erhalten eine Fehlermeldung, wenn Sie versuchen, die AWS CloudHSM OpenSSL Dynamic Engine zu verwenden, wenn der FIPS-Anbieter für OpenSSL-Versionen 3.x aktiviert ist.
- Umgehung: Um die AWS CloudHSM OpenSSL Dynamic Engine mit OpenSSL Version 3.x zu verwenden, stellen Sie sicher, dass der „Standard“ -Anbieter konfiguriert ist. Lesen Sie mehr über den Standardanbieter auf der OpenSSL-Website.

Bekannte Probleme mit Amazon-EC2-Instances, auf denen Amazon Linux 2 ausgeführt wird

Problem: Amazon Linux 2 Version 2018.07 verwendet ein aktualisiertes **`ncurses`** Paket (Version 6), das derzeit nicht mit den SDKs kompatibel ist AWS CloudHSM

Beim Ausführen von AWS CloudHSMcloudhsm_mgmt_util oder key_mgmt_util wird der folgende Fehler angezeigt:

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util: error while loading shared libraries:
 libncurses.so.5: cannot open shared object file: No such file or directory
```

- Auswirkung: Instances, die auf Amazon Linux 2 Version 2018.07 ausgeführt werden, können nicht alle AWS CloudHSM Dienstprogramme verwenden.
- Problemumgehung: Geben Sie den folgenden Befehl an Ihre Amazon-Linux-2-EC2-Instances aus, um das unterstützte `ncurses`-Paket (Version 5) zu installieren:

  ```
  sudo yum update && yum install ncurses-compat-libs
  ```

- Stand der Lösung: Dieses Problem wurde in AWS CloudHSM -Client Version 1.1.2 behoben. Sie müssen auf diesen Client upgraden, um das Problem zu beheben.

Bekannte Probleme bei der Integration von Drittanbieter-Anwendungen

Problem: Client-SDK 3 unterstützt nicht das Setzen des PKCS #11-Attributs **`CKA_MODIFIABLE`** durch Oracle während der Hauptschlüsselerzeugung

Dieses Limit ist in der PKCS #11-Bibliothek definiert. Weitere Informationen finden Sie unter Anmerkung 1 zu Unterstützte PKCS #11-Attribute.

- Auswirkung: Die Erstellung des Oracle-Masterschlüssels schlägt fehl.
- Problemumgehung: Setzen Sie die spezielle Umgebungsvariable `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` auf TRUE, wenn Sie einen neuen Masterschlüssel erstellen. Diese Umgebungsvariable wird nur für die Generierung des Masterschlüssels benötigt. Sie müssen diese Umgebungsvariable für nichts anderes verwenden. Sie würden diese Variable beispielsweise für den ersten Masterschlüssel verwenden, den Sie

erstellen, und würden dann diese Umgebungsvariable nur erneut verwenden, wenn Sie Ihre Masterschlüssel-Edition rotieren möchten. Weitere Informationen finden Sie unter Generieren des Oracle TDE-Master-Verschlüsselungsschlüssels.

- Behebungsstatus: Wir verbessern die HSM-Firmware, um das Attribut CKA_MODIFIABLE vollständig zu unterstützen. Updates werden im AWS CloudHSM Forum und auf der Seite mit dem Versionsverlauf angekündigt

Fehler bei der Schlüsselsynchronisierung des Client-SDK 3

Wenn die clientseitige Synchronisation fehlschlägt, versucht AWS CloudHSM in Client-SDK 3 nach besten Kräften, alle unerwünschten Schlüssel zu bereinigen, die möglicherweise erstellt wurden (und jetzt unerwünscht sind). Bei diesem Vorgang wird unerwünschtes Schlüsselmaterial sofort entfernt oder unerwünschtes Material für ein späteres Entfernen markiert. In beiden Fällen sind für die Lösung keine Maßnahmen Ihrerseits erforderlich. In den seltenen Fällen, in denen AWS CloudHSM unerwünschtes Schlüsselmaterial nicht entfernen und markieren kann, müssen Sie das Schlüsselmaterial löschen.

Problem: Sie versuchen, einen Token-Schlüssel zu generieren, zu importieren oder zu entpacken, und es werden Fehler angezeigt, die darauf hindeuten, dass Tombstone nicht korrekt ausgeführt wurde.

```
2018-12-24T18:28:54Z liquidSecurity ERR: print_node_ts_status:
[create_object_min_nodes]Key: 264617 failed to tombstone on node:1
```

Ursache: AWS CloudHSM Das Entfernen und Markieren von unerwünschtem Schlüsselmaterial ist fehlgeschlagen.

Lösung: Ein HSM in Ihrem Cluster enthält unerwünschtes Schlüsselmaterial, das nicht als unerwünscht markiert ist. Sie müssen das Schlüsselmaterial manuell entfernen. Um unerwünschtes Schlüsselmaterial manuell zu löschen, verwenden Sie key_mgmt_util (KMU) oder eine API aus der PKCS #11-Bibliothek oder dem JCE-Anbieter. Weitere Informationen hierzu finden Sie unter deleteKey oder Client-SDKs.

Um Token-Schlüssel langlebiger zu machen, schlagen bei AWS CloudHSM Schlüsselerstellungsvorgänge fehl, die bei der Mindestanzahl von HSMs, die in den Einstellungen für die clientseitige Synchronisation angegeben ist, nicht erfolgreich sind. Weitere Informationen finden Sie unter Schlüsselsynchronisierung in AWS CloudHSM.

Client-SDK 3: Überprüfen Sie die HSM-Leistung mit dem pkpspeed-Tool

In diesem Thema wird beschrieben, wie die HSM-Leistung mit dem Client-SDK 3 überprüft wird.

Um die Leistung der HSMs in Ihrem AWS CloudHSM-Cluster zu überprüfen, können Sie das Tool pkpspeed (Linux) oder pkpspeed_blocking (Windows) verwenden, das im Client-SDK 3 enthalten ist. Das Tool pkpspeed wird unter idealen Bedingungen ausgeführt und ruft das HSM direkt auf, um Operationen auszuführen, ohne ein SDK wie PKCS11 verwenden zu müssen. Wir empfehlen, Ihre Anwendung unabhängig voneinander unter Last zu testen, um Ihre Skalierungsanforderungen zu ermitteln. Es wird nicht empfohlen, die folgenden Tests auszuführen: Random (I), ModExp (R) und EC Point Mul (Y).

Weitere Informationen zum Installieren des Clients auf einer Linux EC2 Instance finden Sie unter Den AWS CloudHSM Client installieren und konfigurieren (Linux). Weitere Informationen zum Installieren des Clients auf einer Windows-Instance finden Sie unter Installieren und Konfigurieren des AWS CloudHSM-Clients (Windows).

Nachdem Sie den AWS CloudHSM-Client installiert und konfiguriert haben, führen Sie den folgenden Befehl aus, um ihn zu starten.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Wenn Sie die Client-Software bereits installiert haben, müssen Sie möglicherweise die neueste Version herunterladen und installieren, um „pkpspeed“ abzurufen. Sie finden das Tool pkpspeed unter Linux in `/opt/cloudhsm/bin/pkpspeed` und unter Windows in `C:\Program Files \Amazon\CloudHSM\`.

Um pkpspeed zu verwenden, führen Sie den Befehl pkpspeed oder pkpspeed_blocking.exe aus. Geben Sie dabei den Benutzernamen und das Passwort eines Crypto-Benutzers (CU) im HSM an. Legen Sie dann die Optionen unter Beachtung der folgenden Empfehlungen fest.

Empfehlungen testen

- Wählen Sie zum Testen der Leistung von RSA-Signatur- und Prüfoperationen die Verschlüsselung `RSA_CRT` unter Linux oder Option B unter Windows. Wählen Sie nicht `RSA` (Option A unter Windows). Die Verschlüsselungen sind zwar gleichwertig, `RSA_CRT` ist jedoch leistungsoptimiert.
- Beginnen Sie mit einer geringen Anzahl von Threads. Zum Testen der AES-Leistung genügt in der Regel ein Thread, um eine maximale Leistung zu demonstrieren. Zum Testen der RSA-Leistung (`RSA_CRT`) sind drei oder vier Threads in der Regel ausreichend.

Konfigurierbare Optionen für das Tool pkpspeed

- FIPS-Modus: AWS CloudHSM befindet sich immer im FIPS-Modus (Einzelheiten finden Sie in den häufig gestellten Fragen zu AWS CloudHSM). Dies kann überprüft werden, indem Sie die im AWS CloudHSM-Benutzerhandbuch dokumentierten CLI-Tools verwenden und den getHSMInfo -Befehl ausführen, der den Status des FIPS-Modus anzeigt.
- Testtyp (blockierend versus nicht blockierend): Dies gibt an, wie Operationen in Thread-Form ausgeführt werden. Sie werden höchstwahrscheinlich bessere Zahlen erhalten, wenn Sie nicht blockierend verwenden. Das liegt daran, dass sie Threads und Parallelität nutzen.
- Anzahl der Threads: Anzahl der Threads, mit denen der Test ausgeführt werden soll.
- Zeit in Sekunden für die Ausführung des Tests (max. = 600): pkpspeed erzeugt Ergebnisse, die in „Operationen/Sekunde“ gemessen werden, und meldet diesen Wert für jede Sekunde, in der der Test ausgeführt wird. Wenn der Test beispielsweise 5 Sekunden lang ausgeführt wird, kann die Ausgabe wie die folgenden Beispielwerte aussehen:
  - `OPERATIONS/second 821/1`
  - `OPERATIONS/second 833/1`
  - `OPERATIONS/second 845/1`
  - `OPERATIONS/second 835/1`
  - `OPERATIONS/second 837/1`

Tests, die mit dem Tool pkpspeed ausgeführt werden können

- AES GCM: Testet die Verschlüsselung im AES-GCM-Modus.
- Basic 3DES CBC: Testet die Verschlüsselung im 3DES CBC-Modus. Eine bevorstehende Änderung finden Sie im Hinweis 1 unten.

- Basic AES: Testet die AES-CBC/ECB-Verschlüsselung.
- Digest: Testet den Hash-Digest.
- ECDSA-Zeichen: Testet das ECDSA-Zeichen.
- ECDSA-Verifizierung: Testet ECDSA-Verifizierung.
- FIPS Random: Testet die Generierung einer FIPS-konformen Zufallszahl (Hinweis: Diese kann nur im Sperrmodus verwendet werden).
- HMAC: Testet HMAC.
- Random: Dieser Test ist nicht relevant, da wir FIPS 140-2 HSMs verwenden.
- RSA non-CRT versus RSA_CRT: Testet RSA-Signier- und Verifizierungsvorgänge.
- RSA OAEP Enc: Testet die RSA OAEP-Verschlüsselung.
- RSA OAEP Dec: Testet die RSA OAEP-Entschlüsselung.
- RSA Private Dec Non-CRT: Testet die Verschlüsselung mit privatem RSA-Schlüssel (nicht optimiert).
- RSA Private Key dec CRT: Testet die Verschlüsselung mit privatem RSA-Schlüssel (optimiert).
- RSA PSS Sign: Testet das RSA PSS-Zeichen.
- RSA PSS Verify: Teste RSA PSS verifizieren.
- RSA Public Key Enc: Testet die Verschlüsselung mit öffentlichem RSA-Schlüssel.

Bei der Verschlüsselung mit öffentlichem RSA-Schlüssel, der Entschlüsselung mit privatem RSA-Schlüssel (nicht CRT) und der Entschlüsselung mit privatem RSA-Schlüssel (CRT) wird der Benutzer außerdem aufgefordert, die folgenden Fragen zu beantworten:

```
Do you want to use static key [y/n]
```

Wenn y eingegeben wird, wird ein vorberechneter Schlüssel in das HSM importiert.

Wenn n eingegeben wird, wird ein neuer Schlüssel generiert.

[1] Aus Gründen der FIPS-Konformität gemäß den NIST-Richtlinien nach 2023 nicht zulässig. Details dazu finden Sie unter FIPS-140-Konformität: Mechanismus 2024 nicht mehr unterstützt.

Beispiele

Die folgenden Beispiele zeigen die Optionen, die Sie mit pkpspeed (Linux) oder pkpspeed_blocking (Windows) zum Testen der HSM-Leistung für RSA- und AES-Operationen auswählen können.

Example – Verwenden von pkpspeed zum Testen der RSA-Leistung

Sie können dieses Beispiel unter Windows, Linux und kompatiblen Betriebssystemen ausführen.

Linux

Verwenden Sie diese Anweisungen für Linux und kompatible Betriebssysteme.

```
/opt/cloudhsm/bin/pkpspeed -s CU user name -p password

SDK Version: 2.03

        Available Ciphers:
                AES_128
                AES_256
                3DES
                RSA  (non-CRT. modulus size can be 2048/3072)
                RSA_CRT (same as RSA)
For RSA, Exponent will be 65537

Current FIPS mode is: 00002
Enter the number of thread [1-10]: 3
Enter the cipher: RSA_CRT
Enter modulus length: 2048
Enter time duration in Secs: 60
Starting non-blocking speed test using data length of 245 bytes...
[Test duration is 60 seconds]

Do you want to use static key[y/n] (Make sure that KEK is available)?n
```

Windows

```
c:\Program Files\Amazon\CloudHSM>pkpspeed_blocking.exe -s CU user name -p password

Please select the test you want to run

RSA non-CRT------------------->A
RSA CRT----------------------->B
Basic 3DES CBC---------------->C
Basic AES--------------------->D
FIPS Random------------------->H
Random------------------------>I
AES GCM ---------------------->K
```

```
eXit------------------------>X
B

Running 4 threads for 25 sec

Enter mod size(2048/3072):2048
Do you want to use Token key[y/n]n
Do you want to use static key[y/n] (Make sure that KEK is available)?  n
OPERATIONS/second                821/1
OPERATIONS/second                833/1
OPERATIONS/second                845/1
OPERATIONS/second                835/1
OPERATIONS/second                837/1
OPERATIONS/second                836/1
OPERATIONS/second                837/1
OPERATIONS/second                849/1
OPERATIONS/second                841/1
OPERATIONS/second                856/1
OPERATIONS/second                841/1
OPERATIONS/second                847/1
OPERATIONS/second                838/1
OPERATIONS/second                843/1
OPERATIONS/second                852/1
OPERATIONS/second                837/
```

Example – Verwenden von pkpspeed zum Testen der AES-Leistung

Linux

Verwenden Sie diese Anweisungen für Linux und kompatible Betriebssysteme.

```
/opt/cloudhsm/bin/pkpspeed -s <CU user name> -p <password>

SDK Version: 2.03

        Available Ciphers:
                AES_128
                AES_256
                3DES
                RSA  (non-CRT. modulus size can be 2048/3072)
                RSA_CRT (same as RSA)
For RSA, Exponent will be 65537
```

```
Current FIPS mode is: 00000002
Enter the number of thread [1-10]: 1
Enter the cipher: AES_256
Enter the data size [1-16200]: 8192
Enter time duration in Secs: 60
Starting non-blocking speed test using data length of 8192 bytes...
```

Windows

```
c:\Program Files\Amazon\CloudHSM>pkpspeed_blocking.exe -s CU user name -p password
login as USER
Initializing Cfm2 library
        SDK Version: 2.03

 Current FIPS mode is: 00000002
Please enter the number of threads [MAX=400] : 1
Please enter the time in seconds to run the test [MAX=600]: 20


Please select the test you want to run

RSA non-CRT------------------->A
RSA CRT----------------------->B
Basic 3DES CBC---------------->C
Basic AES--------------------->D
FIPS Random------------------->H
Random------------------------>I
AES GCM ---------------------->K

eXit------------------------->X
D

Running 1 threads for 20 sec

Enter the key size(128/192/256):256
Enter the size of the packet in bytes[1-16200]:8192
OPERATIONS/second                9/1
OPERATIONS/second                10/1
OPERATIONS/second                11/1
OPERATIONS/second                10/1
OPERATIONS/second                10/1
OPERATIONS/second                10/...
```

Der Client-SDK 5-Benutzer enthält inkonsistente Werte

Der `user list`-Befehl gibt eine Liste aller Benutzer und Benutzereigenschaften in Ihrem Cluster zurück. Wenn eine der Eigenschaften eines Benutzers den Wert „inkonsistent“ hat, wird dieser Benutzer nicht in Ihrem gesamten Cluster synchronisiert. Das bedeutet, dass der Benutzer mit unterschiedlichen Eigenschaften auf verschiedenen HSMs im Cluster existiert. Je nachdem, welche Eigenschaft inkonsistent ist, können unterschiedliche Reparaturschritte unternommen werden.

Die folgende Tabelle enthält Schritte zum Beheben von Inkonsistenzen für einen einzelnen Benutzer. Wenn ein einzelner Benutzer mehrere Inkonsistenzen aufweist, lösen Sie diese, indem Sie die folgenden Schritte von oben nach unten ausführen. Wenn mehrere Benutzer Inkonsistenzen aufweisen, arbeiten Sie diese Liste für jeden Benutzer durch und beheben Sie die Inkonsistenzen für diesen Benutzer vollständig, bevor Sie mit dem nächsten fortfahren.

> **Note**
>
> Um diese Schritte durchzuführen, sollten Sie idealerweise als Administrator angemeldet sein. Wenn Ihr Administratorkonto nicht konsistent ist, gehen Sie wie folgt vor, melden Sie sich beim Administrator an und wiederholen Sie die Schritte, bis alle Eigenschaften konsistent sind. Sobald Ihr Administratorkonto konsistent ist, können Sie diesen Administrator verwenden, um andere Benutzer im Cluster zu synchronisieren.

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
Die „Rolle“ des Benutzers ist „inkonsistent“	`{ "username": "test_user", "role": "inconsistent ", "locked": "false", "mfa": [], "cluster-coverage": "full"`	Dieser Benutzer ist bei einigen HSMs CryptoUser und bei anderen HSMs Admin. Dies kann passieren, wenn zwei SDKs versuchen, denselben Benutzer gleichzeitig mit unterschiedlichen Rollen zu erstellen.	1. Melden Sie sich als Administrator an. 2. Löschen Sie den Benutzer auf allen HSMs: user delete --username **<user's name>** --role admin

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
	<pre>}</pre>	Sie müssen diesen Benutzer entfernen und ihn mit der gewünschten Rolle neu erstellen.	user delete --username **<user's name>** --role crypto-user 3. Erstellen Sie den Benutzer mit der gewünschten Rolle: user create --username **<user's name>** --role **<desired role>**

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode

Der Benutzer „Cluster-Abdeckung“ ist „inkonsistent“	<pre>{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [],
	"cluster-coverage":
	"**inconsistent** "
	}</pre>
		Dieser Benutzer ist in einer Teilmenge von HSMs im Cluster vorhanden. Dies kann passieren, wenn ein user create teilweise erfolgreich war oder wenn ein user delete teilweise erfolgreich war.
		
		Sie müssen Ihren vorherigen Vorgang beenden, indem Sie diesen Benutzer entweder erstellen oder aus Ihrem Cluster entfernen.
			Wenn der Benutzer nicht existieren sollte, gehen Sie wie folgt vor:
			
			1. Melden Sie sich als Administrator an.
			2. Führen Sie diesen Befehl aus:
			
			user delete --username**<user's name>** --role admin
			3. Führen Sie den Befehl jetzt aus:
			
			user delete --username**<user's name>** --role crypto-user
			
			Wenn der Benutzer existieren sollte, gehen Sie wie folgt vor:
			
			1. Melden Sie sich als Administrator an.
			2. Führen Sie den Befehl aus:
			
			user create --username **<user's name>**

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
			--role **<desired role>**

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
Der Parameter „Gesperrt“ des Benutzers ist „inkonsistent“ oder „wahr“	`{ "username": "test_user", "role": "crypto-user", "locked": **inconsistent**, "mfa": [], "cluster-coverage": "full" }`	Dieser Benutzer ist für eine Teilmenge von HSMs gesperrt. Dies kann passieren, wenn ein Benutzer das falsche Passwort verwendet und sich nur mit einer Teilmenge von HSMs im Cluster verbindet. Sie müssen die Anmeldeinformationen des Benutzers ändern, damit sie im gesamten Cluster einheitlich sind.	Wenn der Benutzer MFA aktiviert hat, gehen Sie wie folgt vor: 1. Melden Sie sich als Administrator an. 2. Verwenden Sie den folgenden Befehl, um MFA vorübergehend zu deaktivieren: user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable 3. Ändern Sie das Passwort des Benutzers, damit er sich bei allen HSMs anmelden kann: user change-password --username **<user's name>** --role **<desired role>** Wenn MFA für den Benutzer aktiv sein

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
			soll, befolgen Sie diese Schritte: 1. Lassen Sie den Benutzer sich anmelden und MFA erneut aktiviere n (dazu muss er Tokens signieren und seinen öffentlic hen Schlüssel in einer PEM-Datei angeben): user change-mfa token-sig n --username **<user's name>** --role **<desired role>** —token <File>

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode

Der MFA-Status ist „inkonsistent“	{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [
	{
	"strategy":
	"token-sign",
	"status":
	"**inconsistent** "
	}
	],
	"cluster-coverage":
	"full"
	}
		Dieser Benutzer hat unterschiedliche MFA-Flags auf verschiedenen HSMs im Cluster.
		
		Dies kann passieren, wenn ein MFA-Vorgang nur für eine Teilmenge von HSMs abgeschlossen wurde.
		
		Sie müssen das Passwort des Benutzers zurücksetzen und ihm erlauben, MFA erneut zu aktivieren.
			Wenn der Benutzer MFA aktiviert hat, gehen Sie wie folgt vor:
			
			1. Melden Sie sich als Administrator an.
			2. Verwenden Sie den folgenden Befehl, um MFA vorübergehend zu deaktivieren:
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			
			3. Anschließend müssen Sie auch das Passwort des Benutzers ändern, damit er sich bei allen HSMs anmelden kann:
			
			user change-password --username **<user's name>** --role **<desired role>**

Inkonsistentes Merkmal	Beispielausgabe einer Benutzerliste	Auswirkung	Wiederherstellungsmethode
			Wenn MFA für den Benutzer aktiv sein soll, befolgen Sie diese Schritte: 1. Lassen Sie den Benutzer sich anmelden und MFA erneut aktiviere n (dazu muss er Tokens signieren und seinen öffentlic hen Schlüssel in einer PEM-Datei angeben): user change-mfa token-sig n --username **<user's name>** --role **<desired role>** —token <File>

Bei der Überprüfung der Schlüsselverfügbarkeit ist ein Fehler aufgetreten

Problem: Ein HSM gibt den folgenden Fehler zurück:

```
Key <KEY HANDLE> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

Ursache: Bei der Überprüfung der Verfügbarkeit von Schlüsseln wird nach Schlüsseln gesucht, die in seltenen, aber möglichen Fällen verloren gehen könnten. Dieser Fehler tritt normalerweise in Clustern

mit nur einem HSM oder in Clustern mit zwei HSMs während eines Zeitraums auf, in dem eines von ihnen ersetzt wird. In diesen Situationen haben die folgenden Kundenvorgänge wahrscheinlich zu dem oben genannten Fehler geführt:

- Ein neuer Schlüssel wurde mit einem Befehl wie key generate-symmetric oder generiertkey generate-asymmetric-pair .
- Eine key list-Operation wurde gestartet.
- Eine neue Instance des SDK wurde gestartet.

> **Note**
>
> OpenSSL verzweigt häufig neue Instances des SDK.

Lösung/Empfehlung: Wählen Sie aus den folgenden Aktionen, um das Auftreten dieses Fehlers zu verhindern:

- Verwenden Sie den --disable-key-availability-check-Parameter, um die Schlüsselverfügbarkeit in der Konfigurationsdatei Ihres Configure Tools auf Falsch zu setzen. Weitere Informationen finden Sie im Abschnitt Parameter des Configure Tools.
- Wenn Sie einen Cluster mit zwei HSMs verwenden, vermeiden Sie es, die Operationen zu verwenden, die zu dem Fehler geführt haben, außer während des Initialisierungscodes.
- Erhöhen Sie die Anzahl der HSMs in Ihrem Cluster auf mindestens drei.

Extrahieren von Schlüsseln mit JCE

getEncoded getPrivateExponentoder getS gibt null zurück

`getEncoded`, `getPrivateExponent` und `getS` geben Null zurück, da sie standardmäßig deaktiviert sind. Informationen zum Aktivieren finden Sie unter Schlüsselextraktion mit JCE.

Wenn `getEncoded`, `getPrivateExponent` und `getS` nach der Aktivierung Null zurückgeben, erfüllt Ihr Schlüssel nicht die richtigen Voraussetzungen. Weitere Informationen finden Sie unter Schlüsselextraktion mit JCE.

getEncoded oder getS getPrivateExponent-Rückgabeschlüsselbytes außerhalb des HSM

Sie oder jemand mit Zugriff auf Ihr System haben die Extraktion von klaren Schlüsseln aktiviert. Auf den folgenden Seiten finden Sie weitere Informationen, unter anderem, wie Sie diese Konfiguration auf den deaktivierten Standardstatus zurücksetzen können.

- Schlüsselextraktion mit JCE
- Schlüssel aus einem HSM schützen und extrahieren

HSM-Drosselung

Wenn Ihre Workload die HSM-Kapazität Ihres Clusters überschreitet, erhalten Sie Fehlermeldungen, die darauf hinweisen, dass HSMs ausgelastet oder gedrosselt sind. In diesem Fall kann es zu einem verringerten Durchsatz oder zu einer erhöhten Anzahl von Ablehnungsanfragen von HSMs kommen. Darüber hinaus senden HSMs möglicherweise die folgenden Busy-Fehler.

Für Client-SDK 5

- In PKCS11 werden Busy-Fehler `CKR_FUNCTION_FAILED` zugeordnet. Dieser Fehler kann aus mehreren Gründen auftreten. Wenn dieser Fehler jedoch durch HSM-Drosselung verursacht wird, werden die folgenden Protokollzeilen in Ihrem Protokoll angezeigt:
  - `[cloudhsm_provider::hsm1::hsm_connection::e2e_encryption::error] Failed to prepare E2E response. Error: Received error response code from Server. Response Code: 187`
  - `[cloudhsm_pkcs11::decryption::aes_gcm] Received error from the server. Error: This operation is already in progress. Internal error code: 0x000000BB`
- In JCE werden Busy-Fehler `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected error with the Provider: The HSM could not queue the request for processing.` zugeordnet.
- Bei Auslastungsfehlern anderer SDKs wird die folgende Meldung ausgegeben: `Received error response code from Server. Response Code: 187`.

Für Client-SDK 3

- In PKCS11 werden Busy-Fehler `CKR_OPERATION_ACTIVE`Fehlern zugeordnet.
- In JCE werden Busy-Fehler `CFM2Exception` mit dem Status `0xBB (187)` zugeordnet. Anwendungen können die `getStatus()`-Funktion auf `CFM2Exception` verwenden, um zu überprüfen, welcher Status vom HSM zurückgegeben wird.
- Bei Auslastungsfehlern anderer SDKs wird die folgende Meldung ausgegeben: `HSM Error: HSM is already busy generating the keys(or random bytes) for another request.`

Behebung

Sie können diese Probleme beheben, indem Sie mindestens eine der folgenden Maßnahmen ergreifen:

- Fügen Sie Wiederholungsbefehle für abgelehnte HSM-Operationen in Ihrer Anwendungsebene hinzu. Stellen Sie vor der Aktivierung von Wiederholungsbefehlen sicher, dass Ihr Cluster ausreichend dimensioniert ist, um Spitzenlasten zu bewältigen.

  > **Note**
  >
  > Für Client-SDK 5.8.0 und höher sind Wiederholungsbefehle standardmäßig aktiviert. Einzelheiten zur Konfiguration der Wiederholungsbefehle der einzelnen SDKs finden Sie unter Erweiterte Konfigurationen für das Client-SDK-5-Configure-Tool.

- Fügen Sie Ihrem Cluster weitere HSMs hinzu, indem Sie den Anweisungen unter Hinzufügen oder Entfernen von HSMs in einem Cluster AWS CloudHSM folgen.

  > **Important**
  >
  > Wir empfehlen, Ihren Cluster einem Belastungstest zu unterziehen, um die zu erwartende Spitzenlast zu ermitteln, und dann ein weiteres HSM hinzuzufügen, um eine hohe Verfügbarkeit zu gewährleisten.

Aufrechthalten der Synchronität von HSM-Benutzern in den verschiedenen HSMs im Cluster

Zum Verwalten der Benutzer Ihres HSM, verwenden Sie das AWS CloudHSM-Befehlszeilen-Tool cloudhsm_mgmt_util. Dieses kommuniziert nur mit den HSMs in der Konfigurationsdatei des Tools. Es berücksichtigt keine anderen HSMs im Cluster, die sich nicht in der Konfigurationsdatei befinden.

AWS CloudHSM synchronisiert die Schlüssel auf Ihren HSMs in allen anderen HSMs im Cluster, jedoch nicht die Benutzer oder Richtlinien des HSM. Wenn Sie cloudhsm_mgmt_util zum manage HSM users (Verwalten der HSM-Benutzer) verwenden, wirken sich diese Benutzeränderungen möglicherweise nur auf einige HSMs des Clusters ausnämlich auf diejenigen, die sich in der cloudhsm_mgmt_util-Konfigurationsdatei befinden. Dies kann zu Problemen führen, wenn AWS CloudHSM Schlüssel in verschiedenen HSMs im Cluster synchronisiert, da die Benutzer, die diese Schlüssel besitzen, möglicherweise nicht in allen HSMs im Cluster vorhanden sind.

Um diese Probleme zu vermeiden, bearbeiten Sie die cloudhsm_mgmt_util-Konfigurationsdatei, bevor Sie Benutzer verwalten. Weitere Informationen finden Sie unter ???.

Verbindung zum Cluster getrennt

Bei der Konfiguration des AWS CloudHSM Clients haben Sie die IP-Adresse des ersten HSM in Ihrem Cluster angegeben. Diese IP-Adresse wird in der Konfigurationsdatei für den AWS CloudHSM Client gespeichert. Wenn der Client gestartet wird, wird versucht, eine Verbindung zu dieser IP-Adresse einzurichten. Wenn das nicht möglich ist – weil beispielsweise der HSM fehlgeschlagen ist oder Sie ihn gelöscht haben – wird möglicherweise eine der folgenden Fehlermeldungen angezeigt:

```
LIQUIDSECURITY: Daemon socket connection error
```

```
LIQUIDSECURITY: Invalid Operation
```

Um diese Fehler zu beheben, aktualisieren Sie die Konfigurationsdatei mit der IP-Adresse eines aktiven, verfügbaren HSM im Cluster.

Um die Konfigurationsdatei für den AWS CloudHSM Client zu aktualisieren

1. Verwenden Sie eine der folgenden Methoden, um die IP-Adresse eines aktiven HSM in Ihrem Cluster zu ermitteln.

- Zeigen Sie die Registerkarte HSMs auf der Cluster-Detailseite in der AWS CloudHSM - Konsole an.
- Verwenden Sie die AWS Command Line Interface (CLI), um den describe-clustersBefehl auszuführen.

Sie benötigen diese IP-Adresse in einem späteren Schritt.

2. Verwenden Sie den folgenden Befehl, um den Client zu beenden.

Amazon Linux

```
$ sudo stop cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

Windows

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

  Verwenden Sie Strg + C im Befehlsfenster, in dem Sie den AWS CloudHSM Client gestartet haben.

3. Verwenden Sie den folgenden Befehl ein, um die Konfigurationsdatei des Clients zu aktualisieren, und geben Sie dazu die IP-Adresse an, die Sie in einem vorherigen Schritt erhalten haben.

```
$ sudo /opt/cloudhsm/bin/configure -a <IP address>
```

4. Verwenden Sie den folgenden Befehl, um den -Client zu starten.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Für Windows-Client 1.1.2 und höher:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Für Windows-Clients 1.1.1 und früher:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Fehlende AWS CloudHSM-Auditprotokolle in CloudWatch

Wenn Sie einen Cluster vor dem 20. Januar 2018 erstellt haben, müssen Sie manuell eine service-linked role (serviceverknüpfte Rolle) konfigurieren, damit die Audit-Protokolle des Clusters bereitgestellt werden. Anweisungen zum Aktivieren einer serviceverknüpften Rolle in einem HSM-Cluster finden Sie unter Understanding Service-Linked Roles (Grundlegendes zu serviceverknüpften Rollen), sowie unter Creating a Service-Linked Role (Erstellen einer serviceverknüpften Rolle) im IAM-Benutzerleitfaden.

Benutzerdefinierte IVs mit nicht kompatibler Länge für AES-Schlüsselumhüllung

Mithilfe dieses Themas zur Problembehandlung können Sie feststellen, ob Ihre Anwendung unwiederbringliche verpackte Schlüssel generiert. Wenn Sie von diesem Problem betroffen sind, verwenden Sie dieses Thema, um das Problem zu lösen.

Themen

Stellen Sie fest, ob Ihr Code unwiederbringliche verpackte Schlüssel generiert

Sie sind nur betroffen, wenn Sie alle folgenden Bedingungen erfüllen:

Bedingung	Woher soll ich das wissen?
Ihre Anwendung verwendet die PKCS #11-Bibliothek	Die PKCS #11-Bibliothek wird als `libpkcs11.so` -Datei in Ihrem `/opt/cloudhsm/lib` -Ordner installiert. In der Sprache C geschriebene Anwendungen verwenden im Allgemeinen direkt die PKCS #11-Bibliothek, während in Java geschriebene Anwendungen die Bibliothek möglicherweise indirekt über eine Java-Abstraktionsschicht verwenden. Wenn Sie Windows verwenden, sind Sie davon NICHT betroffen, da die PKCS #11-Bibliothek derzeit nicht für Windows verfügbar ist.
Ihre Anwendung verwendet speziell Version 3.0.0 der PKCS #11-Bibliothek	Wenn Sie eine E-Mail vom AWS CloudHSM-Team erhalten haben, verwenden Sie wahrscheinlich Version 3.0.0 der PKCS #11-Bibliothek.

Bedingung	Woher soll ich das wissen?
	Verwenden Sie diesen Befehl, um die Softwareversion auf Ihren Anwendungs-Instanc es zu überprüfen: `rpm -qa | grep ^cloudhsm`
Schlüssel werden mithilfe von AES-Schlü sselumbruch umgebrochen	AES-Schlüsselumbruch bedeutet, dass Sie einen AES-Schlüssel verwenden, um einen anderen Schlüssel zu umschließen. Der Name des entsprechenden Mechanismus lautet `CKM_AES_KEY_WRAP` . Er wird zusammen mit der Funktion `C_WrapKey` verwendet. Andere AES-basierte Wrapping-Mechanism en, die Initialisierungsvektoren (IV) verwenden , wie z. B. `CKM_AES_GCM` und `CKM_CLOUD HSM_AES_GCM` , sind von diesem Problem nicht betroffen. Erfahren Sie mehr über Funktionen und Mechanismen.
Sie geben eine benutzerdefinierte IV an, wenn Sie AES Key Wrapping aufrufen, und die Länge dieser IV ist kürzer als 8	Der AES-Schlüsselumbruch wird im Allgemein en mit einer `CK_MECHANISM` Struktur wie folgt initialisiert: `CK_MECHANISM mech = {CKM_AES_ KEY_WRAP, IV_POINTER, IV_LENGTH };` Dieses Problem trifft auf Sie nur in folgenden Fällen zu: • IV_POINTER ist nicht NULL • IV_LENGTH ist weniger als 8 Byte

Wenn Sie nicht alle oben genannten Bedingungen erfüllen, können Sie jetzt mit dem Lesen aufhören. Ihre verpackten Schlüssel können ordnungsgemäß ausgepackt werden, und dieses Problem hat

keine Auswirkungen auf Sie. Andernfalls lesen Sie unter the section called “Maßnahmen, die Sie ergreifen müssen, wenn Ihr Code unwiederbringliche verpackte Schlüssel generiert” weiter.

Maßnahmen, die Sie ergreifen müssen, wenn Ihr Code unwiederbringliche verpackte Schlüssel generiert

Sie sollten die folgenden drei Schritte ausführen:

1. Aktualisieren Sie Ihre PKCS #11-Bibliothek sofort auf eine neuere Version
   - Aktuelle PKCS #11-Bibliothek für Amazon Linux, CentOS 6 und RHEL 6
   - Aktuelle PKCS #11-Bibliothek für Amazon Linux 2, CentOS 7 und RHEL 7
   - Aktuelle PKCS #11-Bibliothek für Ubuntu 16.04 LTS
2. Aktualisieren Sie Ihre Software, um eine standardkonforme IV zu verwenden

   Wir empfehlen Ihnen dringend, unserem Beispielcode zu folgen und einfach eine NULL IV anzugeben, sodass das HSM die standardkonforme Standard-IV verwendet. Alternativ können Sie die IV explizit als `0xA6A6A6A6A6A6A6A6` mit einer entsprechenden IV-Länge von 8 angeben. Wir empfehlen, keine anderen IV für das Umschließen von AES-Schlüsseln zu verwenden, und werden benutzerdefinierte IVs für das Umschließen von AES-Schlüsseln in einer künftigen Version der PKCS #11-Bibliothek ausdrücklich deaktivieren.

   Der Beispielcode für die korrekte Angabe der IV befindet sich in aes_wrapping.c auf GitHub.
3. Identifizieren Sie vorhandene verpackte Schlüssel und stellen Sie sie wieder her

   Sie sollten alle Schlüssel identifizieren, die Sie mit Version 3.0.0 der PKCS #11-Bibliothek verpackt haben, und sich dann an den Support (https://aws.amazon.com/support) wenden, um Unterstützung bei der Wiederherstellung dieser Schlüssel zu erhalten.

> **Important**
>
> Dieses Problem betrifft nur Schlüssel, die mit Version 3.0.0 der PKCS #11-Bibliothek umschlossen wurden. Sie können Schlüssel mithilfe früherer Versionen (2.0.4 und Pakete mit niedrigeren Nummern) oder neueren Versionen (3.0.1 und Pakete mit höheren Nummern) der PKCS #11-Bibliothek umbrechen.

Beheben von Cluster-Erstellungsfehlern

Wenn Sie einen Cluster erstellen, AWS CloudHSM wird die AWSServiceRoleForCloudHSM serviceverknüpfte Rolle erstellt, sofern die Rolle noch nicht vorhanden ist. Wenn die dienstverknüpfte Rolle AWS CloudHSM nicht erstellt werden kann, schlägt Ihr Versuch, einen Cluster zu erstellen, möglicherweise fehl.

In diesem Thema wird erklärt, wie Sie die am häufigsten auftretenden Probleme lösen und so erfolgreich einen Cluster erstellen. Sie müssen diese Rolle nur einmal erstellen. Sobald die servicegebundene Rolle in Ihrem Konto erstellt wurde, können Sie mit jeder der unterstützten Methoden weitere Cluster erstellen und verwalten.

Die folgenden Abschnitte halten Ratschläge bereit, mit der Fehler bei der Cluster-Erstellung behoben werden können, falls diese im Zusammenhang mit der servicegebundenen Rolle stehen. Wenn Sie nach Ausführung dieser Ratschläge immer noch nicht in der Lage sind, Cluster zu erstellen, wenden Sie sich an den AWS Support. Weitere Informationen zur AWSServiceRoleForCloudHSM dienstbezogenen Rolle finden Sie unter. Mit Diensten verknüpfte Rollen für AWS CloudHSM

Themen

Fügen Sie die fehlende Berechtigung hinzu

Um eine servicegebundene Rolle erstellen zu können, muss der Benutzer über die `iam:CreateServiceLinkedRole`-Berechtigung verfügen. Wenn der IAM-Benutzer, der den Cluster erstellt, nicht über diese Berechtigung verfügt, schlägt der Clustererstellungsprozess fehl, wenn versucht wird, die dienstverknüpfte Rolle in Ihrem Konto zu erstellen. AWS

Wenn eine fehlende Berechtigung den Fehler verursacht, enthält die Fehlermeldung den folgenden Text.

```
This operation requires that the caller have permission to call
 iam:CreateServiceLinkedRole to create the CloudHSM Service Linked Role.
```

Um diesen Fehler zu beheben, geben Sie dem IAM-Benutzer, der den Cluster erstellt, die `AdministratorAccess`-Berechtigung oder fügen Sie der IAM-Richtlinie des Benutzers die

`iam:CreateServiceLinkedRole`-Berechtigung hinzu. Weitere Anleitungen finden Sie unter Hinzufügen von Berechtigungen zu einem neuen oder vorhandenen Benutzer (Konsole).

Versuchen Sie anschließend erneut, den Cluster zu erstellen.

Manuelles Erstellen der serviceverknüpften Rolle

Sie können die IAM-Konsole, CLI oder API verwenden, um die AWSServiceRoleForCloudHSM serviceverknüpfte Rolle zu erstellen. Weitere Informationen finden Sie unter Erstellen einer serviceverknüpfte Rolle im IAM-Leitfaden.

Verwenden Sie einen nicht verbundenen Benutzer

Verbundbenutzer, deren Anmeldeinformationen außerhalb von stammen AWS, können viele der Aufgaben eines nicht verbundenen Benutzers ausführen. Mit AWS können Benutzer jedoch keine API-Aufrufe zum Erstellen einer servicegebundenen Rolle von einem verbundenen Endpunkt aus ausführen.

Um dieses Problem zu lösen, erstellen Sie einen nicht verbundenen Benutzer mit der `iam:CreateServiceLinkedRole`-Berechtigung oder erteilen Sie einem vorhandenen nicht verbundenen Benutzer die `iam:CreateServiceLinkedRole`-Berechtigung. Lassen Sie diesen Benutzer dann über die CLI einen Cluster erstellen. Dadurch wird die servicegebundene Rolle in Ihrem Konto erstellt.

Sobald die servicegebundene Rolle erstellt wurde, können Sie den Cluster, der den nicht verbundenen Benutzer erstellt hat, löschen. Das Löschen des Clusters hat keine Auswirkungen auf die Rolle. Danach kann jeder Benutzer mit den erforderlichen Berechtigungen, einschließlich Verbundbenutzer, AWS CloudHSM Cluster in Ihrem Konto erstellen.

Um zu überprüfen, ob die Rolle erstellt wurde, öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/ und wählen Sie Rollen aus. Oder verwenden Sie den Befehl IAM get-role in der CLI.

```
$  aws iam get-role --role-name AWSServiceRoleForCloudHSM
{
    "Role": {
        "Description": "Role for CloudHSM service operations",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
```

```
            "Statement": [
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "cloudhsm.amazonaws.com"
                    }
                }
            ]
        },
        "RoleId": "AROAJ4I6WN5QVGG5G7CBY",
        "CreateDate": "2017-12-19T20:53:12Z",
        "RoleName": "AWSServiceRoleForCloudHSM",
        "Path": "/aws-service-role/cloudhsm.amazonaws.com/",
        "Arn": "arn:aws:iam::111122223333:role/aws-service-role/cloudhsm.amazonaws.com/
AWSServiceRoleForCloudHSM"
    }
}
```

Abruf von Clientkonfigurationsprotokollen

AWS CloudHSM bietet Tools für Client-SDK 3 und Client-SDK 5, mit denen Sie Informationen über Ihre Umgebung sammeln können, damit der AWS-Support Probleme beheben kann.

Themen

- Support-Tool für das Client-SDK 5
- Support-Tool für das Client-SDK 3

Support-Tool für das Client-SDK 5

Das Skript extrahiert die folgenden Informationen:

- Die Konfigurationsdatei für die Client-SDK 5-Komponente
- Verfügbare Protokolldateien
- Aktuelle Version des Betriebssystems
- Paketinformationen

Das Info-Tool für Client-SDK 5 wird ausgeführt

Das Client-SDK 5 umfasst ein Client-Support-Tool für jede Komponente, aber alle Tools funktionieren gleich. Führen Sie das Tool aus, um eine Ausgabedatei mit allen erfassten Informationen zu erstellen.

Die Tools verwenden eine Syntax wie diese:

```
[ pkcs11 | dyn | jce ]_info
```

Um beispielsweise Informationen zur Unterstützung von einem Linux-Host zu sammeln, auf dem die PKCS #11-Bibliothek läuft, und das System in das Standardverzeichnis schreiben zu lassen, würden Sie diesen Befehl ausführen:

```
/opt/cloudhsm/bin/pkcs11_info
```

Das Tool erstellt die Ausgabedatei innerhalb des `/tmp`-Verzeichnisses.

PKCS #11 library

Um Unterstützungsdaten für die PKCS #11-Bibliothek unter Linux zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

  ```
  /opt/cloudhsm/bin/pkcs11_info
  ```

Um Unterstützungsdaten für die PKCS #11-Bibliothek unter Windows zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

  ```
  C:\Program Files\Amazon\CloudHSM\bin\pkcs11_info.exe
  ```

OpenSSL Dynamic Engine

Um Unterstützungsdaten für OpenSSL Dynamic Engine unter Linux zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

  ```
  /opt/cloudhsm/bin/dyn_info
  ```

JCE provider

Um Supportdaten für den JCE-Anbieter unter Linux zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
/opt/cloudhsm/bin/jce_info
```

Um Supportdaten für den JCE-Anbieter unter Windows zu sammeln

- Verwenden Sie das Support-Tool, um Daten zu sammeln.

```
C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe
```

Abrufen von Protokollen aus einer Serverless-Umgebung

Um für Serverless-Umgebungen wie Fargate oder Lambda zu konfigurieren, empfehlen wir Ihnen, den AWS CloudHSM-Protokolltyp auf `term`. Nach der Konfiguration von `term` kann die Serverless-Umgebung Daten an CloudWatch ausgeben.

Informationen zum Abrufen der Client-Protokolle von CloudWatch Logs finden Sie unter Arbeiten mit Protokollgruppen und Protokollstreams im Amazon-CloudWatch-Logs-Benutzerhandbuch.

Support-Tool für das Client-SDK 3

Das Skript extrahiert die folgenden Informationen:

- Betriebssystem und seine aktuelle Version
- Clientkonfigurationsinformationen aus `cloudhsm_client.cfg`-, `cloudhsm_mgmt_util.cfg`-, und `application.cfg`-Dateien
- Clientprotokolle von dem für die Plattform spezifischen Speicherort
- Cluster- und HSM-Informationen mithilfe von cloudhsm_mgmt_util
- OpenSSL-Informationen
- Aktuelle Client- und Build-Version
- Installationsprogrammversion

Das Info-Tool für Client-SDK 3 wird ausgeführt

Das Skript erstellt eine Ausgabedatei mit allen erfassten Informationen. Das Skript erstellt die Ausgabedatei innerhalb des `/tmp`-Verzeichnisses.

Linux: `/opt/cloudhsm/bin/client_info`

Windows: `C:\Program Files\Amazon\CloudHSM\client_info`

> **Warning**
>
> Dieses Skript hat ein bekanntes Problem für die Client-SDK 3-Versionen 3.1.0 bis 3.3.1. Wir empfehlen dringend, auf Version 3.3.2 zu aktualisieren, die eine Lösung für dieses Problem enthält. Weitere Informationen finden Sie auf der Seite Bekannte Probleme, die Sie lesen sollten, bevor Sie dieses Tool verwenden.

AWS CloudHSM-Kontingente

Kontingente, früher Limits genannt, sind die zugewiesenen Werte für AWS-Ressourcen. Die folgenden Kontingente gelten für Ihre AWS CloudHSM-Ressourcen je AWS-Region und AWS-Konto. Das Standardkontingent ist der Anfangswert, der von AWS angewendet wird, und diese Werte sind in der folgenden Tabelle aufgeführt. Ein anpassbares Kontingent kann über das Standardkontingent hinaus erhöht werden.

Service Quotas

Ressource	Standardkontingent	Anpassbar?
Cluster	4	Ja
HSMs	6	Ja
HSMs pro Cluster	28	Nein

Die empfohlene Möglichkeit, eine Kontingenterhöhung anzufordern, besteht darin, die Servicekontingente-Konsole zu öffnen. Wählen Sie in der Konsole Ihren Service und Ihr Kontingent aus, und senden Sie Ihre Anfrage. Weitere Informationen finden Sie in der Dokumentation zu Servicekontingenten.

Die Kontingente in der folgenden Systemkontingentetabelle sind nicht anpassbar.

Systemkontingente

Ressource	Kontingent
Maximale Schlüssel pro Cluster	3.300
Maximale Anzahl Benutzer pro Cluster	1,024
Maximale Länge eines Benutzernamens	31 Zeichen
Erforderlich Passwortlänge	Zwischen 7 und 32 Zeichen
Maximale Anzahl von gleichzeitigen Client-Verbindungen pro Cluster[1]	900

Ressource	Kontingent
Maximale Anzahl von PKCS #11-Sitzungen pro Anwendung	1,024

[1] Eine Client-Verbindung für Client-SDK 3 ist ein Client-Daemon. Für das Client-SDK 5 ist eine Client-Verbindung eine Anwendung.

Weitere Informationen finden Sie unter Systemressourcen.

Systemressourcen

Systemressourcen-Kontingente sind Kontingente für die Ressourcen, die der AWS CloudHSM-Client verwenden darf, wenn er ausgeführt wird.

Dateideskriptoren sind der Mechanismus eines Betriebssystems zum Identifizieren und Verwalten offener Dateien pro Prozess.

Der CloudHSM-Client-Daemon verwendet Dateideskriptoren, um Verbindungen zwischen Anwendungen und dem Client sowie zwischen dem Client und dem Server zu verwalten.

Standardmäßig weist die CloudHSM-Client-Konfiguration 3 000 Dateideskriptoren zu. Dieser Standardwert ist darauf ausgelegt, eine optimale Sitzung und Threading-Kapazität zwischen dem Client-Daemon und Ihren HSM zu erzielen.

In seltenen Fällen kann es erforderlich sein, diese Standardwerte zu ändern, wenn Sie Ihren Client in einer Umgebung mit eingeschränkten Ressourcen ausführen.

> **Note**
>
> Wenn Sie diese Werte ändern, wird möglicherweise die Leistung Ihres CloudHSM-Clients beeinträchtigt und/oder Ihre Anwendung ist nicht mehr betriebsbereit.

1. Bearbeiten Sie die `/etc/security/limits.d/cloudhsm.conf`-Datei.

```
#
```

```
# DO NOT EDIT THIS FILE
#
hsmuser soft nofile 3000
hsmuser hard nofile 3000
```

2. Bearbeiten Sie die numerischen Werte nach Bedarf.

> **Note**
>
> Das `soft`-Kontingent muss kleiner oder gleich dem `hard`-Kontingent sein.

3. Starten Sie Ihren CloudHSM-Client-Daemon-Prozess neu.

> **Note**
>
> Diese Konfigurationsoption ist auf Microsoft Windows-Plattformen nicht verfügbar.

Downloads für das AWS CloudHSM Client-SDK

Downloads

Im März 2021 wurde die Client-SDK-Version 5.0.0 AWS CloudHSM veröffentlicht, die ein völlig neues Client-SDK mit unterschiedlichen Anforderungen, Funktionen und Plattformunterstützung einführt.

Client SDK 5 wird für Produktionsumgebungen vollständig unterstützt und bietet dieselben Komponenten und denselben Support wie Client SDK 3, mit Ausnahme der Unterstützung für CNG- und KSP-Anbieter. Weitere Informationen finden Sie unter Vergleich der Client-SDK-Komponenten.

> **Note**
>
> Informationen darüber, welche Plattformen von den einzelnen Client-SDKs unterstützt werden, finden Sie unter und. Client-SDK 5 unterstützte Plattformen Unterstützte Plattformen vom Client-SDK 3

Neuste Version

Dieser Abschnitt enthält die neueste Version des Client-SDK.

Version 5 des Client-SDK: Version 5.12.0

Amazon Linux 2

Laden Sie die Version 5.12.0 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE-Anbieter (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)

- CloudHSM-CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

Laden Sie die Version 5.12.0 der Software für Amazon Linux 2 auf der ARM64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum c28a1f27e23e6ab1550dab6a353c6c9338a391a84d57f4ac99a1a3a9810c753f)
- OpenSSL Dynamic Engine (SHA256 checksum 7d2e864c31c13f55443c1b1d04589fbdd4558fe103954de4384691e2c429a872)
- JCE-Anbieter (SHA256 checksum e9a35eb87b2f257c47fb083d286deb835da45858b2d89759ca7d5bb4ef747b4b)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum 28b6f918912b5c63bf10018824b642a805b309c21947a1d0ebbdc44647e80554)

Amazon Linux 2023

Laden Sie die Version 5.12.0 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 02801365cba449c5238a4e5ad3df1ddf7edd00ade976f47e956e885286503f3f)
- OpenSSL Dynamic Engine (SHA256 checksum 0abed69a7c6acaafdaabdcc5fab7d56611ffd94f5480cade6f8beace9aeae056)
- JCE-Anbieter (SHA256 checksum 3d5d9a903d3a216eca40f92dbb0b4030b7a86ad7ceee8d62241c97a6e1881e25)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum f96671d882b862033bba0b3633448dc6a26e45a25063e29b79a5cd4b7fc4945c)

Laden Sie die Version 5.12.0 der Software für Amazon Linux 2023 auf der ARM64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 53d05006b46bda8e9c1dd76e8307a780bfe0a67b10a9a87723c97f94e29f5b8e)
- OpenSSL Dynamic Engine (SHA256 checksum ec1cca8e01b3303ff9473eeef6b33dc85b6affac7a47387b098905f9f2fc85ba)
- JCE-Anbieter (SHA256 checksum c828ae56f46233215b9f35798b5859ebdac962af442acbc457081c3baaa44f11)

  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum ddd5dcd68d01f4fafaf13dc0b4ddcf98e3731ed51bdd51f85535b29353644a9f)

CentOS 7 (7.8+)

Laden Sie die Version 5.12.0 der Software für CentOS 7 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE-Anbieter (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 7 (7.8+)

Laden Sie die Version 5.12.0 der Software für RHEL 7 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE-Anbieter (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 8 (8.3+)

Laden Sie die Softwareversion 5.12.0 für RHEL 8 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 6e51e95122fd0991278888287f0c408808b26fb5f1196c46168477b9090fc478)

- OpenSSL Dynamic Engine (SHA256 checksum 1f1d52ff7af6c537d8cfeb5973c691a9d90a518accd685ff9b66cd78daf98928)
- JCE-Anbieter (SHA256 checksum 156944607de987d6b39bd8a2d21ccd294c01377a9e35f9f15f8b0f4c8bb90033)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum 351e802f79dd2d0b5f7d23bb74c146be05e5169b603c9aace24189094a45a35d)

RHEL 9 (9.2+)

Laden Sie die Softwareversion 5.12.0 für RHEL 9 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum d1b2f4ac7e6e0c18e788512e7726bc68b571d99a1442ce2f2e80f4b0f9956266)
- OpenSSL Dynamic Engine (SHA256 checksum cf86a3f17cd6c51969d4ce80c1e3ea6513b995611be7e2e72e5e5233c71d6add)
- JCE-Anbieter (SHA256 checksum ae89e256eb89ec6b4fa0f001e7a4e1d8f1c08530423e81aa74d69a17b25d9a99)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum dfe6fe5d890c33b2f5d38f906ade113b06c8c05f3427a327744c454e7302f1a5)

Laden Sie die Version 5.12.0 der Software für RHEL 9 auf der ARM64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum cad72a6ab2232b4c38b90d7c62147520b975d646773dd90d7be897fa0a537d2d)
- OpenSSL Dynamic Engine (SHA256 checksum ad751f756530a2317c3c64380ea3a07865b13e1874fab0e61ac530b21487c7fb)
- JCE-Anbieter (SHA256 checksum d204e69acfbb90996fb08ae3573607b65630b1124fb379e078c002d55ac07766)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum c0f412cc59bafd235e046cdc1a0c5d330f2d72f7d6434672e9522f86bc945090)

Ubuntu 20.04 LTS

Laden Sie die Version 5.12.0 der Software für Ubuntu 20.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum d37b1f872eb2b1ab34303d5b8b803daa925902b645c57c6e15a28bb6321e0f42)
- OpenSSL Dynamic Engine (SHA256 checksum cdc6e737652556b57d26d8816b2bc9820128cb3919360660b6f7fe65f9d39e3f)
- JCE-Anbieter (SHA256 checksum f567a08344414a4776e1c5a9715657476925ca32695c4c2dd84a4f3fc5dc1615)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum f2ee5ad01c5018fc3670f602228fd71087228cd3923bf5b9bc73e4d7084dac6c)

Ubuntu 22.04 LTS

Laden Sie die Version 5.12.0 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 0e78928acd7a1662e4b07b15d5c3ccb88714ff89e47b991c8ab6e4c2229ee5aa)
- OpenSSL Dynamic Engine (SHA256 checksum 4f3168745edc5592234891a7b1d82b179a4947e87c72fade1be3bad58b7ed1a3)
- JCE-Anbieter (SHA256 checksum d4c3655cdc2b00d1ab5ceafac94dfbc5c5244ed20e10fdd9db9f4e741e013733)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum d00bbacb6f2e57bd92d832a2bd11cadede972f8e82cc402ec0684b9c6b23123c)

Laden Sie die Version 5.12.0 der Software für Ubuntu 22.04 LTS auf der ARM64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 0c1121535c523acb864215338292bab32acee438357878b5fc0b6d268713b86f)
- OpenSSL Dynamic Engine (SHA256 checksum dc7a219302021570bc8c36674d2bd33165557bb2f9a0af8fdf114f1b85a70d84)
- JCE-Anbieter (SHA256 checksum af3834a10081f1e4e7894275c8b9c7b7649b8de3b6f0aeb0781a3358183a9046)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum baa253ac62c2fbcc5712561e0fb0feb25461efc3ce68cf86d4c7bf0af0f14a34)

Windows Server 2016

Laden Sie die Version 5.12.0 der Software für Windows Server 2016 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- JCE-Anbieter (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Windows Server 2019

Laden Sie die Software der Version 5.12.0 für Windows Server 2019 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- JCE-Anbieter (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - Javadocs für AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM-CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Das Client-SDK 5.12.0 bietet ARM-Unterstützung für mehrere Plattformen und Leistungsverbesserungen für alle SDKs. Der CloudHSM CLI und der JCE-Anbieter wurden um neue Funktionen erweitert.

Plattformunterstützung

- Unterstützung für Amazon Linux 2023 auf der ARM64-Architektur für alle SDKs hinzugefügt.
- Unterstützung für Red Hat Enterprise Linux 9 (9.2+) auf der ARM64-Architektur für alle SDKs hinzugefügt.
- Unterstützung für Ubuntu 22.04 LTS auf der ARM64-Architektur für alle SDKs hinzugefügt.

CloudHSM-CLI

- Der folgende Befehl wurde hinzugefügt:

  - Schlüsselreplikat
- Unterstützung für die Verbindung zu mehreren Clustern hinzugefügt. Weitere Informationen finden Sie unter Mit CLI eine Verbindung zu mehreren Clustern herstellen.

JCE-Anbieter

- Hinzugefügt `KeyReferenceSpec` für das Abrufen von Schlüsseln mit`KeyStoreWithAttributes`.
- Hinzugefügt `getKeys` zum gleichzeitigen Abrufen mehrerer Schlüssel mithilfe von. `KeyStoreWithAttributes`

Leistungsverbesserungen

- Leistungsverbesserungen für den NoPadding AES-CBC-Betrieb für alle SDKs.

Frühere Client-SDK-Versionen

In diesem Abschnitt werden frühere Client-SDK-Versionen aufgeführt.

Version 5.11.0

Amazon Linux 2

Laden Sie die Version 5.11.0 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE-Anbieter (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

Laden Sie die Version 5.11.0 der Software für Amazon Linux 2 auf der ARM64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 5ac16449ec149c9b5e7776865803245ab17d0f1ad56df80173840c5e8d257b19)
- OpenSSL Dynamic Engine (SHA256 checksum 28c2eb7f3f60172b0186e5c25f71bb7341537058a71f288673936766048083c1)
- JCE-Anbieter (SHA256 checksum 06c9d9d281c12b1d2bd9a7b601d6317e46cedf175706bbfa3e4dcaed6ba05448)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum 218982bb17aa751969a7866b0a9ff27e7aa5007a07817627d9cc1f7d60a78160)

Amazon Linux 2023

Laden Sie die Version 5.11.0 der Software für Amazon Linux 2023 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 55310ab333d18bcfabdc4b74115b040386b4508934bdff93e1d054c4c4a6f9ea)
- OpenSSL Dynamic Engine (SHA256 checksum f3d4934dc872a9b5212a180b9814ca2af3eca01ee228a8725563f1770add0dce)
- JCE-Anbieter (SHA256 checksum 757d3abb515aeb08f4b1c83970ee0979399efee00ee78c9a9dbec05f4ed9768d)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum 22af8f0501ff9a45a9e0683a408a63771c2c06c66abf5478d310d6d32e013555)

CentOS 7 (7.8+)

Laden Sie die Version 5.11.0 der Software für CentOS 7 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE-Anbieter (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 7 (7.8+)

Laden Sie die Version 5.11.0 der Software für RHEL 7 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE-Anbieter (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 8 (8.3+)

Laden Sie die Software der Version 5.11.0 für RHEL 8 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum b95b9f588656fb14fd08bb66ce0e0da807b96daa38348dec07a508c9bef7403a)
- OpenSSL Dynamic Engine (SHA256 checksum 7bb437b91a52e863b2b00ff7f427ce22522026daf757be873ee031ec6ffffd88)
- JCE-Anbieter (SHA256 checksum e0db887e05eb535314f4d99f21da12d87d35ebb8baf9726f4ce8f01d9df0ea01)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum 8485b5a6d679767ca9b4f611718159a643cf3e85090a8e4d20fe53c3707e25c3)

RHEL 9 (9.2+)

Laden Sie die Version 5.11.0 der Software für RHEL 9 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 87b56a20accf67df53a203b7f115655b2acfaec4516682d4976d9475b10bec8e)
- OpenSSL Dynamic Engine (SHA256 checksum 83a6b58572e985df937beede4b10e867b0ac6050ace8010dc8d535be365d2747)
- JCE-Anbieter (SHA256 checksum ee95213d02d913250478d0793d6dd578e5c54d765e635c7468a49bdff4c2a6f3)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum 7e168ed3bef8e9c5110645e9960680e9a57f7b94e16aec71422e3c67ebc58fb5)

Ubuntu 20.04 LTS

Laden Sie die Version 5.11.0 der Software für Ubuntu 20.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum abc3a339d1fe5850db65620804e9a910f8b4f913624ef9b7189f2f0df1825c01)
- OpenSSL Dynamic Engine (SHA256 checksum 075fc3f9974d552f27ad67fa92c8abff31b756b9add875b8cd4957e6801583a4)
- JCE-Anbieter (SHA256 checksum 5de45c519133a0dae8da3ac01809db7974be25c14c15eb773fc5c972c0178c13)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum 83e0e4505a063792c19feb3d4cfd032b9089091916168d92b0f51a967a007734)

Ubuntu 22.04 LTS

Laden Sie die Version 5.11.0 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum b8f20be125c8530b2a7bd945956e9c04296fba5634af408b40be4e03bdbad72a)
- OpenSSL Dynamic Engine (SHA256 checksum d728c156eb4ee5c67159e57d6b092785800baa5fb61c14d64f460a8b8f53a778)
- JCE-Anbieter (SHA256 checksum 44e943b8cd1176ad666e249342687744a280c6222df58b5a9f084c932f628284)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum 8ccf5389d459611be813e42d7f9d040090f94f3fe88f9d110bcfb25e9619e4a7)

Windows Server 2016

Laden Sie die Software Version 5.11.0 für Windows Server 2016 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- JCE-Anbieter (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)

- CloudHSM-CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Windows Server 2019

Laden Sie die Software Version 5.11.0 für Windows Server 2019 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- JCE-Anbieter (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - Javadocs für AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM-CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Das Client-SDK 5.11.0 fügt neue Funktionen hinzu, verbessert die Stabilität und beinhaltet Bugfixes für alle SDKs.

Plattformunterstützung

- Unterstützung für Amazon Linux 2023 und RHEL 9 (9.2+) für alle SDKs hinzugefügt.
- Die Unterstützung für Ubuntu 18.04 LTS wurde aufgrund des kürzlichen Ablaufs der Laufzeit entfernt.
- Die Unterstützung für Amazon Linux wurde aufgrund des kürzlichen Auslaufs entfernt.

CloudHSM-CLI

- Die folgenden Befehle wurden hinzugefügt:
  - Crypto-Zeichen
  - Crypto-Verifizierung
  - Schlüsselimportstift
  - Schlüssel entpacken
  - Schlüsselumbruch
- key generate-fileunterstützt jetzt den Export von öffentlichen Schlüsseln.

OpenSSL Dynamic Engine

- Die AWS CloudHSM OpenSSL Dynamic Engine wird jetzt auf Plattformen unterstützt, auf denen eine OpenSSL-Bibliotheksversion 3.x installiert ist. Dazu gehören Amazon Linux 2023, RHEL 9 (9.2+) und Ubuntu 22.04.

JCE

- Unterstützung für JDK 17 und JDK 21 hinzugefügt.
- Unterstützung für AES-Schlüssel, die für HMAC-Operationen verwendet werden sollen, wurde hinzugefügt.
- Das neue Schlüsselattribut `ID` wurde hinzugefügt.
- Eine neue `DataExceptionCause` Variante für die Erschöpfung von Schlüsseln wurde eingeführt:`DataExceptionCause.KEY_EXHAUSTED`.

Fehlerbehebungen/Verbesserungen

- Die maximale Länge für das `label` Attribut wurde von 126 auf 127 Zeichen erhöht.
- Es wurde ein Fehler behoben, der das Auspacken von EC-Schlüsseln mit dem RsaOaep Mechanismus verhinderte.
- Ein bekanntes Problem für den GetKey-Vorgang im JCE-Anbieter wurde behoben. Weitere Einzelheiten finden Sie unter Problem: Speicherverlust im Client-SDK 5 bei GetKey-Vorgängen.
- Die Protokollierung in allen SDKs für Triple DES-Schlüssel, die ihr maximales Blocklimit für die Verschlüsselung erreicht haben, wurde gemäß FIPS 140-2 verbessert.
- Bekannte Probleme für die OpenSSL Dynamic Engine wurden hinzugefügt. Details dazu finden Sie unter Bekannte Probleme für die OpenSSL Dynamic Engine.

Version 5.10.0

Amazon Linux

Laden Sie die Version 5.10.0 der Software für Amazon Linux auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum d63adf3e96c19c2d894b2defcbadd916dbb0398993050b1358bd93a36aa5acab)

- OpenSSL Dynamic Engine (SHA256 checksum 4daa3e591ffd5f7ce8ef3759c41deaa38867f5e5d21f15927aea83afb1678ac5)
- JCE-Anbieter (SHA256 checksum 6c1ac94d3080f1c609d9dafbcb14480911beef3a488c4ed6f2b11b377da9b477)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum c12617fcd7990ba53e96f477979b410e3a5f17842ca7a912861b8b820809b5b5)

Amazon Linux 2

Laden Sie die Version 5.10.0 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE-Anbieter (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

Laden Sie die Version 5.10.0 der Software für Amazon Linux 2 auf der ARM64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 5d8dfd835f1ed5a7f5a4fcc8ecf81cfa29883aca7e2985de69b5db723ab663db)
- OpenSSL Dynamic Engine (SHA256 checksum 91fb8efe2646bf0dbd9087554baa09554714e9d56e9bfd5c0dc3023a9f485574)
- JCE-Anbieter (SHA256 checksum 99f6e55c37fdf00085a816d46835aeff54470797b3b71f4d28a70dc79c9caf44)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum 4a88ba9b4cf0dd5573f3dd88ab9dc257e4c486069cb529c5d554979ee2dd83af)

CentOS 7 (7.8+)

Laden Sie die Version 5.10.0 der Software für CentOS 7 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE-Anbieter (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 7 (7.8+)

Laden Sie die Version 5.10.0 der Software für RHEL 7 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE-Anbieter (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 8 (8.3+)

Laden Sie die Version 5.10.0 der Software für RHEL 8 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 96afb7042a148ddc7a60ab6235b49e176d0460d1c2957bd76ca3d8406ac1cb03)
- OpenSSL Dynamic Engine (SHA256 checksum 2caad2bffea8aef73c91ad422d09772ef830fe7f80a7be19020e6a107eadfbe8)
- JCE-Anbieter (SHA256 checksum 3543551f08fbe3900821ea2d4ea148b4e86e2334bc94d7ffef6f3b831457cd71)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum 812eccaadfc490f13bcd0b0a835ef58f3a3d4344ad7e0a237de476dd24509525)

Ubuntu 18.04 LTS

Laden Sie die Version 5.10.0 der Software für Ubuntu 18.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum be4c61766b8b46e1f6c14c3dcf90aaab9f38240fcd9c68b4009704276c5f6f4a)
- OpenSSL Dynamic Engine (SHA256 checksum 64bd8af827b6dc3786e8ad28858cbc4ef6a0fd42164a0945f427eddcf5f02858)
- JCE-Anbieter (SHA256 checksum 9fcbdf08e93641468588b608173f26f18781bbc029ed95b2e086da29a968cc00)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum 13808bddddb7eedeb2b8486d23a9976c7fa8d9220149a6b9400626bcaff3b513)

> **Note**
>
> Aufgrund des kürzlichen Ablaufs der Lebensdauer von Ubuntu 18.04 LTS AWS CloudHSM wird diese Plattform mit der nächsten Version nicht mehr unterstützt werden können.

Ubuntu 20.04 LTS

Laden Sie die Version 5.10.0 der Software für Ubuntu 20.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 99ae96504580ff85ed4958a582903a847f666bdaafafbe887a5a76db58f24500)
- OpenSSL Dynamic Engine (SHA256 checksum 13e3f6fe086acf9617b163f66e3941f973daa583fb9322d16c396aa29fc3611d)
- JCE-Anbieter (SHA256 checksum 44562cebd9af1aa965840cd9bcb237e518d24c715b3c8bca1405c9c1871835e2)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum ab71b4ec531c5e6d05c91539c7edc1c07e6c748052ebf6200f148cb6812538c5)

Ubuntu 22.04 LTS

Laden Sie die Version 5.10.0 der Software für Ubuntu 22.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum ee331a44fbe4936ec98a3ae55d58e67ed38e8bbff0a4f4ce8b1bd8239b75877b)
- Unterstützung für OpenSSL Dynamic Engine ist für diese Plattform noch nicht verfügbar.
- JCE-Anbieter (SHA256 checksum 9e44d14dd33624f6fe36711633013e47e4a93f4d4635e08900546113ded56e3d)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum 2df361546848cd3f8965b1007dca42a0c959eb10d9e3f4995e8e1c852406751d)

Windows Server 2016

Laden Sie die Version 5.10.0 der Software für Windows Server 2016 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- JCE-Anbieter (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Windows Server 2019

Laden Sie die Version 5.10.0 der Software für Windows Server 2019 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- JCE-Anbieter (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
  - Javadocs für AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM-CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Das Client-SDK 5.10.0 verbessert die Stabilität und beinhaltet Bugfixes für alle SDKs.

CloudHSM-CLI

- Es wurden neue Befehle hinzugefügt, mit denen Kunden Schlüssel mithilfe der CloudHSM-CLI verwalten können, darunter:
  - Symmetrische Schlüssel und asymmetrische Schlüsselpaare erstellen
  - Freigabe und Aufheben der Freigabe von Schlüsseln
  - Schlüssel mithilfe von Schlüsselattributen auflisten und filtern
  - Festlegen von Schlüsselattributen
  - Wichtige Referenzdateien generieren
  - Löschen von Schlüsseln
- Verbesserte Fehlerprotokollierung.
- Unterstützung für mehrzeilige Unicode-Befehle im interaktiven Modus hinzugefügt.

Fehlerbehebungen/Verbesserungen

- Verbesserte Leistung beim Importieren, Entpacken, Ableiten und Erstellen von Sitzungsschlüsseln für alle SDKs.
- Es wurde ein Fehler im JCE-Anbieter behoben, der verhinderte, dass temporäre Dateien beim Beenden entfernt wurden.
- Es wurde ein Fehler behoben, der unter bestimmten Bedingungen zu einem Verbindungsfehler führte, nachdem HSMs im Cluster ersetzt wurden.
- Das JCE`getVersion`-Ausgabeformat wurde geändert, um große kleinere Versionsnummern zu verarbeiten und die Patch-Nummer einzubeziehen.

Plattformunterstützung

- Unterstützung für Ubuntu 22.04 mit JCE, PKCS #11 und CloudHSM-CLI hinzugefügt (Unterstützung für OpenSSL Dynamic Engine ist noch nicht verfügbar).

Version 5.9.0

Amazon Linux

Laden Sie die Version 5.9.0 der Software für Amazon Linux auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 4f368be41f006b751ac41b14e1435c27841f60bbde0f032ec02a359fea637dcf)
- OpenSSL Dynamic Engine (SHA256 checksum 81af0d34683825cd6ff844ccacf9c8f4842a4ba76e3875a89121d09a286b4490)
- JCE-Anbieter (SHA256 checksum e8e5bc09d8e0b3cb24f30ab420fe08902a19073012335ac94382ec55fcc45abd)
  - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum 17284144b45043204ce012fe8b62b1973f10068950abedbd9c2c6172ed0979c6)

Amazon Linux 2

Laden Sie die Version 5.9.0 der Software für Amazon Linux 2 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE-Anbieter (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

Laden Sie die Version 5.9.0 der Software für Amazon Linux 2 auf der ARM64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 4337dca5a08c5194b1118fa197bb4a4f7988df4e1b961e6f2e367295ba99d61d)
- OpenSSL Dynamic Engine (SHA256 checksum 4f08689934e877662a7ce64554fb04eb4b2c213b936018609ff187d100e34a85)
- JCE-Anbieter (SHA256 checksum b337b80271a2d308949d5911971fe6ad35df4e34876a481fcac347f1d897fe39)
  - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum a4d466e6b5f74dcd283ba32c9dd87441941d5e5a05936b7c2b4cc7ef85eb1071)

CentOS 7 (7.8+)

Laden Sie die Version 5.9.0 der Software für CentOS 7 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE-Anbieter (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 7 (7.8+)

Laden Sie die Version 5.9.0 der Software für RHEL 7 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE-Anbieter (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 8 (8.3+)

Laden Sie die Version 5.9.0 der Software für RHEL 8 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 081887f6ea1d9df9d1e409b2b5bde83e965c42229acbeb1f950c8fe478361edc)
- OpenSSL Dynamic Engine (SHA256 checksum 6b0500a42fd57c39f076f14e5079f80145b6ebd2c441395761eb04600c07bda5)
- JCE-Anbieter (SHA256 checksum 2bc7ac26b259af92a65fbd5a30d5eb2a92ce0e70efe41feb53bf82f168aa90bb)
  - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum 79ecbe9b4c5316ccf447d8c59b76b5ac2cc854bd79cd50c1f29197aa8cb080db)

Ubuntu 18.04 LTS

Laden Sie die Version 5.9.0 der Software für Ubuntu 18.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum bc6d2227edd7b5a83fed32741fbacbb1756d5df89ebb3435d96f0609a180db65)
- OpenSSL Dynamic Engine (SHA256 checksum 2d6a26434fa6faf337f1dfb42de033220fa405a82d4540e279639a03b3ee6e9d)
- JCE-Anbieter (SHA256 checksum e12aef122f490e9026452ce31c25625b1accb9a5866b3d470488f10f047f1873)
  - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum f0bcabe594db3e8ff86cc0f65c2a10858d34452eb6b9fc33d7aac05c0f5f4f30)

Ubuntu 20.04 LTS

Laden Sie die Version 5.9.0 der Software für Ubuntu 20.04 LTS auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum 15dde8182f432de9e7d369b05e384e1f2d80dcca85db3b16ecc26cdef1a34bb9)
- OpenSSL Dynamic Engine (SHA256 checksum c8ba94a999038af87d4905b7c1feb4cc87e20d1776a32ef6f6d11ee000b5a896)
- JCE-Anbieter (SHA256 checksum de33cd3e8130a06d9da5207079533aac8276a1319ac435a3737b4f65bd8fb972)
  - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum cfa31535ad9a99a5113496c06fbace38e9593491aca9bb031a18b51075973e68)

Windows Server 2016

Laden Sie die Version 5.9.0 der Software für Windows Server 2016 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- JCE-Anbieter (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)

    - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Windows Server 2019

Laden Sie die Version 5.9.0 der Software für Windows Server 2019 auf der x86_64-Architektur herunter:

- PKCS-#11-Bibliothek (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- JCE-Anbieter (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
    - Javadocs für AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM-CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Das Client-SDK 5.9.0 verbessert die Stabilität und beinhaltet Bugfixes für alle SDKs. Für alle SDKs wurde eine Optimierung vorgenommen, sodass Anwendungen sofort über Betriebsfehler informiert werden, wenn festgestellt wird, dass ein HSM nicht verfügbar ist. Diese Version enthält Leistungsverbesserungen für JCE.

JCE-Anbieter

- Verbesserte Leistung
- Ein bekanntes Problem mit der Erschöpfung des Sitzungspools wurde behoben

Version 3.4.4

Um das Client SDK 3 auf Linux-Plattformen zu aktualisieren, müssen Sie einen Batch-Befehl verwenden, der den Client-Daemon und alle Bibliotheken gleichzeitig aktualisiert. Weitere Informationen finden Sie unter Client-SDK-3-Upgrade.

Wählen Sie zum Herunterladen der Software die Registerkarte für das gewünschte Betriebssystem und wählen Sie dann den Link des betreffenden Softwarepakets.

Amazon Linux

Software-Version 3.4.4 für Amazon Linux herunterladen:

- AWS CloudHSM Kunde (SHA256 checksum 900de424d70f41e661aa636f256a6a79cc43bea6b0fe6eb95c2aaa63e5289505)
- PKCS-#11-Bibliothek (SHA256 checksum a3f93f084d59fee5d7c859292bc02cb7e7f15fb06e971171ebf9b52bbd229c30)
- OpenSSL Dynamic Engine (SHA256 checksum 8db07b9843d49016b0b6fec46d39881d94e426fcaae1cee2747be14af9313bb0)
- JCE-Anbieter (SHA256 checksum 360617c55bf4caa8e6e78ede079ca68cf9ef11473e7918154c22ba908a219843)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum c9961ffe38921131bd6f3702e10d73588e68b8ab10fbb241723e676f4fa8c4fa)

Amazon Linux 2

Software-Version 3.4.4 für Amazon Linux 2 herunterladen:

- AWS CloudHSM Kunde (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS-#11-Bibliothek (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE-Anbieter (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 6

AWS CloudHSM unterstützt CentOS 6 nicht mit Client SDK Version 3.4.4.

Verwenden Sie the section called “Version 3.2.1” für CentOS 6 oder wählen Sie eine unterstützte Plattform.

CentOS 7 (7.8+)

Software-Version 3.4.4 für CentOS 7 herunterladen:

- AWS CloudHSM Kunde (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS-#11-Bibliothek (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE-Anbieter (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 8

Software-Version 3.4.4 für CentOS 8 herunterladen:

- AWS CloudHSM Kunde (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- PKCS-#11-Bibliothek (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- JCE-Anbieter (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

> **Note**
>
> Aufgrund des kürzlichen Auslaufs von CentOS 8 werden wir diese Plattform mit der nächsten Version nicht mehr unterstützen können.

RHEL 6

AWS CloudHSM unterstützt RedHat Enterprise Linux 6 mit Client SDK Version 3.4.4 nicht.

Verwenden Sie es the section called “Version 3.2.1” für RedHat Enterprise Linux 6 oder wählen Sie eine unterstützte Plattform.

RHEL 7 (7.8+)

Laden Sie die Version 3.4.4 der Software für RedHat Enterprise Linux 7 herunter:

- AWS CloudHSM Kunde (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS-#11-Bibliothek (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE-Anbieter (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

RHEL 8 (8.3+)

Laden Sie die Version 3.4.4 der Software für RedHat Enterprise Linux 8 herunter:

- AWS CloudHSM Kunde (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- PKCS-#11-Bibliothek (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- JCE-Anbieter (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

Ubuntu 16.04 LTS

Software-Version 3.4.4 für Ubuntu 16.04 LTS herunterladen:

- AWS CloudHSM Kunde (SHA256 checksum 317c92c2e0b5d60afab1beb947f053d13ddaacb994cccc2c2b898e997ece29b9)
- PKCS-#11-Bibliothek (SHA256 checksum 91451c420c51488a022569fd32f052a3b988a2883ea4c2ac952acb61a2fea37c)
- OpenSSL Dynamic Engine (SHA256 checksum 4098771ad0e38df9bf14d50520ca49b9395f819f0387e2bc3b0e61abb5888e66)
- JCE-Anbieter (SHA256 checksum e136ff183271c2f9590a9fccb8261a7eb809506686b070e3854df1b8686c6641)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum cbf24a4032f393a913a9898b1b27036392104e8e05d911cab84049b2bcca2541)

> **Note**
>
> Aufgrund des bevorstehenden EOL von Ubuntu 16.04 beabsichtigen wir, die Unterstützung für diese Plattform mit der nächsten Version einzustellen.

Ubuntu 18.04 LTS

Software-Version 3.4.4 für Ubuntu 18.04 LTS herunterladen:

- AWS CloudHSM Kunde (SHA256 checksum cf57d5e0e95efbf032aac8887aebd59ac8cc80e97c69e7c39fdad40873374fe8)
- PKCS-#11-Bibliothek (SHA256 checksum 428f8bdad7925db5401112f707942ee8f3ca554f4ab53fa92237996e69144d2f)
- JCE-Anbieter (SHA256 checksum 1ff17b8f7688e84f7f0bfc96383564dca598a1cab2f2c52c888d0361682f2b9e)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum afe253046146ed6177c520b681efc680dac1048c4a95b3d8ad0f305e79bbe93e)

Windows Server

AWS CloudHSM unterstützt 64-Bit-Versionen von Windows Server 2012, Windows Server 2012 R2, Windows Server 2016 und Windows Server 2019. Die AWS CloudHSM 3.4.4-Clientsoftware für Windows Server umfasst die erforderlichen CNG- und KSP-Anbieter. Einzelheiten finden Sie unter Installation und Konfiguration des AWS CloudHSM Clients (Windows). Laden Sie die aktuelle Version (3.4.4) der Software für Windows Server herunter:

- AWS CloudHSM für Windows Server (SHA256 checksum d51a7db588e9121d8f0b0351606bd986e1c4de6547f2c8235200dc8a5ffbe53e)
- AWS CloudHSM Verwaltungsdienstprogramm (SHA256 checksum 0c12d7da9086735cdf189535937a8e036163009c5018dcaf2ee9cdbb6bd4c06f)

Version 3.4.4 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Client-Software

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.17.1.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Veraltete Versionen

Versionen 5.8.0 und früher sind veraltet. Wir raten davon ab, veraltete Versionen in Produktionsumgebungen zu verwenden. Wir stellen keine abwärtskompatiblen Updates für veraltete Versionen bereit und stellen auch keine veralteten Versionen zum Herunterladen bereit. Wenn Sie bei der Verwendung veralteter Versionen Auswirkungen auf die Produktion feststellen, müssen Sie ein Upgrade durchführen, um Softwarekorrekturen zu erhalten.

Veraltete Versionen des Client SDK 5

In diesem Abschnitt werden veraltete Versionen des Client SDK 5 aufgeführt.

Version 5.8.0

Version 5.8.0 bietet Quorum-Authentifizierung für CloudHSM CLI, SSL/TLS-Offload mit JSSE, Multi-Slot-Unterstützung für PKCS #11, Multi-Cluster-/Multi-Benutzer-Unterstützung für JCE, Schlüsselextraktion mit JCE, unterstütztes keyFactory für JCE, neue Wiederholungskonfigurationen für Rückgabecodes ohne Terminal und enthält verbesserte Stabilität und Bugfixes für alle SDKs.

PKCS-#11-Bibliothek

- Unterstützung für die Konfiguration mit mehreren Steckplätzen wurde hinzugefügt.

JCE-Anbieter

- Konfigurationsbasierte Schlüsselextraktion hinzugefügt.
- Unterstützung für Konfigurationen mit mehreren Clustern und mehreren Benutzern hinzugefügt.
- Unterstützung für SSL- und TLS-Offload mit JSSE hinzugefügt.
- Unwrap-Unterstützung für AES/CBC/ wurde hinzugefügt. NoPadding
- Neue Typen von Schlüsselfabriken hinzugefügt: und. SecretKeyFactory KeyFactory

CloudHSM-CLI

- Unterstützung für Quorum-Authentifizierung hinzugefügt

Version 5.7.0

Version 5.7.0 führt die CloudHSM-CLI ein und enthält einen neuen verschlüsselten CMAC-Algorithmus (Message Authentication Code). Diese Version fügt die ARM-Architektur auf Amazon Linux 2 hinzu. Javadocs des JCE-Anbieters sind jetzt verfügbar für AWS CloudHSM.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.
- Wird jetzt auf der ARM-Architektur mit Amazon Linux 2 unterstützt.
- Algorithmen
  - CKM_AES_CMAC (signieren und verifizieren)

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.
- Wird jetzt auf der ARM-Architektur mit Amazon Linux 2 unterstützt.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.
- Algorithmen
  - AESCMAC

Version 5.6.0

Version 5.6.0 beinhaltet die Unterstützung neuer Mechanismen für die PKCS-#11-Bibliothek und den JCE-Anbieter. Darüber hinaus unterstützt Version 5.6 Ubuntu 20.04.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.
- Mechanismen
  - CKM_RSA_X_509, für die Modi Verschlüsseln, Entschlüsseln, Signieren und Verifizieren

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.
- Verschlüsselungen
  - RSA/ECB/, für Verschlüsselungs- und NoPadding Entschlüsselungsmodi

Unterstützte Schlüssel

- EC mit den Kurven secp224r1 und secp521r1

Plattformunterstützung

- Unterstützung für Ubuntu 20.04 hinzugefügt.

Version 5.5.0

Version 5.5.0 bietet Unterstützung für OpenJDK 11, Keytool- und Jarsigner-Integration sowie zusätzliche Mechanismen für den JCE-Anbieter. Behebt ein bekanntes Problem, bei dem eine KeyGenerator Klasse den Schlüsselgrößenparameter fälschlicherweise als Anzahl von Bytes statt als Bits interpretiert.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Unterstützung für die Dienstprogramme Keytool und Jarsigner
- Unterstützung für OpenJDK 11 auf allen Plattformen
- Verschlüsselungen
  - AES/CBC/Verschlüsselungs- und NoPadding Entschlüsselungsmodus
  - Verschlüsselungs- und Entschlüsselungsmodus für AES/ECB/PKCS5Padding
  - NoPadding AES/CTR/Verschlüsselungs- und Entschlüsselungsmodus
  - AES/GCM/ Wrap- und Unwrap-Modus NoPadding
  - Verschlüsselungs- und Entschlüsselungsmodus für DESede/ECB/PKCS5Padding
  - DeSede/CBC/Verschlüsselungs- und Entschlüsselungsmodus NoPadding
  - NoPadding AESWrap/ECB/Wrap- und Unwrap-Modus
  - Entpack- und Einpack-Modus für AESWrap/ECB/PKCS5Padding
  - ZeroPadding AESWrap/ECB/Wrap- und Unwrap-Modus
  - Entpack- und Einpack-Modus für RSA/ECB/PKCS1Padding
  - Entpack- und Einpack-Modus für RSA/ECB/OAEPPadding
  - Entpack- und Einpack-Modus für RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
  - Entpack- und Einpack-Modus für RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - Entpack- und Einpack-Modus für RSA/ECB/OAEPWithSHA-256ANDMGF1Padding
  - Entpack- und Einpack-Modus für RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - Entpack- und Einpack-Modus für RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
  - Entpack- und Einpack-Modus für RSAAESWrap/ECB/OAEPPadding
  - Entpack- und Einpack-Modus für RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding
  - Entpack- und Einpack-Modus für RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding

  - Entpack- und Einpack-Modus für RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding
  - Entpack- und Einpack-Modus für RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding
  - Entpack- und Einpack-Modus für RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding
- KeyFactory und SecretKeyFactory
  - RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.
  - AES – 128-, 192- und 256-Bit-AES-Schlüssel
  - EC-Schlüsselpaare für die NIST secp256r1- (P-256), secp384r1- (P-384) und secp256k1-Kurven.
  - DESede (3DES)
  - GenericSecret
  - HMAC – mit SHA1-, SHA224-, SHA256-, SHA384- und SHA512-Hash-Unterstützung
- Signieren/Überprüfen
  - RSASSA-PSS
  - SHA1withRSA/PSS
  - SHA224withRSA/PSS
  - SHA256withRSA/PSS
  - SHA384withRSA/PSS
  - SHA512withRSA/PSS
  - SHA1withRSAandMGF1
  - SHA224withRSAandMGF1
  - SHA256withRSAandMGF1
  - SHA384withRSAandMGF1
  - SHA512withRSAandMGF1

Version 5.4.2

Version 5.4.2 enthält verbesserte Stabilität und Bugfixes für alle SDKs. Dies ist auch die letzte Version für die CentOS-8-Plattform. Weitere Informationen finden Sie auf der CentOS-Website.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.4.1

Version 5.4.1 behebt ein bekanntes Problem mit der PKCS-#11-Bibliothek. Dies ist auch die letzte Version für die CentOS-8-Plattform. Weitere Informationen finden Sie auf der CentOS-Website.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.4.0

Version 5.4.0 bietet anfängliche Unterstützung für den JCE-Anbieter für alle Plattformen. Der JCE-Anbieter ist mit OpenJDK 8 kompatibel.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Schlüsseltypen
  - RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.
  - AES – 128-, 192- und 256-Bit AES-Schlüssel.
  - ECC-Schlüsselpaare für die NIST secp256r1- (P-256), secp384r1- (P-384) und secp256k1-Kurven.
  - DESede (3DES)
  - HMAC – mit SHA1-, SHA224-, SHA256-, SHA384- und SHA512-Hash-Unterstützung
- Chiffren (nur verschlüsseln und entschlüsseln)
  - AES/GCM/ NoPadding
  - AES/EZB/ NoPadding
  - AES/CBC/PKCS5Padding
  - Desede/EZB/ NoPadding
  - DESede/CBC/PKCS5Padding
  - AES/CTR/ NoPadding
  - RSA/ECB/PKCS1Padding
  - RSA/ECB/OAEPPadding
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-256ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
- Digests
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- Signieren/Überprüfen
  - NONEwithRSA

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

- Integration mit Java KeyStore

Version 5.3.0

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Unterstützung für ECDSA-Signieren/Verifizieren mit den Kurven P-256, P-384 und secp256k1 hinzugefügt.
- Unterstützung für folgende Plattformen hinzufügen: Amazon Linux, Amazon Linux 2, Centos 7.8+, RHEL 7 (7.8+).
- Unterstützung für OpenSSL Version 1.0.2 hinzugefügt.
- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Schlüsseltypen
  - RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.
  - AES – 128-, 192- und 256-Bit AES-Schlüssel.

  - EC-Schlüsselpaare für die NIST secp256r1- (P-256), secp384r1- (P-384) und secp256k1-Kurven.
  - DESede (3DES)
  - HMAC – mit SHA1-, SHA224-, SHA256-, SHA384- und SHA512-Hash-Unterstützung
- Chiffren (nur verschlüsseln und entschlüsseln)
  - AES/GCM/ NoPadding
  - AES/EZB/ NoPadding
  - AES/CBC/PKCS5Padding
  - Desede/EZB/ NoPadding
  - DESede/CBC/PKCS5Padding
  - AES/CTR/ NoPadding
  - RSA/ECB/PKCS1Padding
  - RSA/ECB/OAEPPadding
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-256ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
- Digests
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- Signieren/Überprüfen
  - NONEwithRSA
  - SHA1withRSA
  - SHA224withRSA
  - SHA256withRSA
  - SHA384withRSA

- SHA512withRSA
- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA
- Integration mit Java KeyStore

Version 5.2.1

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.2.0

Version 5.2.0 erweitert die PKCS-#11-Bibliothek um Unterstützung für zusätzliche Schlüsseltypen und Mechanismen.

PKCS-#11-Bibliothek

Schlüsseltypen

- ECDSA – Kurven P-224, P-256, P-384, P-521 und secp256k1
- Triple DES (3DES)

Mechanismen

- CKM_EC_KEY_PAIR_GEN
- CKM_DES3_KEY_GEN
- CKM_DES3_CBC

- CKM_DES3_CBC_PAD
- CKM_DES3_ECB
- CKM_ECDSA
- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512
- CKM_RSA_PKCS für Verschlüsseln/Entschlüsseln

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.1.0

Mit Version 5.1.0 wird die PKCS-#11-Bibliothek um zusätzliche Mechanismen erweitert.

PKCS-#11-Bibliothek

Mechanismen

- CKM_RSA_PKCS für einpacken/entpacken
- CKM_RSA_PKCS_PSS
- CKM_SHA1_RSA_PKCS_PSS
- CKM_SHA224_RSA_PKCS_PSS
- CKM_SHA256_RSA_PKCS_PSS
- CKM_SHA384_RSA_PKCS_PSS
- CKM_SHA512_RSA_PKCS_PSS
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_SP800_108_COUNTER_KDF

- CKM_GENERIC_SECRET_KEY_GEN
- CKM_SHA_1_HMAC
- CKM_SHA224_HMAC
- CKM_SHA256_HMAC
- CKM_SHA384_HMAC
- CKM_SHA512_HMAC
- CKM_RSA_PKCS_OAEP nur einpacken/entpacken
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

API-Operationen

- C_ CreateObject
- C_ DeriveKey
- C_ WrapKey
- C_ UnWrapKey

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

Version 5.0.1

Version 5.0.1 fügt anfängliche Unterstützung für OpenSSL Dynamic Engine hinzu.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Erste Version von OpenSSL Dynamic Engine.

- Diese Version bietet Unterstützung für Einsteiger für Schlüsseltypen und OpenSSL-APIs:
  - RSA-Schlüsselgenerierung für 2048-, 3072- und 4096-Bit-Schlüssel
  - OpenSSL APIs:
    - RSA Sign mit RSA PKCS mit SHA1/224/256/384/512 und RSA PSS
    - Generierung von RSA-Schlüsseln

  Weitere Informationen finden Sie unter OpenSSL Dynamic Engine.
- Unterstützte Plattformen: CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3+ und Ubuntu 18.04 LTS
  - Benötigt: OpenSSL 1.1.1

  Weitere Informationen finden Sie unter Unterstützte Plattformen.
- Unterstützung für SSL/TLS Offload auf CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3 und Ubuntu 18.04 LTS, einschließlich NGINX 1.19 (für ausgewählte Cipher Suites).

  Weitere Informationen erhalten Sie unter Verwendung der SSL-/TLS-Auslagerung unter Linux.

Version 5.0.0

Version 5.0.0 ist die erste Version.

PKCS-#11-Bibliothek

- Dies ist die Erstversion.

Unterstützung für Einsteiger der PKCS-#11-Bibliothek in Client SDK Version 5.0.0

Dieser Abschnitt beschreibt die Unterstützung für Schlüsseltypen, Mechanismen, API-Operationen und Attribute Client-SDK Version 5.0.0.

Schlüsseltypen:

- AES – 128-, 192- und 256-Bit-AES-Schlüssel
- RSA – 2048-Bit- bis 4096-Bit-RSA-Schlüssel, in Schritten von je 256 Bit.

Mechanismen:

- CKM_AES_GCM

- CKM_AES_KEY_GEN
- CKM_CLOUDHSM_AES_GCM
- CKM_RSA_PKCS
- CKM_RSA_X9_31_KEY_PAIR_GEN
- CKM_SHA1
- CKM_SHA1_RSA_PKCS
- CKM_SHA224
- CKM_SHA224_RSA_PKCS
- CKM_SHA256
- CKM_SHA256_RSA_PKCS
- CKM_SHA384
- CKM_SHA384_RSA_PKCS
- CKM_SHA512
- CKM_SHA512_RSA_PKCS

API-Operationen:

- C_ CloseAllSessions
- C_ CloseSession
- C_Decrypt
- C_ DecryptFinal
- C_ DecryptInit
- C_ DecryptUpdate
- C_ DestroyObject
- C_Digest
- C_ DigestFinal
- C_ DigestInit
- C_ DigestUpdate
- C_Encrypt
- C_ EncryptFinal

- C_ EncryptInit
- C_ EncryptUpdate
- C_Finalize
- C_ FindObjects
- C_ FindObjectsFinal
- C_ FindObjectsInit
- C_ GenerateKey
- C_ GenerateKeyPair
- C_ GenerateRandom
- C_ GetAttributeValue
- C_ GetFunctionList
- C_ GetInfo
- C_ GetMechanismInfo
- C_ GetMechanismList
- C_ GetSessionInfo
- C_ GetSlotInfo
- C_ GetSlotList
- C_ GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_ OpenSession
- C_Sign
- C_ SignFinal
- C_ SignInit
- C_ SignUpdate
- C_Verify
- C_ VerifyFinal
- C_ VerifyInit
- C_ VerifyUpdate

Attribute:

- GenerateKeyPair
  - Alle RSA-Schlüsselattribute
- GenerateKey
  - Alle AES-Schlüsselattribute
- GetAttributeValue
  - Alle RSA-Schlüsselattribute
  - Alle AES-Schlüsselattribute

Beispiele:

- Schlüssel generieren (AES, RSA, EC)
- Schlüsselattribute auflisten
- Verschlüsseln und Entschlüsseln von Daten mit AES GCM
- Signieren und Verifizieren von Daten mit RSA

Veraltete Versionen des Client SDK 3

In diesem Abschnitt werden veraltete Client SDK 3-Versionen aufgeführt.

Version 3.4.3

Version 3.4.3 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Client-Software

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.17.0.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.4.2

Version 3.4.2 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.16.0.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.4.1

Version 3.4.1 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.15.0.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.4.0

Version 3.4.0 enthält Aktualisierungen für alle Komponenten.

AWS CloudHSM Software für Kunden

- Verbesserte Stabilität und Fehlerbehebungen.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

Windows (CNG- und KSP-Anbieter)

- Verbesserte Stabilität und Fehlerbehebungen.

Version 3.3.2

Version 3.3.2 behebt ein Problem mit dem Skript client_info.

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.3.1

Version 3.3.1 enthält Aktualisierungen für alle Komponenten.

AWS CloudHSM Software für Kunden

- Verbesserte Stabilität und Fehlerbehebungen.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.

Windows (CNG- und KSP-Anbieter)

- Verbesserte Stabilität und Fehlerbehebungen.

Version 3.3.0

Version 3.3.0 fügt die Zwei-Faktor-Authentifizierung (2FA) und weitere Verbesserungen hinzu.

AWS CloudHSM Software für Kunden

- 2FA-Authentifizierung für Crypto Officer (CO) hinzugefügt. Weitere Informationen finden Sie unter Verwalten der Zwei-Faktor-Authentifizierung für Crypto Officer.
- Die Plattformunterstützung für RedHat Enterprise Linux 6 und CentOS 6 wurde entfernt. Weitere Informationen finden Sie unter Linux-Support.
- Es wurde eine eigenständige Version von CMU zur Verwendung mit Client-SDK 5 oder Client-SDK 3 hinzugefügt. Dies ist dieselbe CMU-Version, die im Client-Daemon der Version 3.3.0 enthalten ist. Jetzt können Sie CMU herunterladen, ohne den Client-Daemon herunterzuladen.

PKCS-#11-Bibliothek

- Verbesserte Stabilität und Fehlerbehebungen.
- Die Plattformunterstützung für RedHat Enterprise Linux 6 und CentOS 6 wurde entfernt. Weitere Informationen finden Sie unter Linux-Support.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.
- Die Plattformunterstützung für RedHat Enterprise Linux 6 und CentOS 6 wurde entfernt. Weitere Informationen finden Sie unter Linux-Support.

JCE-Anbieter

- Verbesserte Stabilität und Fehlerbehebungen.
- Die Plattformunterstützung für RedHat Enterprise Linux 6 und CentOS 6 wurde entfernt. Weitere Informationen finden Sie unter Linux-Support.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.2.1

Version 3.2.1 fügt eine Konformitätsanalyse zwischen der AWS CloudHSM Implementierung der PKCS #11 -Bibliothek und des PKCS #11 -Standards, neuen Plattformen und anderen Verbesserungen hinzu.

AWS CloudHSM Software für den Kunden

- Plattformunterstützung für CentOS 8, RHEL 8 und Ubuntu 18.04 LTS hinzugefügt. Weitere Informationen finden Sie unter ???.

PKCS-#11-Bibliothek

- Bericht zur Einhaltung der PKCS-#11-Bibliothek für das Client-SDK 3.2.1
- Plattformunterstützung für CentOS 8, RHEL 8 und Ubuntu 18.04 LTS hinzugefügt. Weitere Informationen finden Sie unter ???.

OpenSSL Dynamic Engine

- Keine Unterstützung für CentOS 8, RHEL 8 und Ubuntu 18.04 LTS. Weitere Informationen finden Sie unter ???.

JCE-Anbieter

- Plattformunterstützung für CentOS 8, RHEL 8 und Ubuntu 18.04 LTS hinzugefügt. Weitere Informationen finden Sie unter ???.

Windows (CNG- und KSP-Anbieter)

- Verbesserte Stabilität und Fehlerbehebungen.

Version 3.2.0

Version 3.2.0 bietet Unterstützung für das Maskieren von Passwörtern und andere Verbesserungen.

AWS CloudHSM Software für Kunden

- Fügt Unterstützung für das Verbergen Ihres Passworts hinzu, wenn Sie Befehlszeilentools verwenden. Weitere Informationen finden Sie unter loginHSM und logoutHSM (cloudhsm_mgmt_util) und loginHSM und logoutHSM (key_mgmt_util).

PKCS-#11-Bibliothek

- Integriert die Unterstützung für das Hashing großer Datenmengen in Software für einige PKCS-#11-Mechanismen, die zuvor nicht unterstützt wurden. Weitere Informationen finden Sie unter unterstütze Mechanismen.

OpenSSL Dynamic Engine

- Verbesserte Stabilität und Fehlerbehebungen.

JCE-Anbieter

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Windows (CNG- und KSP-Anbieter)

- Verbesserte Stabilität und Fehlerbehebungen.

Version 3.1.2

Version 3.1.2 enthält Aktualisierungen für den JCE-Anbieter.

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Aktualisieren Sie log4j auf Version 2.13.3.

Windows (CNG- und KSP-Anbieter)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.1.1

AWS CloudHSM Software für Kunden

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

PKCS-#11-Bibliothek

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

OpenSSL Dynamic Engine

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

JCE-Anbieter

- Fehlerbehebungen und Leistungsverbesserungen.

Windows (CNG, KSP)

- Versionsnummer wurde aus Konsistenzgründen aktualisiert.

Version 3.1.0

Version 3.1.0 fügt das standardkonforme AES Key Wrapping hinzu.

AWS CloudHSM Software für Kunden

- Eine neue Upgrade-Anforderung: Die Version Ihres Clients muss mit der Version aller von Ihnen verwendeten Softwarebibliotheken übereinstimmen. Für ein Upgrade müssen Sie einen Stapelbefehl verwenden, der gleichzeitig den Client und alle Bibliotheken aktualisiert. Weitere Informationen finden Sie unter Client-SDK-3-Upgrade.
- Key_mgmt_util (KMU) enthält folgende Aktualisierungen:
  - Es wurden zwei neue AES-Verschlüsselungsmethoden hinzugefügt: standardkonforme AES-Verschlüsselung mit Zero Padding und AES-Verschlüsselung ohne Padding. Weitere Informationen finden Sie unter wrapKey oder unwrapKey.
  - Deaktivierte Funktion, einen benutzerdefinierten IV anzugeben, wenn ein Schlüssel mithilfe von AES_KEY_WRAP_PAD_PKCS5 verpackt wird. Weitere Informationen finden Sie unter AES Key Wrapping.

PKCS-#11-Bibliothek

- Es wurden zwei neue AES-Verschlüsselungsmethoden hinzugefügt: standardkonforme AES-Verschlüsselung mit Zero Padding und AES-Verschlüsselung ohne Padding. Weitere Informationen finden Sie unter AES Key Wrapping.
- Sie können die Salt-Länge für RSA-PSS-Signaturen konfigurieren. Informationen zur Verwendung dieser Funktion finden Sie unter Konfigurierbare Salzlänge für RSA-PSS-Signaturen auf. GitHub

OpenSSL Dynamic Engine

- ABWÄRTSKOMPATIBLE ÄNDERUNG: TLS 1.0 und 1.2 Cipher Suites mit SHA1 sind in OpenSSL Engine 3.1.0 nicht verfügbar. Dieses Problem wird in Kürze behoben.
- Wenn Sie beabsichtigen, die OpenSSL Dynamic Engine-Bibliothek auf RHEL 6 oder CentOS 6 zu installieren, informieren Sie sich über ein bekanntes Problem bezüglich der OpenSSL-Standardversion, die auf diesen Betriebssystemen installiert ist.
- Stabilitätsverbesserungen und Fehlerbehebungen

JCE-Anbieter

- ABWÄRTSKOMPATIBLE ÄNDERUNG: Um ein Problem mit der Java Cryptography Extension (JCE)-Konformität zu beheben, verwenden die AES-Verschlüsselung und -Entschlüsselung jetzt anstelle des AES-Algorithmus den AESWrap-Algorithmus korrekt. Dies bedeutet, dass `Cipher.WRAP_MODE` und `Cipher.UNWRAP_MODE` nicht mehr mit den AES/EZB- und AES/CBC-Mechanismen funktionieren.

  Um ein Upgrade auf die Client-Version 3.1.0 durchzuführen, müssen Sie Ihren Code aktualisieren. Wenn Sie bereits verpackte Schlüssel haben, müssen Sie besonders auf den Mechanismus zum Entpacken achten und darauf, wie sich die IV-Standardwerte geändert haben. Wenn Sie mit der Client-Version 3.0.0 oder früher verpackte Schlüssel haben, müssen Sie in 3.1.1 AesWrap/ECB/PKCS5Padding verwenden, um die vorhandenen Schlüssel zu entpacken. Weitere Informationen finden Sie unter AES Key Wrapping.
- Sie können mehrere Schlüssel mit demselben Label aus der JCE-Anbieter auflisten. Informationen zur Iteration aller verfügbaren Schlüssel finden Sie unter Suchen Sie nach allen Schlüsseln auf. GitHub
- Sie können bei der Schlüsselerstellung restriktivere Werte für Attribute festlegen, einschließlich der Angabe unterschiedlicher Labels für öffentliche und private Schlüssel. Weitere Informationen finden Sie unter Unterstützte Java-Attribute.

Windows (CNG, KSP)

- Verbesserte Stabilität und Fehlerbehebungen.

E-Veröffentlichungen nd-of-life

AWS CloudHSM kündigt das Ende der Lebensdauer von Versionen an, die nicht mehr mit dem Service kompatibel sind. Um die Sicherheit Ihrer Anwendung zu gewährleisten, behalten wir uns das Recht vor, Verbindungen zu end-of-life Releases aktiv abzulehnen.

- Derzeit sind keine Versionen des Client-SDK end-of-life Releases.

Dokumentverlauf

In diesem Thema werden wichtige Aktualisierungen im AWS CloudHSM -Benutzerhandbuch beschrieben.

Themen

Neueste Aktualisierungen

Die folgende Tabelle enthält signifikante Änderungen an dieser Dokumentation seit April 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 signifikante Änderungen benachrichtigt werden möchten, verwenden Sie den Link oben rechts, um den RSS-Feed zu abonnieren.

Einzelheiten zu neuen Versionen finden Sie unter Downloads für das AWS CloudHSM Client-SDK.

Änderung	Beschreibung	Datum
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.12.0 veröffentlicht.	20. März 2024
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.11.0 veröffentlicht.	17. Januar 2024
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.10.0 veröffentlicht.	28. Juli 2023
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.9.0 veröffentlicht.	23. Mai 2023
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.8.0 veröffentlicht.	16. März 2023

Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.7.0 veröffentlicht.	16. November 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.6.0 veröffentlicht.	01. September 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.5.0 veröffentlicht.	13. Mai 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.4.2 veröffentlicht.	18. März 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.4.1 veröffentlicht.	10. Februar 2022
Neue Version hinzugefügt	Die AWS CloudHSM JCE-Provider-Version 5.4.0 für Windows-Plattformen wurde veröffentlicht.	1. Februar 2022
Neue Version hinzugefügt	Die AWS CloudHSM Client-Version 5.4.0 wurde veröffentlicht, die anfängliche Unterstützung für den JCE-Anbieter für alle Linux-Plattformen bietet.	28. Januar 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.3.0 veröffentlicht.	3. Januar 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.4 veröffentlicht.	3. Januar 2022
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.3 veröffentlicht.	20. Dezember 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.2 veröffentlicht.	15. Dezember 2021

Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.1 veröffentlicht.	10. Dezember 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.2.1 veröffentlicht.	4. Oktober 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.4.0 veröffentlicht.	25. August 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.2.0 veröffentlicht.	3. August 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.3.2 veröffentlicht.	2. Juli 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.1.0 veröffentlicht.	1. Juni 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.3.1 veröffentlicht.	26. April 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.0.1 veröffentlicht.	8. April 2021
Neue Version hinzugefügt	AWS CloudHSM Client-Version 5.0.0 veröffentlicht.	12. März 2021
Zusätzliche neue Inhalte	Schnittstellen-VPC-Endpunkt hinzugefügt, eine AWS-Funktion, mit der Sie eine private Verbindung zwischen Ihrer VPC herstellen können, AWS CloudHSM ohne dass ein Zugriff über das Internet oder über ein NAT-Gerät, eine VPN-Verbindung oder eine AWS Direct Connect Verbindung erforderlich ist.	10. Februar 2021

Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.3.0 veröffentlicht.	3. Februar 2021
Zusätzliche neue Inhalte	Es wurde die verwaltete Aufbewahrung von Sicherungen hinzugefügt, eine Funktion, die alte Sicherungen automatisch löscht.	18. November 2020
Zusätzliche neue Inhalte	Es wurde ein Konformitätsbericht hinzugefügt, der die Implementierung der PKCS #11 -Bibliothek im AWS CloudHSM Client-SDK 3.2.1 mit dem PKCS #11 -Standard analysiert.	29. Oktober 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.2.1 veröffentlicht.	8. Oktober 2020
Zusätzliche neue Inhalte	Es wurde eine Dokumentation hinzugefügt, die die wichtigsten Synchronisationseinstellungen in AWS CloudHSM beschreibt.	1. September 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.2.0 veröffentlicht.	31. August 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.1.2 veröffentlicht.	30. Juli 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.1.1 veröffentlicht.	3. Juni 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.1.0 veröffentlicht.	21. Mai 2020

Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.0.1 veröffentlicht.	20. April 2020
Neue Version hinzugefügt	AWS CloudHSM Client-Version 3.0.0 für die Windows Server-Plattform veröffentlicht.	30. Oktober 2019
Neue Version hinzugefügt	Veröffentlichte AWS CloudHSM Client-Version 3.0.0 für alle Plattformen außer Windows.	22. Oktober 2019
Neue Version hinzugefügt	AWS CloudHSM Client-Version 2.0.4 veröffentlicht.	26. August 2019
Neue Version hinzugefügt	AWS CloudHSM Client-Version 2.0.3 veröffentlicht.	13. Mai 2019
Neue Version hinzugefügt	AWS CloudHSM Client-Version 2.0.1 veröffentlicht.	21. März 2019
Neue Version hinzugefügt	AWS CloudHSM Client-Version 2.0.0 veröffentlicht.	6. Februar 2019
Zusätzlicher Support für Regionen	AWS CloudHSM Unterstützung für die Regionen EU (Stockholm) und AWS GovCloud (USA-Ost) hinzugefügt.	19. Dezember 2018
Neue Version hinzugefügt	AWS CloudHSM Client-Version 1.1.2 für Windows veröffentlicht.	20. November 2018
Bekannte Probleme aktualisiert	Neue Inhalte wurden dem Handbuch zur Fehlerbehebung hinzugefügt.	8. November 2018

Neue Version hinzugefügt	AWS CloudHSM Client-Version 1.1.2 für Linux-Plattformen veröffentlicht.	8. November 2018
Zusätzlicher Support für Regionen	AWS CloudHSM Unterstützung für die Regionen EU (Paris) und Asien-Pazifik (Seoul) hinzugefügt.	24. Oktober 2018
Zusätzliche neue Inhalte	Es wurde die Möglichkeit hinzugefügt, AWS CloudHSM Backups zu löschen und wiederherzustellen.	10. September 2018
Zusätzliche neue Inhalte	Automatische Übermittlung von Auditprotokollen zu Amazon CloudWatch Logs hinzugefügt.	13. August 2018
Zusätzliche neue Inhalte	Es wurde die Möglichkeit hinzugefügt, ein AWS CloudHSM Cluster-Backup regionsübergreifend zu kopieren.	30. Juli 2018
Zusätzlicher Support für Regionen	AWS CloudHSM Unterstützung für die Region EU (London) hinzugefügt.	13. Juni 2018
Zusätzliche neue Inhalte	AWS CloudHSM Client- und Bibliotheksunterstützung für Amazon Linux 2, Red Hat Enterprise Linux (RHEL) 6, Red Hat Enterprise Linux (RHEL) 7, CentOS 6, CentOS 7 und Ubuntu 16.04 LTS hinzugefügt.	10. Mai 2018

Neue Version hinzugefügt	AWS CloudHSM Ein Windows-Client wurde hinzugefügt.	30. April 2018

Frühere Aktualisierungen

In der folgenden Tabelle werden die wichtigen Änderungen gegenüber der AWS CloudHSM Version vor 2018 beschrieben.

Änderung	Beschreibung	Datum
Neuer Inhalt	Quorum-Authentifizierung für Verschlüsselungsverantwortliche (Crypto Officer, COs) hinzugefügt. Weitere Informationen finden Sie unter Verwendung von CloudHSM Management Utility (CMU) zur Verwaltung der Quorum-Authentifizierung (M-von-N-Zugriffskontrolle).	9. November 2017
Aktualisierung	Dokumentation zur Verwendung des Befehlszeilen-Tools key_mgmt_util hinzugefügt. Weitere Informationen finden Sie unter key_mgmt_util-Befehlsreferenz.	9. November 2017
Neuer Inhalt	Oracle Transparent Data Encryption hinzugefügt. Weitere Informationen finden Sie unter Oracle Database-Verschlüsselung.	25. Oktober 2017

Änderung	Beschreibung	Datum
Neuer Inhalt	SSL-Auslagerung hinzugefügt. Weitere Informationen finden Sie unter SSL/TLS-Auslagerung.	12. Oktober 2017
Neues Handbuch	Diese Version stellt vor AWS CloudHSM	14. August 2017

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.