



AWS KMS 暗号化の詳細

AWS Key Management Service



AWS Key Management Service: AWS KMS 暗号化の詳細

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

Table of Contents

序章	1
概念	2
設計目標	4
AWS Key Management Service の基礎	6
暗号化の基本	6
エントロピーと乱数生成	6
対称キーのオペレーション (暗号化のみ)	6
非対称キーのオペレーション (暗号化、デジタル署名、署名の検証)	7
キー導出関数	7
AWS KMS 内部での デジタル署名の使用	7
エンベロープ暗号化	8
AWS KMS key 階層	8
ユースケース	11
EBS ボリュームの暗号化	11
クライアント側の暗号化	13
AWS KMS keys	15
呼び出し CreateKey	16
キーマテリアルのインポート	18
呼び出し ImportKeyMaterial	18
キーの有効化と無効化	19
キーの削除	19
キーマテリアルのローテーション	20
顧客データオペレーション	21
データキーの生成	21
暗号化	23
Decrypt	24
暗号化されたオブジェクトの再暗号化	25
AWS KMS 内部オペレーション	28
ドメインとドメインの状態	28
ドメインキー	29
エクスポートされたドメイントークン	29
ドメイン状態の管理	30
内部通信セキュリティ	32
キー確立	32

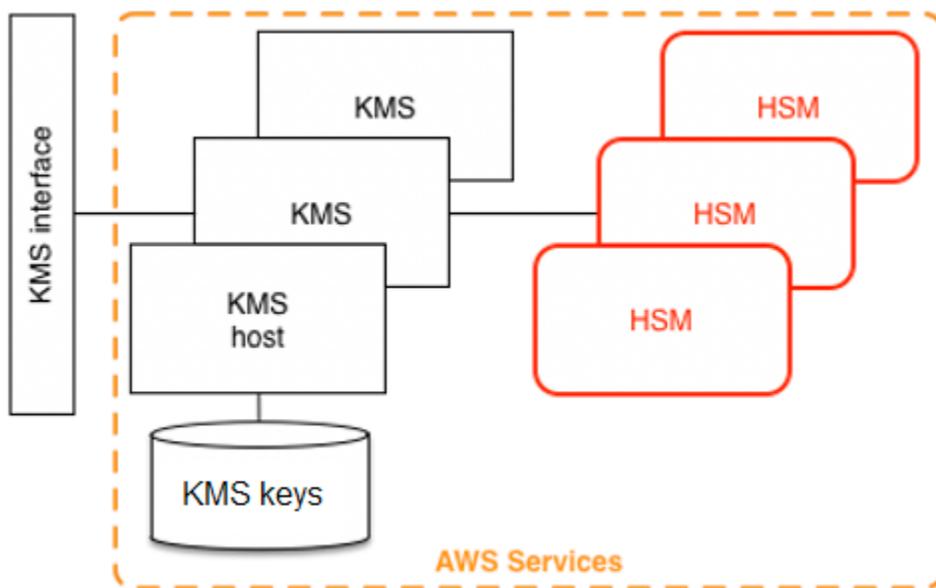
HSM セキュリティ境界	33
定足数署名付きコマンド	33
認証済みセッション	34
マルチリージョンキーのレプリケーションプロセス	35
耐久性の保護	36
リファレンス	37
省略形	37
キー	38
寄稿者	39
Bibliography	40
ドキュメント履歴	42
.....	xliii

AWS KMS の暗号化の詳細説明

AWS Key Management Service (AWS KMS) には暗号化キーを生成および管理するためのウェブインターフェイスがあり、データを保護するための暗号化サービスプロバイダーとして機能します。AWS KMS は、AWS のサービスと統合された従来型のキー管理サービスを備え、AWS 全体でお客様のキーが一貫して見えるようになっています。また、一元管理および監査機能を備えています。このホワイトペーパーでは、AWS KMS の暗号化オペレーションを詳細に説明し、このサービスが提供する機能の評価に役立つようにしています。

AWS KMS には、AWS Management Console、コマンドラインインターフェイス、および RESTful API オペレーションによるウェブインターフェイスが含まれ、FIPS 140-2 検証済みのハードウェアセキュリティモジュール (HSM) [1] の分散フリートに対する暗号化オペレーションをリクエストできます。AWS KMS HSM は、AWS KMS のセキュリティおよびスケーラビリティ要件を満たす専用の暗号化機能を提供するように設計されたマルチチップのスタンドアロンハードウェア暗号化アプライアンスです。AWS KMS keys として管理しているキーで、独自の HSM ベースの暗号化階層を確立できます。これらのキーは HSM 上でのみ使用でき、暗号化リクエストの処理に必要な時間だけメモリ内にあります。複数の KMS キーを作成できます。各キーはキー ID で表されます。それぞれのお客様が管理する AWS IAM ロールとアカウントでのみ、お客様の KMS キーを作成、削除、またはデータの暗号化、復号化、署名、検証をするために使用できます。キーにアタッチされたポリシーを作成することで、KMS キーを管理および/または使用できるユーザーに関するアクセス制御を定義できます。このようなポリシーを使用すると、API オペレーションごとにキーにアプリケーション固有の使用を定義できます。

加えて、ほとんどの AWS のサービスは、KMS キーを使用した保管中のデータの暗号化をサポートします。この機能により、お客様は、KMS キーへのアクセス方法とタイミングを制御することで、AWS のサービスが暗号化されたデータにアクセスする方法とタイミングを制御できます。



AWS KMS は、ウェブとの接点である AWS KMS ホストと階層化された HSM で構成された階層型サービスです。これらの階層型ホストのグループ化は、AWS KMS スタックを形成します。AWS KMS へのすべてのリクエストは、Transport Layer Security プロトコル (TLS) を介して行われ、AWS KMS ホスト上で終了する必要があります。AWS KMS ホストは、perfect [forward secrecy](#) を提供する暗号スイートを使用する TLS のみを許可します。AWS KMS では、他のすべての AWS API オペレーションに利用可能な AWS Identity and Access Management (IAM) と同じ認証情報およびポリシーメカニズムを使用して、リクエストを認証および承認します。

基本概念

AWS Key Management Service を最大限理解するのに役立つ、基本的な用語と概念について説明します。

AWS KMS key

i Note

AWS KMS では、カスタマーマスターキー (CMK) という用語が AWS KMS key および KMS キーに置き換えられています。この概念に変更はありません。AWS KMS では、互換性を破る変更を避けるため、この用語にいくつかのバリエーションがあります。

キー階層の最上位を表す論理キーです。KMS キーには、一意のキー識別子またはキー ID を含む Amazon リソースネーム (ARN) が与えられます。AWS KMS keys には、次の 3 つのタイプがあります。

- カスタマーマネージド型キー – お客様が作成およびライフサイクルとキーポリシーの管理を行います。これらのキーに対して行われたすべてのリクエストは、CloudTrail イベントとして記録されます。
- AWS マネージドキー – AWS が作成およびライフサイクルポリシーとキーポリシーの管理を行います。AWS マネージドキーは、お客様の AWS アカウントのリソースです。お客様はのアクセスポリシーと CloudTrail イベントを表示できますが AWS マネージドキー、これらのキーのどの側面も管理することはできません。これらのキーに対して行われたすべてのリクエストは、CloudTrail イベントとして記録されます。
- AWS 所有のキー – これらのキーは AWS によって作成され、さまざまな AWS サービスで内部の暗号化オペレーションに使用されます。お客様は、のキーポリシーや AWS 所有のキー使用状況を可視化できません CloudTrail。

エイリアス

KMS キーに関連付けられているわかりやすい名前です。エイリアスは、多くの AWS KMS API オペレーションでキー ID と同じように使用できます。

アクセス許可

キーに対するアクセス許可を定義する KMS キーにアタッチされたポリシーです。デフォルトのポリシーでは、お客様が定義するすべてのプリンシパルが許可されます。また AWS アカウントによる、キーを参照する IAM ポリシーの追加も許可されます。

権限

最初に目的の IAM プリンシパルまたは使用期間が知られていないため、キーや IAM ポリシーに追加できない場合、KMS キーの使用のために委任されるアクセス許可です。許可の使用方の 1 つは、AWS サービスにおける KMS キーの使用法についてのスコープダウンされたアクセス許可を定義することです。サービスでは、直接署名された API コールがない場合、ユーザーに代わり暗号化されたデータに対して非同期作業を行うためにキーを使用することがあります。

データキー

HSM で生成され、KMS キーによって保護される暗号化キーです。AWS KMS では、許可されたエンティティにより KMS キーで保護されるデータキーを取得できます。これらは、プレーンテキスト (暗号化されていない) データキーおよび暗号化されたデータキーの両方として返すことができます。データキーは、対称キーでも非対称キーでも構いません (パブリックおよびプライベートの両方の部分が返されます)。

暗号文

AWS KMS の暗号化された出力です。混乱を解消するため、顧客の暗号文と呼ばれることもあります。暗号文には、追加情報を含む暗号化されたデータが含まれています。この情報により、復号化のプロセスで使用する KMS キーを識別できます。暗号化されたデータキーは KMS キーを使用する際に生成される暗号文の一般的な例の 1 つですが、サイズが 4 KB 未満のデータは KMS キーで暗号化し、暗号文を生成できます。

暗号化コンテキスト

AWS KMS で保護されている情報に関連付けられた、追加情報のキーと値のペアマップです。AWS KMS では、データキーが保護に認証済みの暗号化が使用されます。暗号化コンテキストは、AWS KMS で暗号化された暗号文で、認証済みの暗号化の AAD に組み込まれます。このコンテキスト情報はオプションで、キー (または暗号化オペレーション) のリクエスト時には返されません。使用する場合、このコンテキスト値は復号オペレーションを正常に完了するために必要です。暗号化コンテキストの使用目的の 1 つは、追加の認証情報を提供することです。この情報は、ポリシーを適用し、AWS CloudTrail ログに含めるのに役立ちます。例えば、{"key name": "satellite uplink key"} のキーと値のペアを使用して、データキーに名前を付けられます。その後のキーが使用されると、"key name": "satellite uplink key" を含む AWS CloudTrail エントリが作成されます。この追加情報は、特定の KMS キーが使用された理由を理解するのに役立つコンテキストを提供します。

パブリックキー

非対称暗号 (RSA または楕円曲線) を使用する場合、パブリックキーはパブリック/プライベートのキーペアの "public component" です。パブリックキーは、パブリック/プライベートのキーペアの所有者のデータを暗号化するエンティティに共有および分散できます。デジタル署名オペレーションでは、パブリックキーを使用して署名を検証できます。

プライベートキー

非対称暗号 (RSA または楕円曲線) を使用する場合、プライベートキーはパブリック/プライベートのキーペアの「プライベートコンポーネント」です。プライベートキーは、データの復号またはデジタル署名の作成に使用されます。対称 KMS キーと同様に、プライベートキーは HSM で暗号化されます。これらは、暗号化リクエストの処理に必要な期間、HSM の短期メモリにのみ復号されます。

AWS KMS 設計目標

AWS KMS は、次の要件を満たすように設計されています。

耐久性

暗号化キーの耐久性は、AWS で最高の耐久性を持つサービスと同等に設計されています。1 つの暗号化キーで、長期間にわたって蓄積された大量のデータを暗号化できます。

信頼性

キーの使用は、ユーザーが定義および管理するアクセス制御ポリシーによって保護されます。プレーンテキストの KMS キーをエクスポートするメカニズムはありません。暗号化キーの機密性は重要です。HSM で管理アクションを実行するには、定足数ベースのアクセス制御にロール固有のアクセス権を持つ複数の Amazon 従業員が必要です。

低レイテンシーと高スループット

AWS KMS には、AWS の他のサービスで使用するのに適したレイテンシーとスループットのレベルでの暗号化オペレーションが用意されています。

リージョンの独立性

AWS は、異なるリージョンでデータアクセスを制限する必要があるお客様向けに、リージョンの独立性を確保しています。キーの使用は、AWS リージョン 内に限ることができます。

元になる乱数の安全性

強力な暗号は、真に予測不可能な乱数生成に依存するため、AWS KMS には、高品質で検証済みの乱数のソースが用意されています。

監査

AWS KMS は、暗号化キーの使用と管理を AWS CloudTrail ログに記録します。AWS CloudTrail ログを使用して、AWS のサービスによる自身のための使用を含め、暗号化キーの使用状況を調べることができます。

これらの目標を達成するために、AWS KMS システムには、「ドメイン」を管理する一連の AWS KMS オペレーターとサービスホストオペレーター (総称して「オペレーター」) が含まれています。ドメインとは、リージョンに定義された AWS KMS サーバー、HSM、およびオペレーターのセットです。各 AWS KMS オペレーターは、アクションを認証するために使用されるプライベートとパブリックのキーペアを含むハードウェアトークンを持っています。HSM には、HSM 状態の同期を保護する暗号化キーを確立するために、追加のプライベートとパブリックのキーペアがあります。

このホワイトペーパーでは、暗号化するキーやその他のデータを AWS KMS が保護する方法を説明します。このドキュメントでは、暗号化する暗号化キーまたはデータを「シークレット」または「シークレットマテリアル」と呼びます。

AWS Key Management Service の基礎

この章のトピックでは、AWS Key Management Service の暗号化の基本およびそれらを使用する場所について説明します。また、AWS KMS の基本的な要素についても紹介します。

トピック

- [暗号化の基本](#)
- [AWS KMS key 階層](#)

暗号化の基本

AWS KMS では、承認されたアルゴリズムまたはモードから別のアルゴリズムにすばやくシステムを移行できるように、設定可能な暗号化アルゴリズムを使用します。初期の暗号化アルゴリズムのデフォルトセットは、安全性とパフォーマンスのため、連邦情報処理標準 (FIPS 承認) のアルゴリズムから選択されています。

エントロピーと乱数生成

AWS KMS のキー生成は、AWS KMS HSM で実行されます。HSM では、[AES-256 を使った NIST SP800-90A 決定論的ランダムビットジェネレーター \(DRBG\) CTR_DRBG](#) を使用したハイブリッド乱数生成機能を実装しています。非決定論的ランダムビットジェネレーターは 384 ビットのエントロピーでシードされ、追加のエントロピーで更新されます。これにより、暗号化マテリアルの呼び出しごとに予測抵抗が提供されます。

対称キーのオペレーション (暗号化のみ)

HSM 内で使用されるすべての対称キーの暗号化コマンドには、[高度暗号化規格 \(AES\)](#) で 256 ビットキーの [Galois Counter Mode \(GCM\)](#) を使用します。復号化のための同様の呼び出しには、逆関数を使用します。

AES-GCM は認証済みの暗号化スキームです。プレーンテキストの暗号化による暗号文の生成に加えて、暗号文と認証が必要な追加データ (追加で認証されたデータ、または AAD) に対する認証タグを計算します。認証タグは、データが意図されたソースからのものであり、暗号文および AAD が変更されていないことを確認するのに役立ちます。

AWS では、特にデータキーの暗号化を参照する際、AAD の説明が頻繁に省略されます。これらのケースでは周囲のテキストにより、暗号化される構造が、暗号化されるプレーンテキストと保護されるクリアテキスト AAD の間で分割されることが暗示されます。

AWS KMS では、AWS KMS に依存することなく AWS KMS key にキーマテリアルをインポートするオプションが提供されています。[RSAES-OAEP](#) または [RSAES-PKCS1-v1_5](#) を使用して、このインポートされたキーマテリアルを暗号化することで、AWS KMS HSM への転送中のキーを保護できます。RSA キーペアは、AWS KMS HSM で生成されます。インポートされたキーマテリアルは、AWS KMS HSM で復号化され、サービスに保存される前に、AES-GCM で再暗号化されます。

非対称キーのオペレーション (暗号化、デジタル署名、署名の検証)

AWS KMS では、暗号化およびデジタル署名のオペレーションの両方で非対称キーオペレーションの使用をサポートしています。非対称キーのオペレーションは、数学的に関連するパブリックキーとプライベートキーのペアに依存します。これらは暗号化および復号化、または署名およびその検証に使用できますが、両方には使用できません。プライベートキーが暗号化されないまま AWS KMS から出ていくことはありません。AWS KMS API オペレーションを呼び出すことによって、AWS KMS 内でパブリックキーを使用することも、パブリックキーをダウンロードして AWS KMS の外部で使用することもできます。

AWS KMS では、2 種類の非対称暗号がサポートされています。

- RSA-OAEP (暗号化用) および RSA-PSS/RSA-PKCS-#1-v1_5 (署名および検証用) – さまざまなセキュリティ要件に対応できるように、RSA キーの長さは 2048、3072、4096 (ビット単位) がサポートされています。
- 楕円曲線 (ECC) – 署名と検証にのみ使用できます。ECC 曲線 NIST P256、P384、P521、SECP 256k1 がサポートされています。

キー導出関数

キー導出関数は、最初のシークレットまたはキーから追加のキーを取得するために使用されます。AWS KMS では、キー導出関数 (KDF) を使用して、AWS KMS key での暗号化の呼び出しごとにキーを取得します。すべての KDF オペレーションでは、SHA256 [\[FIPS180\]](#) で HMAC [\[FIPS197\]](#)、[カウンタモードの KDF](#) を使用します。256 ビットの導出キーは AES-GCM とともに使用され、顧客データとキーを暗号化または復号化します。

AWS KMS 内部での デジタル署名の使用

デジタル署名は、AWS KMS エンティティ間のコマンドや通信を認証するためにも使用されます。すべてのサービスエンティティには、楕円曲線デジタル署名アルゴリズム (ECDSA) のキーペアがあります。[Use of Elliptic Curve Cryptography \(ECC\) Algorithms in Cryptographic Message Syntax \(CMS\)](#) および X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve

Digital Signature Algorithm (ECDSA) で定義されているように、これらにより ECDSA が実行されます。エンティティでは、[Federal Information Processing Standards Publications, FIPS PUB 180-4](#) で定義されている、SHA384 と呼ばれる安全なハッシュアルゴリズムを使用します。キーは、曲線 `secp384r1` (NIST-P384) 上で生成されます。

エンベロープ暗号化

エンベロープ暗号化は、多くの暗号化システムで使用されている基本的な構造です。エンベロープ暗号化では、2 つ以上の暗号化キーを使用してメッセージを保護します。通常、キーのうち 1 つはより長期的な静的キー `k` から取得します。もう 1 つはメッセージの暗号化のために生成されるメッセージごとのキー `msgKey` です。エンベロープは、メッセージ `ciphertext = Encrypt(msgKey, message)` の暗号化により形成されます。その後、長期的な静的キー `encKey = Encrypt(k, msgKey)` でメッセージキーが暗号化されます。最後に、2 つの値 (`encKey, ciphertext`) が、1 つの構造またはエンベロープで暗号化されたメッセージにパッケージ化されます。

`k` へのアクセス権を持つ受信者は、最初に暗号化されたキーを復号化してからメッセージを復号化することで、エンベロープされたメッセージを開くことができます。

AWS KMS では、これらの長期的な静的キーを管理し、データをエンベロープ暗号化するプロセスを自動化する機能を提供します。

[AWS Encryption SDK](#) では、AWS KMS サービス内で提供される暗号化機能に加えて、クライアント側のエンベロープ暗号化ライブラリを提供します。これらのライブラリを使用して、データとその暗号化に使用している暗号化キーを保護できます。

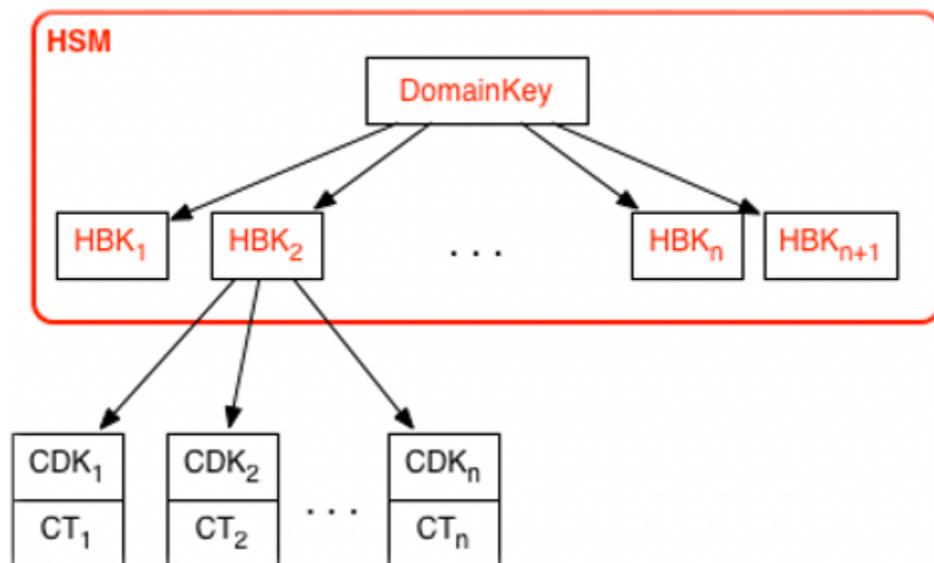
AWS KMS key 階層

キー階層は、最上位レベルの論理キーである AWS KMS key から始まります。KMS キーは、最上位のキーマテリアルのコンテナを表し、Amazon リソースネーム (ARN) を持つ AWS のサービス名前空間内で一意に定義されます。ARN には、一意に生成されたキー識別子である キー ID が含まれています。KMS キーは、AWS KMS を介してユーザーによって開始されたリクエストに基づいて作成されます。受信時に、AWS KMS は、KMS キーコンテナに配置する初期 HSM バックアップキー (HBK) の作成を要求します。HBK は、ドメイン内の HSM 上で生成され、プレーンテキストで HSM からエクスポートされないように設計されています。そうではなく、HBK は HSM 管理ドメインキーで暗号化されてエクスポートされます。これらのエクスポートされた HBK は、エクスポートされたキートークン (EKT) と呼ばれます。

EKT は、耐久性に優れた低レイテンシーのストレージにエクスポートされます。例えば、論理 KMS キーへの ARN を受け取ったとします。これは、キー階層、つまり暗号化コンテキストの最上位を表

します。アカウント内に複数の KMS キーを作成し、他の AWS の名前付きリソースと同様に KMS キーにポリシーを設定できます。

特定の KMS キーの階層内では、HBK は KMS キーのバージョンと考えることができます。KMS キーを AWS KMS でローテーションする場合、新しい HBK が作成され、KMS キーのアクティブな HBK として KMS キーに関連付けられます。古い HBK は保持され、以前に保護されていたデータの復号化と検証に使用できます。ただし、新しい情報を保護するために使用できるのはアクティブな暗号化キーだけです。



KMS キーを使用して情報を直接保護することを AWS KMS を介してリクエストしたり、KMS キーで保護された追加の HSM 生成キーをリクエストしたりできます。これらのキーは、お客様データキー、または CDK と呼ばれます。CDK は、暗号文 (CT) として暗号化して、プレーンテキストで、またはその両方で返すことができます。KMS キーで暗号化されたすべてのオブジェクト (お客様から提供されたデータまたは HSM によって生成されたキー) は、HSM でのみ、AWS KMS を介した呼び出しによって復号化できます。

返された暗号文または復号化されたペイロードは、AWS KMS に保存されることはありません。この情報は、AWS KMS への TLS 接続を介してユーザーに直接返されます。これは、ユーザーに代わって AWS のサービスによって行われた呼び出しにも適用されます。

次の表に、キーの階層と特定のキープロパティを示します。

キー	説明	ライフサイクル
ドメインキー	HSM のメモリ内のみにある 256 ビットの AES-GCM キー。KMS キー	毎日ローテーション ¹

キー	説明	ライフサイクル
	のバージョン (HSM バックアップキー) をまとめるために使用されます。	
HSM バックアップキー	256 ビットの対称キー、または、RSA または楕円曲線のプライベートキー。顧客データとキーを保護するために使用され、ドメインキーの下に暗号化されて保存されます。1 つ以上の HSM バックアップキーで、keyId で表される KMS キーが構成されます。	毎年ローテーション ² (設定はオプション)。
派生暗号化キー	お客様データとキーを暗号化するために使用される HSM のメモリ内のみにある 256 ビット AES-GCM キー。暗号化ごとに HBK から派生します。	暗号化ごとに 1 回使用され、復号時に再生成されます
お客様データキー	プレーンテキストおよび暗号文で HSM からエクスポートされたユーザー定義の対称キーまたは非対称キー。 HSM バックアップキーで暗号化され、TLS チャネル経由で許可されたユーザーに返されます。	アプリケーションによって制御されるローテーションと使用

¹ AWS KMS は、ドメインの管理および設定のタスクを考慮して、ドメインキーのローテーションを最大 1 週間まで緩和することがあります。

² ユーザーに代わって AWS KMS によって作成および管理されるデフォルトの AWS マネージドキーは 1 年に一度、自動的にローテーションされます。

AWS KMS ユースケース

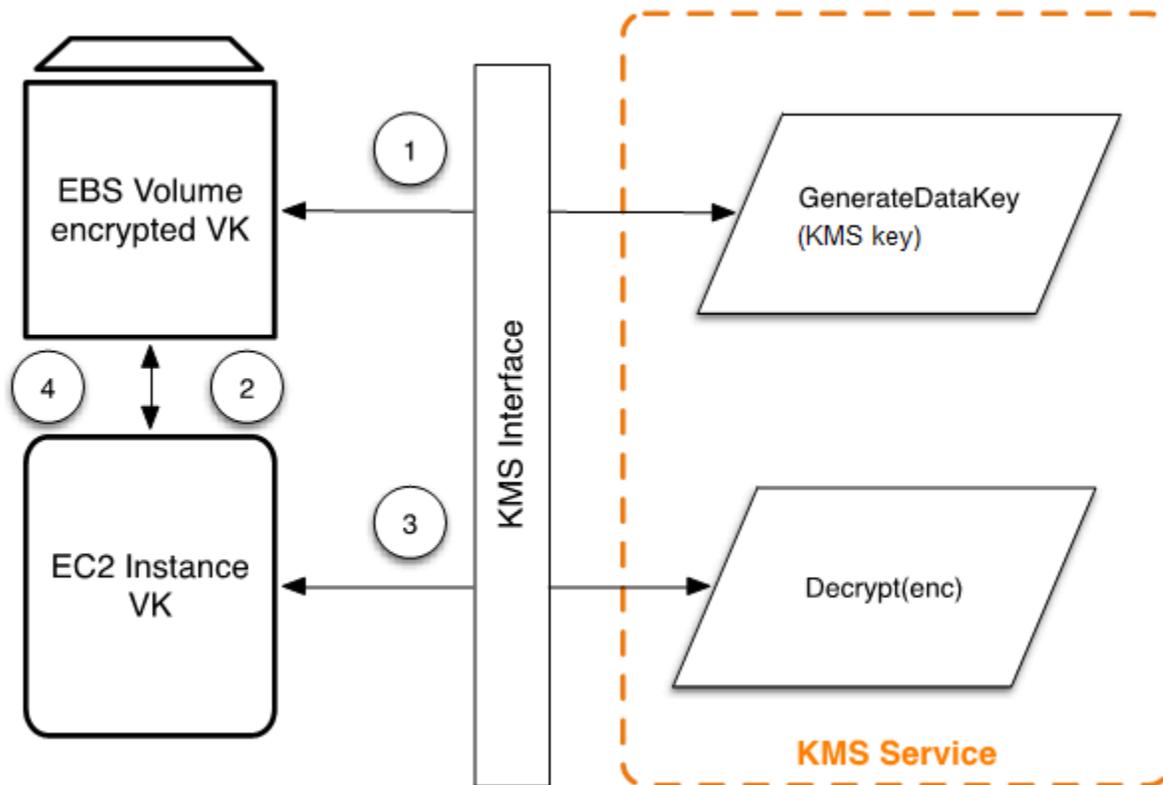
ユースケースは、を最大限に活用するのに役立ちます AWS Key Management Service。最初のは、が Amazon Elastic Block Store (Amazon EBS) ボリューム AWS KMS keys で を使用してサーバー側の暗号化 AWS KMS を実行する方法を示しています。2 つ目は、エンベロープ暗号化を使用してでコンテンツを保護する方法を示すクライアント側のアプリケーションです AWS KMS。

トピック

- [Amazon EBS ボリュームの暗号化](#)
- [クライアント側の暗号化](#)

Amazon EBS ボリュームの暗号化

Amazon EBS には、ボリューム暗号化機能が用意されています。各ボリュームは [AES-256-XTS](#) を使用して暗号化されています。これには 2 つの 256 ビットボリュームキーが必要です。これは、1 つの 512 ビットボリュームキーと考えることができます。ボリュームキーは、アカウントの KMS キーで暗号化されます。Amazon EBS でボリュームを暗号化するには、アカウントの KMS キーの下にボリュームキー (VK) を生成するためのアクセス権が必要です。これを行うには、データキーを作成し、これらのボリュームキーを暗号化および復号化するために、Amazon EBS に KMS キーへのアクセス権を付与します。Amazon EBS は、KMS キー AWS KMS で を使用して AWS KMS 暗号化されたボリュームキーを生成するようになりました。



次のワークフローでは、Amazon EBS ボリュームに書き込まれるデータが暗号化されます。

1. Amazon EBS は、TLS セッション AWS KMS を介して を介して KMS キーで暗号化されたボリュームキーを取得し、ボリュームメタデータとともに暗号化されたキーを保存します。
2. Amazon EBS ボリュームがマウントされると、暗号化されたボリュームキーが取得されます。
3. TLS AWS KMS 経由の の呼び出しは、暗号化されたボリュームキーを復号するために行われます。 は KMS キー AWS KMS を識別し、フリートの HSM に内部リクエストを実行して、暗号化されたボリュームキーを復号します。 AWS KMS その後、 は、TLS セッション経由でインスタンスを含む Amazon Elastic Compute Cloud (Amazon EC2) ホストにボリュームキーを返します。
4. ボリュームキーは、アタッチされた Amazon EBS ボリュームとの間で送受信されるすべてのデータを暗号化および復号化するために使用されます。 Amazon EBS は、メモリ内のボリュームキーが使用できなくなった場合に備えて、暗号化されたボリュームキーを保持します。

KMS キーによる Amazon EBS ボリュームの暗号化の詳細については、「AWS Key Management Service デベロッパーガイド」の「[Amazon Elastic Block Store が使用する AWS KMS 方法](#)」および「Amazon [Amazon EC2 ユーザーガイド](#)」の「Amazon EBS 暗号化」を参照してください。

[Amazon EC2](#)

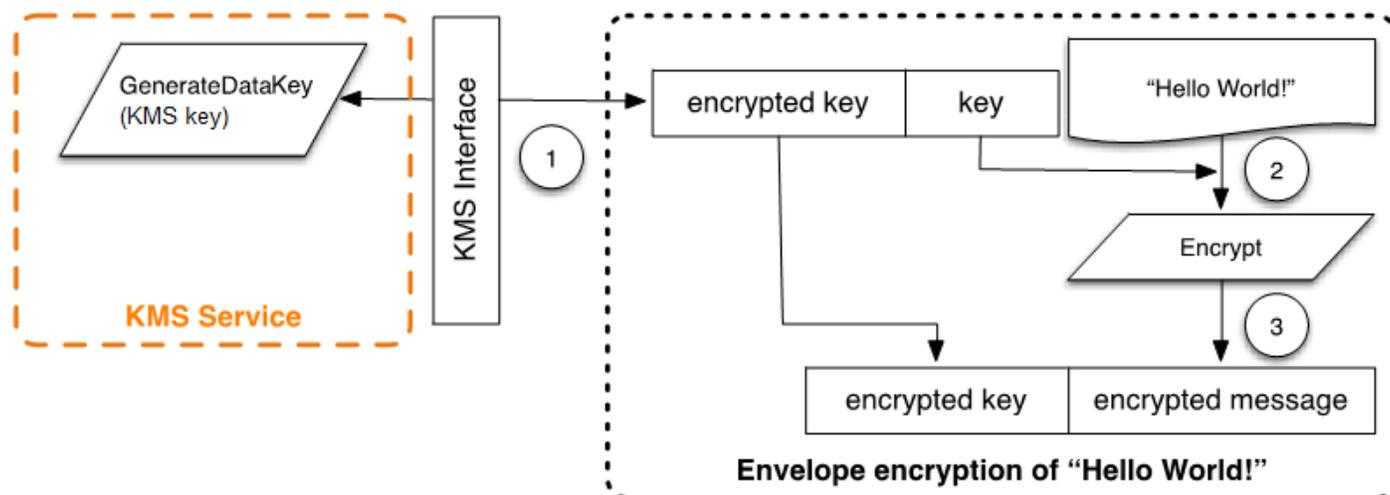
クライアント側の暗号化

[AWS Encryption SDK](#) には、KMS キーを使用してエンベロープ暗号化を実行するための API オペレーションが含まれています。推奨事項と使用方法の詳細については、[関連ドキュメント](#)を参照してください。クライアントアプリケーションは、を使用して AWS Encryption SDK エンベロープ暗号化を実行できます AWS KMS。

```
// Instantiate the SDK
final AwsCrypto crypto = new AwsCrypto();
// Set up the KmsMasterKeyProvider backed by the default credentials
final KmsMasterKeyProvider prov = new KmsMasterKeyProvider(keyId);
// Do the encryption
final byte[] ciphertext = crypto.encryptData(prov, message);
```

クライアントアプリケーションは、次の手順で実行できます。

1. 新しいデータキーのリクエストが KMS キーの下で行われます。暗号化されたデータキーと、データキーのプレーンテキストバージョンが返されます。
2. 内では AWS Encryption SDK、プレーンテキストのデータキーを使用してメッセージを暗号化します。次に、プレーンテキストのデータキーがメモリから削除されます。
3. 暗号化されたデータキーと暗号化されたメッセージは、結合されて単一の暗号テキストのバイト配列になります。



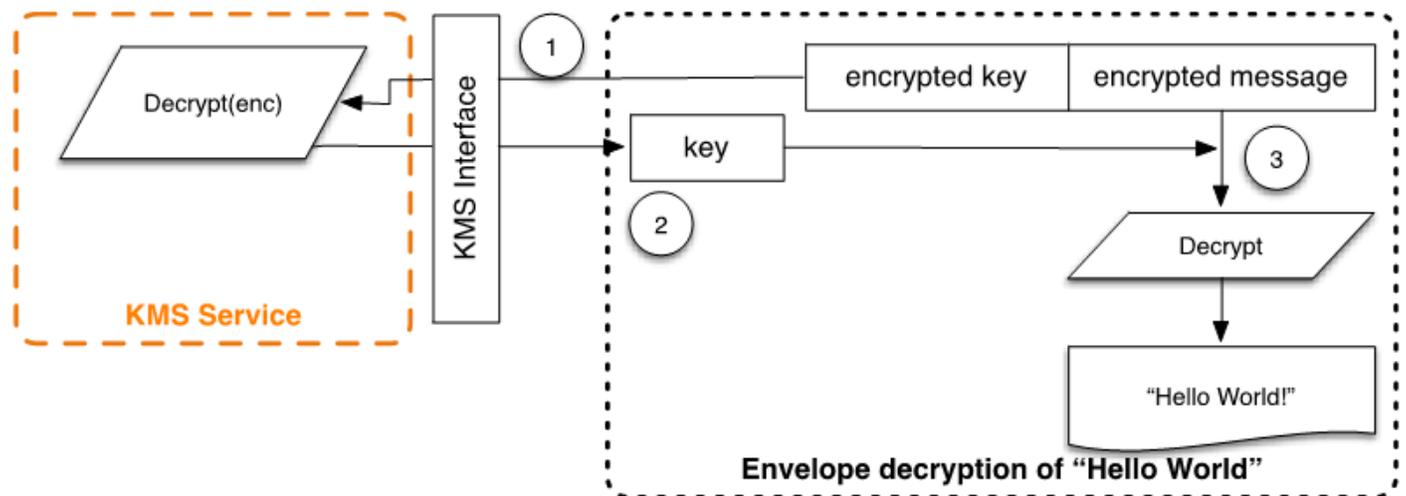
エンベロープ暗号化されたメッセージは、復号化機能を使用して復号化され、最初に暗号化されたメッセージが取得できます。

```

final AwsCrypto crypto = new AwsCrypto();
final KmsMasterKeyProvider prov = new KmsMasterKeyProvider(keyId);
// Decrypt the data
final CryptoResult<byte[], KmsMasterKey> res = crypto.decryptData(prov, ciphertext);
// We need to check the KMS key to ensure that the
// assumed key was used
if (!res.getMasterKeyIds().get(0).equals(keyId)) {
    throw new IllegalStateException("Wrong key id!");
}
byte[] plaintext = res.getResult();

```

1. は、エンベロープで暗号化されたメッセージを AWS Encryption SDK 解析して暗号化されたデータキーを取得し、にデータキーの復号 AWS KMS 化をリクエストします。
2. は、 からプレーンテキストのデータキー AWS Encryption SDK を受け取ります AWS KMS。
3. 次に、データキーを使用してメッセージを復号化し、元のプレーンテキストを返します。



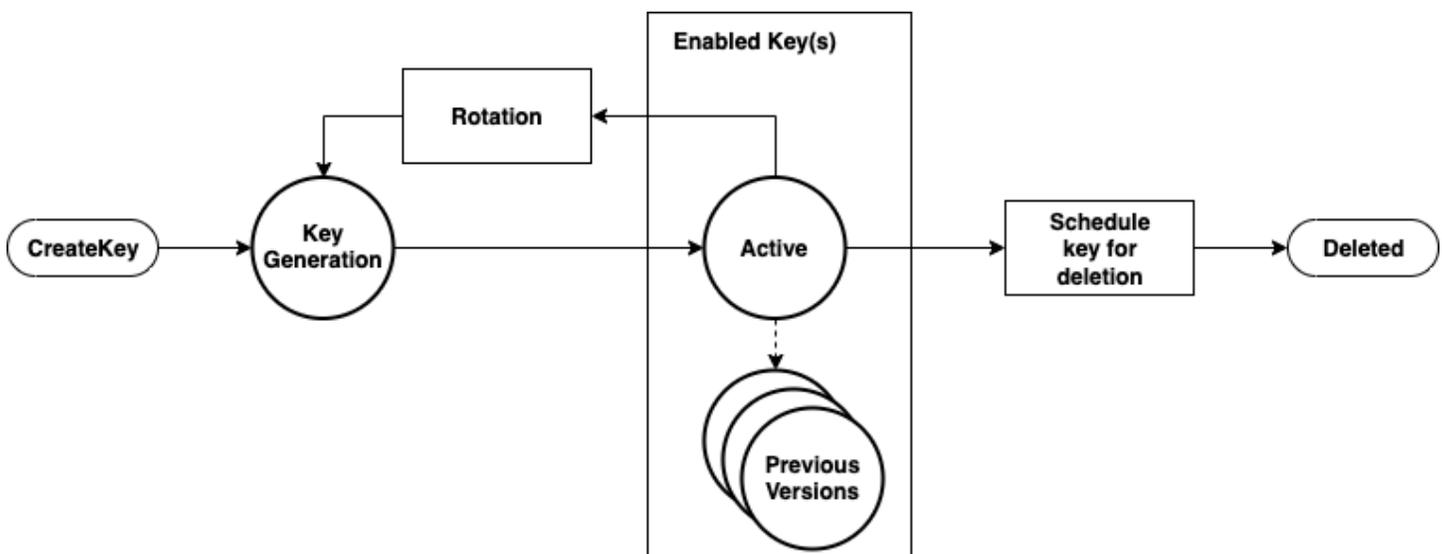
AWS KMS keys の使用

AWS KMS key は、1 つまたは複数の HSM (ハードウェアセキュリティモジュール) バックアップキー (HBK) を参照している可能性がある論理キーを参照します。このトピックでは、KMS キーの作成方法、キーマテリアルのインポート方法、および KMS キーの有効化、無効化、ローテーション、削除方法について説明します。

Note

AWS KMS では、カスターマスターキー (CMK) という用語が AWS KMS key および KMS キーに置き換えられています。この概念に変更はありません。AWS KMS では、互換性を破る変更を避けるため、この用語にいくつかのバリエーションがあります。

この章では、次の図に示すように、作成から削除までの KMS キーのライフサイクルについて説明します。



トピック

- [呼び出し CreateKey](#)
- [キーマテリアルのインポート](#)
- [キーの有効化と無効化](#)
- [キーの削除](#)
- [キーマテリアルのローテーション](#)

呼び出し CreateKey

[CreateKey](#) API コールの呼び出しの結果として AWS KMS key が生成されます。

[CreateKey リクエスト構文](#)のサブセットを次に示します。

```
{
  "Description": "string",
  "KeySpec": "string",
  "KeyUsage": "string",
  "Origin": "string";
  "Policy": "string"
}
```

リクエストは以下の JSON 形式のデータを受け入れます。

説明

(オプション) キーの説明。キーがタスクに適しているかどうかを判断するのに役立つ説明にすることを勧めます。

KeySpec

作成する KMS キーのタイプを指定します。デフォルト値 SYMMETRIC_DEFAULT は、対称暗号化 KMS キーを作成します。このパラメーターは、対称暗号化キーでオプションですが、他のすべてのキー仕様では必須です。

KeyUsage

キーの用途を指定します。有効な値は ENCRYPT_DECRYPT、SIGN_VERIFY、または GENERATE_VERIFY_MAC です。デフォルト値は、ENCRYPT_DECRYPTです。このパラメーターは、対称暗号化キーでオプションですが、他のすべてのキー仕様では必須です。

オリジン

(オプション) KMS キーのキーマテリアルのソースを指定します。デフォルト値は AWS_KMS で、AWS KMS が KMS キーのキーマテリアルの生成と管理を行うことを示します。その他の有効な値には、[インポートされたキーマテリアル](#)のキーマテリアルなしで作成された KMS キーを表す EXTERNAL、およびユーザーが制御する AWS CloudHSM クラスタによってバックアップされる[カスタムキーストア](#)に KSM キーを作成する AWS_CLOUDHSM があります。

ポリシー

(オプション) キーにアタッチするポリシー。ポリシーが省略された場合、キーは次に示すデフォルトのポリシーで作成され、ルートアカウントと AWS KMS プリンシパルを持つ IAM ユーザーがキーを管理できるようになります。

ポリシーの詳細については、「AWS Key Management Service デベロッパーガイド」の「[AWS KMS のキーポリシー](#)」を参照してください。

CreateKey リクエストはキー ARM を含む[レスポンス](#)を返します。

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

Origin が AWS_KMS の場合、ARN が作成された後、認証されたセッションで、AWS KMS HSM に HSM バックアップキー (HBK) のプロビジョニングがリクエストされます。HBK は、KMS キーのキー ID に関連付けられている 256 ビットのキーです。これは HSM 上でのみ生成でき、HSM 境界の外側にクリアテキストでエクスポートされないように設計されています。HBK は、現在のドメインキー DK_0 で暗号化されます。これらの暗号化された HBK は、暗号化キートークン (EKT) と呼ばれます。HSM はさまざまなキーラッピング方法を使用するように設定できますが、現在の実装では、Galois Counter Mode (GCM) の AES-256 と呼ばれる認証された暗号化方式が使用されます。この認証済み暗号化モードでは、クリアテキストでエクスポートされたキートークンのメタデータを保護できます。

これは、スタイル上は次のように表されます。

```
EKT = Encrypt( $DK_0$ , HBK)
```

KMS キーとその後の HBK には、KMS キーに設定された許可ポリシーと、関連付けられた HBK の暗号化保護という 2 つの基礎的な保護方法が用意されています。残りのセクションでは、AWS KMS における暗号化保護と管理機能のセキュリティについて説明します。

ARN に加えて、キーのエイリアスを作成することで、わかりやすい名前を作成して KMS キーに関連付けることができます。エイリアスを KMS キーに関連付けると、エイリアスを使用して、暗号化操作で KMS キーを識別できます。詳細については、AWS Key Management Service デベロッパーガイドの「[エイリアスの使用](#)」を参照してください。

KMS キーの使用には、複数のレベルの認証が必要です。AWS KMS では、暗号化されたコンテンツと KMS キーの承認ポリシーを別々にできます。例えば、AWS KMS インベロープで暗号化された Amazon Simple Storage Service (Amazon S3) オブジェクトは、Amazon S3 バケットのポリシーを

継承します。ただし、必要な暗号化キーへのアクセスは、KMS キーのアクセスポリシーによって決まります。KMS キーの認証の詳細については、AWS Key Management Service デベロッパーガイドの「[AWS KMS の認証とアクセスコントロール](#)」を参照してください。

キーマテリアルのインポート

AWS KMS には、HBK で使用される暗号化マテリアルをインポートするメカニズムがあります。CreateKey 「」で説明されているように呼び出し [CreateKey](#)、を Origin に設定してコマンドを使用すると EXTERNAL、基盤となる HBK を含まない論理 KMS キーが作成されます。暗号化マテリアルは、[ImportKeyMaterial](#) API コールを使用してインポートする必要があります。この機能を使用して、暗号マテリアルのキーの作成と耐久性を制御できます。この機能を使用する場合は、ご使用の環境におけるこれらのキーの取り扱いおよび耐久性に非常に注意することをお勧めします。キーマテリアルのインポートに関する詳細と推奨事項については、AWS Key Management Service デベロッパーガイドの「[Importing key material](#)」を参照してください。

呼び出し ImportKeyMaterial

ImportKeyMaterial リクエストで、HBK に必要な暗号化マテリアルがインポートされます。暗号化マテリアルは 256 ビット対称キーである必要があります。これは、最近の [GetParametersForImport](#) リクエストで返されたパブリックキーを使用して、WrappingAlgorithm で指定されたアルゴリズムを使用して暗号化する必要があります。

[ImportKeyMaterial リクエスト](#) は次の引数を取ります。

```
{
  "EncryptedKeyMaterial": blob,
  "ExpirationModel": "string",
  "ImportToken": blob,
  "KeyId": "string",
  "ValidTo": number
}
```

EncryptedKeyMaterial

GetParametersForImport リクエストで指定されたラッピングアルゴリズムを使用して、そのリクエストで返されたパブリックキーで暗号化されたインポート済みキーマテリアル。

ExpirationModel

キーマテリアルの有効期限が切れているかどうかを指定します。この値が KEY_MATERIAL_EXPIRES の場合は、ValidTo パラメータに有効期限が切れた日を含め

る必要があります。この値が `KEY_MATERIAL_DOES_NOT_EXPIRE` の場合は、`ValidTo` パラメータを含めないでください。有効な値は `"KEY_MATERIAL_EXPIRES"` および `"KEY_MATERIAL_DOES_NOT_EXPIRE"` です。

ImportToken

パブリックキーを提供した同じ `GetParametersForImport` リクエストによって返されるインポートトークン。

KeyId

インポートされたキーマテリアルに関連付けられる KMS キー。KMS キーの `Origin` は、`EXTERNAL` である必要があります。

指定した KMS キーでインポート済みのキーマテリアルを削除して同じキーを再度インポートできますが、その他のキーマテリアルの `KMKMS` キーをインポートすることや関連付けることはできません。

ValidTo

(オプション) インポートされたキーマテリアルの有効期限が切れる時刻。キーマテリアルが有効期限切れになると、AWS KMS はキーマテリアルを削除し、KMS キーは使用不可能になります。このパラメータは、`ExpirationModel` の値が `KEY_MATERIAL_EXPIRES` の場合に必要です。それ以外の場合は、無効です。

リクエストが成功すると、指定された有効期限まで、KMS キーを AWS KMS 内で使用できます (有効期限が指定されている場合)。インポートされたキーマテリアルの有効期限が切れると、EKT は AWS KMS ストレージレイヤーから削除されます。

キーの有効化と無効化

KMS キーを無効にすることで、キーが暗号化オペレーションで使用されるのを防ぎます。KMS キーに関連付けられているすべての HBK を使用する機能が停止されます。有効にすると、HBK と KMS キーの使用が復元されます。[有効化](#)と[無効化](#)は、KMS キーのキー ID またはキー AEN のみを取るシンプルなリクエストです。

キーの削除

承認済みユーザーは、[ScheduleKeyDeletion](#) API を使用して、KMS キーおよび関連付けられたすべての HBK の削除をスケジュールできます。これは本質的に元に戻せない操作であり、AWS

KMS からキーを削除する際は注意が必要です。AWS KMS では、KMS キーの削除時に必ず、少なくとも 7 日間の待機期間を設けています。待機期間中、キーは無効状態になり、キー状態は削除保留中になります。暗号化オペレーションに キーを使用するすべての呼び出しは失敗します。ScheduleKeyDeletion は次の引数を取ります。

```
{
  "KeyId": "string",
  "PendingWindowInDays": number
}
```

KeyId

削除する KMS キーの一意的識別子。この値を指定するには、一意のキー ID または KMS キーのキー ARN を使用します。

PendingWindowInDays

(オプション) 待機期間 (日)。この値はオプションです。範囲は 7 ~ 30 日で、デフォルト値は 30 日です。待機期間が終了した後、AWS KMS は、KMS キーおよび関連するすべての HBK を削除します。

キーマテリアルのローテーション

承認済みユーザーは、カスタマー管理の KMS キーの自動年次ローテーションを有効にできます。AWS マネージドキー は、毎年ローテーションされます。

KMS キーがローテーションされると、新しい HBK が作成され、すべての新しい暗号化リクエストのキーマテリアルの現在のバージョンとしてマークされます。HBK の以前のすべてのバージョンは、この HBK バージョンを使用して暗号化された暗号文を復号化するために引き続き永続的に使用できます。AWS KMS は KMS キーで暗号化された暗号文を格納しないので、古いローテーション済みの HBK で暗号化された暗号文を復号化するには、その HBK が必要です。[ReEncrypt](#) API を使用して KMS キーの新しい HBK で暗号文を再度暗号化すること、またはプレーンテキストを公開することなく別の KMS キーで暗号文を再度暗号化することができます。

キーローテーションの有効化と無効化詳細については、AWS Key Management Service デベロッパーガイドの「[AWS KMS キーのローテーション](#)」を参照してください。

顧客データオペレーション

KMS キーを確立したら、それを使用して暗号化オペレーションを実行できます。KMS キーでデータが暗号化されるたびに、結果として得られるオブジェクトは顧客の暗号文になります。暗号文には、追加の認証データとして認証された暗号化スキームで保護される暗号化されていないヘッダー (またはクリアテキスト) 部分と暗号化された部分の 2 つのセクションがあります。クリアテキストの部分には、HBK 識別子 (HBKID) が含まれます。暗号文の値のこれらの 2 つの不変フィールドにより、今後 AWS KMS でオブジェクトを復号化できるようになります。

トピック

- [データキーの生成](#)
- [暗号化](#)
- [Decrypt](#)
- [暗号化されたオブジェクトの再暗号化](#)

データキーの生成

承認されたユーザーは、GenerateDataKey API (および関連する APIs を使用して、特定のタイプのデータキーまたは任意の長さのランダムキーをリクエストできます。このトピックでは、この API オペレーションの簡略化されたビューを提供します。詳細については、GenerateDataKey APIs AWS Key Management Service」の「API」を参照してください。

- [GenerateDataKey](#)
- [GenerateDataKeyWithoutPlaintext](#)
- [GenerateDataKeyPair](#)
- [GenerateDataKeyPairWithoutPlaintext](#)

次に示すのは、GenerateDataKey のリクエスト構文です。

```
{
  "EncryptionContext": {"string" : "string"},
  "GrantTokens": ["string"],
  "KeyId": "string",
  "NumberOfBytes": "number"
```

```
}
```

リクエストは以下の JSON 形式のデータを受け入れます。

KeyId

データキーの暗号化に使用するキーのキー識別子です。この値は、対称暗号化 KMS キーを識別する必要があります。

このパラメータは必須です。

NumberOfBytes

生成するバイト数を含む整数値です。このパラメータは必須です。

発信者は、KeySpec または NumberOfBytes を指定する必要があります。両方を指定することはできません。

EncryptionContext

(オプション) キーを使用する暗号化および復号化のプロセスで認証する追加データを含む名前と値のペアです。

GrantTokens

(オプション) キーを生成または使用するアクセス許可を提供する権限を示す、許可トークンのリストです。許可や許可トークンの詳細については、AWS Key Management Service デベロッパーガイドの「[Authentication and access control for AWS KMS](#)」を参照してください。

AWS KMS では、コマンドを認証した後、KMS キーに関連する現在アクティブな EKT が取得されます。EKT は、AWS KMS ホストとドメイン内の HSM 間の保護されたセッションを通じて、提供されたリクエストと暗号化コンテキストとともに HSM に渡されます。

HSM は、次の操作を実行します。

1. リクエストされたシークレット材料を生成し、揮発性メモリに保持します。
2. リクエストで定義された KMS キーのキー ID と一致する EKT を復号化し、アクティブな HBK = $\text{Decrypt}(DK_i, \text{EKT})$ を取得します。
3. ランダムなノンズ N を生成します。
4. HBK および N から、256 ビットの AES-GCM 導出暗号化キー K を生成します。
5. シークレット材料 $\text{ciphertext} = \text{Encrypt}(K, \text{context}, \text{secret})$ を暗号化します。

GenerateDataKey は、AWS KMS ホストと HSM 間の安全なチャネルを通じてプレーンテキストのシークレットマテリアルと暗号文をユーザーに返します。その後、それらは AWS KMS により TLS セッションを通じてユーザーに送られます。AWS KMS はプレーンテキストまたは暗号文を保持しません。暗号文、暗号化コンテキスト、KMS キーを使用する権限がない場合、基礎となるシークレットを返すことはできません。

次に示すのは、応答の構文です。

```
{
  "CiphertextBlob": "blob",
  "KeyId": "string",
  "Plaintext": "blob"
}
```

データキーの管理は、アプリケーションデベロッパーに委ねられています。AWS KMS データキーを使用したベストプラクティスクライアントサイド暗号化 (データキーペアではなく) の場合、[AWS Encryption SDK](#) を使用できます。

データキーは任意の頻度でローテーションできます。さらに、データキーは、ReEncrypt API オペレーションを使用して、異なる KMS キーまたはローテーションされる KMS キーに再暗号化することができます。詳細については、AWS Key Management Service API リファレンスの [ReEncrypt](#) 「」を参照してください。

暗号化

AWS KMS の基本的な機能は、KMS キーでオブジェクトを暗号化することです。設計上、AWS KMS では HSM で低レイテンシーの暗号化オペレーションが提供されます。この結果として、暗号化関数への直接の呼び出しで暗号化できるプレーンテキストの量には 4 KB という制限があります。AWS Encryption SDK を使用することで、より大きなメッセージを暗号化できます。AWS KMS では、コマンドを認証した後、KMS キーに関連する現在アクティブな EKT が取得されます。EKT は、プレーンテキストと暗号化コンテキストとともに、リージョン内の利用可能なすべての HSM に渡されます。これらは、AWS KMS ホストとドメイン内の HSM 間の認証されたセッションを通じて送られます。

HSM では、次の操作を実行します。

1. EKT を復号化し、HBK = Decrypt(DK_i, EKT) を取得します。
2. ランダムなノンス N を生成します。

3. HBK および N から、256 ビットの AES-GCM 導出暗号化キー K を取得します。
4. プレーンテキスト ciphertext = Encrypt(K, context, plaintext) を暗号化します。

暗号文の値はユーザーに返され、プレーンテキストのデータおよび暗号文は、AWS インフラストラクチャのどこにも保持されません。暗号文、暗号化コンテキスト、KMS キーを使用する権限がない場合、基礎となるプレーンテキストを返すことはできません。

Decrypt

暗号文の値を復号化する AWS KMS を呼び出すと、暗号化された値の暗号文と暗号化コンテキストが受け入れられます。AWS KMS では、[AWS 署名バージョン 4 の署名付きリクエスト](#)を使用して呼び出しを認証し、暗号文からラッピングキーの HBKID が抽出されます。HBKID は、暗号文、キー ID、およびそのポリシーを復号化するために必要な EKT の取得に使用されます。リクエストは、キーポリシー、存在する可能性のある許可、およびキー ID を参照する関連 IAM ポリシーに基づいて許可されます。Decrypt 関数は、暗号化関数に類似しています。

次に示すのは、Decrypt のリクエスト構文です。

```
{
  "CiphertextBlob": "blob",
  "EncryptionContext": { "string" : "string" }
  "GrantTokens": ["string"]
}
```

次に示すのは、リクエストパラメータです。

CiphertextBlob

メタデータを含む暗号文です。

EncryptionContext

(オプション) 暗号化コンテキストです。これが Encrypt 関数で指定されている場合、ここで指定する必要があります。そうしないと、復号オペレーションが失敗します。詳しくは、AWS Key Management Service デベロッパーガイドの「[Encryption context](#)」を参照してください。

GrantTokens

(オプション) 復号化を実行するためのアクセス許可を提供する権限を示す、許可トークンのリストです。

暗号文と EKT は、認証されたセッションを通じて暗号化コンテキストとともに HSM に送信され、復号化が行われます。

HSM では、次の操作を実行します。

1. EKT を復号化し、 $HBK = \text{Decrypt}(DK_i, EKT)$ を取得します。
2. 暗号文構造から、ノンズ N を抽出します。
3. HBK および N から、256 ビットの AES-GCM 導出暗号化キー K を再生成します。
4. 暗号文を復号化し、 $\text{plaintext} = \text{Decrypt}(K, \text{context}, \text{ciphertext})$ を取得します。

結果のキー ID とプレーンテキストは、安全なセッションを通じて AWS KMS ホストに返され、TLS 接続を通じて呼び出し元の顧客アプリケーションに返されます。

次に示すのは、応答の構文です。

```
{
  "KeyId": "string",
  "Plaintext": blob
}
```

呼び出し元のアプリケーションからプレーンテキストの信頼性を確認したい場合は、返されたキー ID が期待どおりであることを確認する必要があります。

暗号化されたオブジェクトの再暗号化

KMS キーで暗号化された既存の顧客暗号文は、再暗号化コマンドを使用して別の KMS キーに再暗号化できます。再暗号化では、クライアント側にキーのプレーンテキストを公開することなく、サーバー側で新しい KMS キーを使用してデータを暗号化できます。データを復号化してから暗号化します。

次に、リクエスト構文を示します。

```
{
  "CiphertextBlob": "blob",
  "DestinationEncryptionContext": { "string" : "string" },
  "DestinationKeyId": "string",
  "GrantTokens": ["string"],
  "SourceKeyId": "string",
  "SourceEncryptionContext": { "string" : "string" }
```

```
}
```

リクエストは以下の JSON 形式のデータを受け入れます。

CiphertextBlob

再暗号化するデータの暗号文です。

DestinationEncryptionContext

(オプション) データを再暗号化する際に使用される、暗号化コンテキストです。

DestinationKeyId

データの再暗号化に使用されるキーのキー識別子です。

GrantTokens

(オプション) 復号化を実行するためのアクセス許可を提供する権限を示す、許可トークンのリストです。

SourceKeyId

(オプション) データの復号化に使用されるキーのキー ID です。

SourceEncryptionContext

(オプション) CiphertextBlob パラメータで指定されたデータの暗号化と復号化に使用される、暗号化コンテキストです。

このプロセスでは、前の説明の復号化および暗号化オペレーションを組み合わせます。顧客暗号文は、顧客暗号文で参照される最初の HBK で、意図した KMS キーで管理されている最新の HBK に復号化されます。このコマンドで使用されている KMS キーが同じである場合、コマンドは旧バージョンから最新バージョンの HBK に顧客暗号文を移動します。

次に示すのは、応答の構文です。

```
{
  "CiphertextBlob": blob,
  "DestinationEncryptionAlgorithm": "string",
  "KeyId": "string",
  "SourceEncryptionAlgorithm": "string",
  "SourceKeyId": "string"
}
```

呼び出し元のアプリケーションが基になるプレーンテキストの信頼性を確認したい場合は、SourceKeyId 返される が想定どおりであることを確認する必要があります。

AWS KMS 内部オペレーション

AWS KMS 内部は、グローバルに分散したキー管理サービスの HSM を拡張および保護するために必要です。

トピック

- [ドメインとドメインの状態](#)
- [内部通信セキュリティ](#)
- [マルチリージョンキーのレプリケーションプロセス](#)
- [耐久性の保護](#)

ドメインとドメインの状態

AWS リージョン 内の信頼済み内部 AWS KMS エンティティの協力的なコレクションをドメインと呼びます。ドメインには、信頼済みエンティティのセット、ルールのセット、およびドメインキーと呼ばれる秘密キーのセットが含まれます。ドメインキーは、ドメインのメンバーである HSM 間で共有されます。ドメイン状態は次のフィールドで構成されます。

名前

このドメインを識別するドメイン名。

メンバー

ドメインのメンバーである HSM のリスト。パブリック署名キーとパブリック共有キーを含む。

演算子

エンティティ、パブリック署名キー、およびこのサービスのオペレーターを表すロール (AWS KMS オペレーターまたはサービスホスト) のリスト。

ルール

HSM でコマンドを実行するために満たす必要がある各コマンドの定足数ルールのリスト。

ドメインキー

ドメイン内で現在使用されているドメインキー (対称キー) のリスト。

完全なドメイン状態は、HSM 上でのみ取得できます。ドメイン状態は、エクスポートされたドメイントークンとして HSM ドメインメンバー間で同期されます。

ドメインキー

ドメイン内のすべてのHSMは、ドメインキーのセット $\{DK_r\}$ を共有します。これらのキーは、ドメイン状態のエクスポートルーチンで共有されます。エクスポートされたドメイン状態は、ドメインのメンバーになっている任意の HSM にインポートできます。

ドメインキーのセット $\{DK_r\}$ には、常に 1 つのアクティブなドメインキーと、いくつかの非アクティブなドメインキーが含まれます。AWS が必ず「[Recommendation for Key Management - Part 1](#)」に準拠するように、ドメインキーは毎日ローテーションされます。ドメインキーのローテーション中に、送信ドメインキーの下で暗号化された既存の KMS キーはすべて、新しいアクティブなドメインキーの下で再暗号化されます。アクティブなドメインキーが、新しい EKT の暗号化には使用されます。期限切れのドメインキーは、最近ローテーションされたドメインキーの数と同等の日数の間、以前に暗号化された EKT を復号化するためにのみ使用できます。

エクスポートされたドメイントークン

ドメイン参加者間で定期的に状態を同期させる必要があります。これは、ドメインに変更が加えられるたびにドメインの状態をエクスポートすることによって実現されます。ドメインの状態は、エクスポートされたドメイントークンとしてエクスポートされます。

名前

このドメインを識別するドメイン名。

メンバー

ドメインのメンバーである HSM のリスト。署名パブリックキーおよび共有パブリックキーを含む。

演算子

エンティティ、パブリック署名キー、およびこのサービスのオペレーターを表すロールのリスト。

ルール

HSM ドメインメンバーでコマンドを実行するために満たす必要がある各コマンドの定足数ルールのリスト。

暗号化されたドメインキー

エンベロープで暗号化されたドメインキー。ドメインキーは、上記の各メンバーの署名メンバーによって暗号化され、パブリック共有キーにエンベロープされます。

署名

HSM によって生成されるドメイン状態の署名。ドメイン状態をエクスポートしたドメインのメンバーである必要があります。

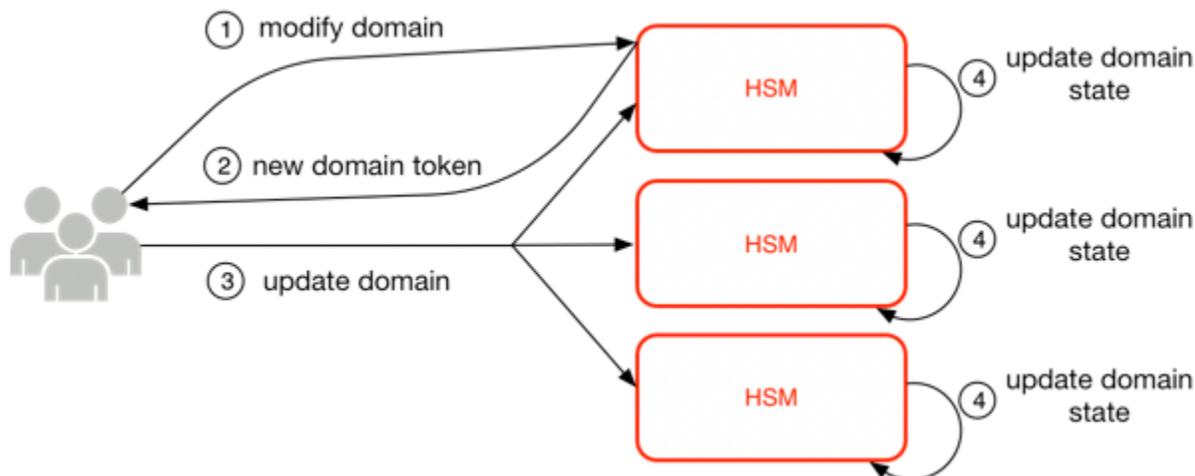
エクスポートされたドメイントークンは、ドメイン内で動作するエンティティの信頼の基盤を形成します。

ドメイン状態の管理

ドメインの状態は、定足数認証コマンドを使用して管理されます。これらの変更には、ドメイン内の信頼済み参加者のリストの変更、HSM コマンドを実行するための定足数ルールの変更、ドメインキーの定期的なローテーションが含まれます。これらのコマンドは、次の図に示すように、認証されたセッションオペレーションではなく、コマンド単位で認証されます。

初期化され動作状態にある HSM は、自己生成の非対称 ID キー、署名キーペア、およびキー確立キーペアのセットを含んでいます。手動プロセスで、AWS KMS オペレーターは、リージョンに最初に HSM が作成されたときに初期ドメインを確立できます。この初期ドメインは、このトピックで以前に定義した完全なドメイン状態で構成されます。ドメイン内の定義済みの HSM メンバーごとに、join コマンドを使用してインストールされます。

HSM が初期ドメインに参加すると、そのドメインで定義されているルールにバインドされます。これらのルールは、お客様の暗号化キーを使用するコマンドや、ホストまたはドメインの状態を変更するコマンドを管理します。暗号化キーを使用する認証済みセッション API オペレーションは、以前に定義されています。



上の図は、ドメインの状態がどのように変更されるかを示しています。このプロセスは次の 4 つのステップで構成されています。

1. ドメインを変更するために、定足数ベースのコマンドが HSM に送信します。
2. 新しいドメイン状態が生成され、新しいエクスポートされたドメイントークンとしてエクスポートされます。HSM 上の状態は変更されません。つまり、HSM 上で変更が行われません。
3. 2 番目のコマンドが新しくエクスポートされたドメイントークンの各 HSM に送信され、ドメインの状態を新しいドメイントークンで更新します。
4. エクスポートされた新しいドメイントークンにリストされている HSM は、コマンドとドメイントークンを認証できます。また、ドメイン内のすべての HSM のドメイン状態を更新するために、ドメインキーを解凍することもできます。

HSM は互いに直接通信しません。代わりに、定足数のオペレーターがドメインの状態の変更を要求し、結果として新しいエクスポートされたドメイントークンを生成します。ドメインのサービスホストメンバーが、ドメイン内のすべての HSM に新しいドメイン状態を配布するために使用されます。

ドメインの脱退と参加は、HSM 管理機能によって行われます。ドメイン状態の変更は、ドメイン管理機能によって行われます。

ドメインからの脱退

HSM はドメインから脱退し、すべての残留物とそのドメインのキーがメモリから削除されます。

ドメインへの参加

HSM は新しいドメインに参加するか、現在のドメイン状態を新しいドメイン状態に更新します。既存のドメインは、このメッセージを認証するためのルールの初期セットのソースとして使用されます。

ドメインの作成

HSM 上に新しいドメインを作成します。ドメインのメンバー HSM に配布できる最初のドメイントークンを返します。

オペレーターの変更

許可されたオペレーターとドメイン内のオペレーターのロールのリストにオペレーターを追加または削除します。

メンバーの変更

ドメイン内の許可された HSM のリストに HSM を追加または削除します。

ルールの変更

HSM でコマンドを実行するために必要な定足数ルールのセットを変更します。

ドメインキーのローテーション

新しいドメインキーが作成され、アクティブドメインキーとしてマークされます。これにより、既存のアクティブキーが非アクティブ化されたキーになり、最も古い非アクティブ化されたキーがドメイン状態から削除されます。

内部通信セキュリティ

サービスホスト間または AWS KMS オペレーターと HSM の間のコマンドは、[認証済みセッション](#)に示されている定足数署名付きリクエストメソッドと HSM サービスホストプロトコルを使用した認証セッションという 2 つのメカニズムによって保護されます。

定足数署名付きコマンドは、HSM が提供する重要なセキュリティ保護を 1 人のオペレーターが改変できないように設計されています。認証されたセッションで実行されるコマンドは、許可されたサービスオペレーターだけが KMS キーに関連する操作を実行できるようにするのに役立ちます。お客様にバインドされたすべての秘密情報は、AWS インフラストラクチャ全体で保護されます。

キー確立

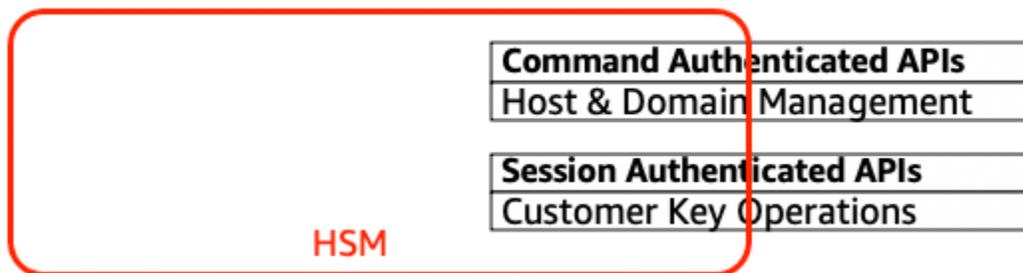
内部通信を保護するために、AWS KMS では、2 つの異なるキー確立方法を使用します。1 つ目は、「[Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography \(Revision 2\)](#)」で、 $C(1, 2, \text{ECC DH})$ と定義されています。このスキームには、静的な署名キーを持つイニシエータがあります。イニシエータは、静的 ECDH 共有キーを持つ受信者を対象として、一時的な楕円曲線 Diffie-Hellman (ECDH) キーを生成し、署名します。このメソッドは、ECDH を使用した 1 つの一時的キーと 2 つの静的キーを使用します。これは、ラベル $C(1, 2, \text{ECC DH})$ の導出です。この方法は、ワンパス ECDH と呼ばれることもあります。

2 つ目のキー確立方法は、[\(2, 2, ECC, DH\)](#) です。この方式では、両当事者は静的な署名キーを持ち、一時的な ECDH キーを生成、署名、交換します。この方法では、ECDH をそれぞれ使用した 2 つの静的キーと 2 つの一時的キーを使用します。これは、ラベル $C(2, 2, \text{ECC, DH})$ の導出です。この方法は、ECDH Ephemeral または ECDHE と呼ばれることもあります。すべての ECDH キーは、曲線 `secp384r1` (NIST-P384) 上で生成されます。

HSM セキュリティ境界

AWS KMS の内部セキュリティ境界は HSM です。HSM には独自のインターフェイスがあり、動作状態では他のアクティブな物理インターフェイスはありません。動作中の HSM は、ドメイン内でのロールを確立するために必要な暗号化キーを使用して初期化中にプロビジョニングされます。HSM の機密暗号化マテリアルは、揮発性メモリにのみ保存され、意図的であるかないかにかかわらず、シャットダウンやリセットなど、動作状態でなくなると消去されます。

HSM API オペレーションは、個々のコマンドによって、またはサービスホストによって確立された相互に認証された機密セッションによって認証されます。



定足数署名付きコマンド

定足数署名付きコマンドは、オペレーターによって HSM に対して発行されます。このセクションでは、定足数ベースのコマンドを作成、署名、および認証する方法について説明します。これらのルールはかなり簡単です。例えば、コマンド Foo には、ロール Bar からの 2 人のメンバーが必要といったものです。定足数ベースのコマンドの作成と検証には、3 つのステップがあります。最初のステップは元になるコマンドの作成で、2 番目のステップは署名する追加のオペレーターへの送信、3 番目のステップは検証と実行です。

概念を説明するために、次のように仮定します。オペレーターのパブリックキーとロールの信頼できるセット $\{QOS_s\}$ があり、一連の定足数ルール $QR = \{Command_i, Rule_{\{i, t\}}\}$ があるとします。ここで、各ルールはロールと最小数 $N \{Role_t, N_t\}$ のセットです。コマンドが定足数ルールを満たすためには、 $\{QOS_s\}$ にリストされている一連のオペレーターが、そのコマンド用にリストされているルールのいずれかを満たすように、コマンドデータセットに署名する必要があります。前述のように、定足数ルールとオペレーターのセットは、ドメイン状態とエクスポートされたドメイントークンに保存されます。

実際には、最初の署名者はコマンド $Sig_1 = \text{Sign}(dO_{p1}, \text{Command})$ に署名します。2 番目のオペレーターもコマンド $Sig_2 = \text{Sign}(dO_{p2}, \text{Command})$ に署名します。2 者に署名されたメッセージは、実行のために HSM に送信されます。HSM は、次の内容を実行します。

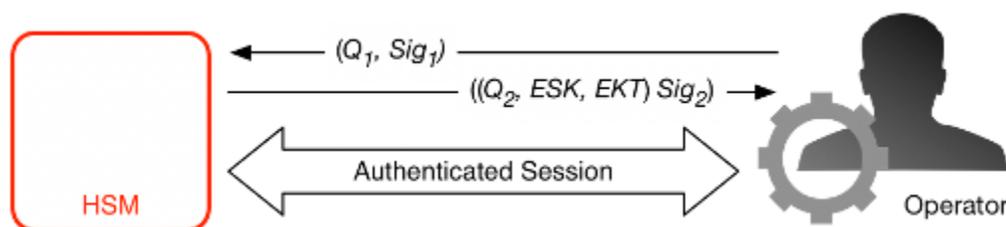
1. 署名ごとに、ドメイン状態から署名者のパブリックキーを抽出し、コマンドの署名を検証します。
2. 署名者のセットがコマンドのルールを満たしていることを確認します。

認証済みセッション

キーのオペレーションは、外部に向けられた AWS KMS ホストと HSM の間で実行されます。これらのコマンドは、暗号キーの作成と使用、および安全な乱数の生成に関係します。コマンドは、サービスホストと HSM の間のセッション認証チャンネル上で実行されます。これらのセッションには、真正性が必要なだけでなく、機密性も必要です。これらのセッションで実行されるコマンドは、発行者に向けてクリアテキストのデータキーや復号化されたメッセージを返送することがあります。これらのセッションを man-in-the-middle 攻撃によって覆すことがないように、セッションは認証されます。

このプロトコルは、HSM とサービスホスト間で相互に認証された ECDHE キー共有を実行します。交換はサービスホストが開始し、HSM が完了させます。HSM は、ネゴシエートされたキーによって暗号化されたセッションキー (SK) と、セッションキーを含むエクスポートされたキートークンも返します。エクスポートされたキートークンには有効期間が含まれています。有効期間が終了すると、サービスホストはセッションキーを再ネゴシエートする必要があります。

サービスホストはドメインのメンバーであり、ID 署名キーペア ($dHOS_i, QHOS_i$) と HSM の ID パブリックキーの信頼できるコピーを持っています。サービスホストは、ID 署名キーのセットを使用して、サービスホストとドメイン内の任意の HSM との間で使用できるセッションキーを安全にネゴシエートします。エクスポートされたキートークンには有効期間が関連付けられています。有効期間が終了すると、新しいキーをネゴシエートする必要があります。



このプロセスは、サービスホストが、自身とドメインの HSM メンバー間で機密通信フローを送受信するにはセッションキーが必要であることを認識することから始まります。

1. サービスホストが ECDH 一時的キーペア (d_1, Q_1) を生成し、これに ID キー $Sig_1 = \text{Sign}(dOS, Q_1)$ で署名します。

2. HSM は、受信したパブリックキーの署名を現在のドメイントークンを使用して検証し、ECDH 一時的キーペア (d_2, Q_2) を作成します。その後、「[Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography \(Revised\)](#)」に従って ECDH キー交換を完了させ、ネゴシエートされた 256 ビット AES-GCM キーを形成します。HSM は、新しい 256 ビット AES-GCM セッションキーを生成します。ネゴシエートされたキーを使用してセッションキーを暗号化し、暗号化されたセッションキー (ESK) を形成します。また、ドメインキーの下にあるセッションキーをエクスポートされたキートークン EKT として暗号化します。最後に、ID キーペア $Sig_2 = \text{Sign}(d\text{HSK}, (Q_2, \text{ESK}, \text{EKT}))$ で戻り値に署名します。
3. サービスホストは、現在のドメイントークンを使用して、受信したキーの署名を検証します。次に、サービスホストは「[Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography \(Revised\)](#)」に従って、ECDH キー交換を完了させます。次に ESK を復号化し、セッションキー SK を取得します。

EKT の有効期間中は、サービスホストはネゴシエートされたセッションキー SK を使用して、エンベロープで暗号化されたコマンドを HSM に送信できます。この認証済みセッションのすべての service-host-initiated コマンドには、EKT が含まれます。HSM は、ネゴシエートされた同じセッションキー SK を使用して応答します。

マルチリージョンキーのレプリケーションプロセス

AWS KMS は、クロスリージョンレプリケーションメカニズムを使用して、KMS キー内のキーマテリアルを、ある AWS リージョン内の HSM から別な AWS リージョン内の HSM にコピーします。このメカニズムを機能させるには、レプリケートされる KMS キーがマルチリージョンキーである必要があります。KMS キーをあるリージョンから別のリージョンにレプリケートする場合、リージョン内の HSM は分離されたネットワーク内にあるため、直接通信できません。代わりに、クロスリージョンレプリケーション中に交換されるメッセージは、プロキシサービスによって配信されます。

クロスリージョンレプリケーション中、AWS KMS HSM によって生成されたすべてのメッセージは、レプリケーション署名キーを使用して暗号で署名されます。レプリケーション署名キー (RSK) は、NIST P-384 曲線上の ECDSA キーです。すべてのリージョンが少なくとも 1 つの RSK を所有しており、各 RSK のパブリックコンポーネントは、同じ AWS パーティション内の他のすべてのリージョンと共有されます。

リージョン A からリージョン B にキーマテリアルをコピーするクロスリージョンレプリケーションプロセスは、次のように機能します。

1. リージョン B の HSM は、NIST P-384 曲線上に一時的な ECDH キーを生成します。Replication Agreement Key (レプリケーションアグリーメントキー) B (RAKB) です。RAKB のパブリックコンポーネントは、プロキシサービスによってリージョン A の HSM に送信されます。
2. リージョン A の HSM は RAKB のパブリックコンポーネントを受け取り、NIST P-384 曲線上に別の一時的な ECDH キーを生成します。Replication Agreement Key (レプリケーションアグリーメントキー) A (RAKA) です。HSM は、RAKA と RAKB のパブリックコンポーネント上で ECDH キー確立スキームを実行し、出力から対称キーを導出します。Replication Wrapping Key (レプリケーションラッピングキー) (RWK) です。RWK は、レプリケートされるマルチリージョン KMS キーのキーマテリアルを暗号化するために使用されます。
3. RAKA のパブリックコンポーネントと RWK で暗号化されたキーマテリアルは、プロキシサービスを通じてリージョン B の HSM に送信されます。
4. リージョン B の HSM は、RAKA のパブリックコンポーネントと RWK を使用して暗号化されたキーマテリアルを受け取ります。HSM は、RAKB と RAKA のパブリックコンポーネント上で ECDH キー確立スキームを実行し RWK で派生します。
5. リージョン B の HSM は、RWK を使用してリージョン A のキーマテリアルを復号化します。

耐久性の保護

オフライン HSM の使用、エクスポートされたドメイントークンの複数の不揮発性ストレージ、および暗号化された KMS キーの冗長ストレージにより、サービスによって生成されたキーに対するサービスの耐久性が強化されます。オフライン HSM は、既存のドメインのメンバーです。オンラインではなく、通常のドメイン操作に参加している場合を除き、オフライン HSM は既存の HSM メンバーと同じドメイン状態で表示されます。

AWS でオンライン HSM またはプライマリストレージシステムに格納されている一連の KMS キーの損失が広範囲に発生する懸念を考慮して、耐久性の設計は、リージョン内のすべての KMS キーを保護することを目的としています。インポートされたキーマテリアルを含む AWS KMS keys は、他の KMS キーの耐久性保護の下には含まれていません。AWS KMS でリージョン全体の障害が発生した場合は、インポートされたキーマテリアルを KMS キーに再インポートする必要がある場合があります。

オフラインの HSM と、それらにアクセスするための認証情報は、複数の独立した地理的場所にある監視された安全室内の金庫に保存されます。各金庫からこれらのマテリアルを取得するには、AWS の 2 つの独立したチームから、少なくとも 1 人の AWS セキュリティオフィサーと 1 人の AWS KMS オペレーターが必要です。これらのマテリアルの使用には、内部ポリシーで定める定足数の AWS KMS オペレーターの立ち合いが必要です。

リファレンス

このドキュメントで引用されている略語、キー、寄稿者、および出典の情報を得るには、次の参考資料を使用してください。

トピック

- [省略形](#)
- [キー](#)
- [寄稿者](#)
- [Bibliography](#)

省略形

次のリストは、このドキュメントで参照されている略語を示しています。

AES

Advanced Encryption Standard

CDK

お客様データキー

DK

ドメインキー

ECDH

Elliptic Curve Diffie-Hellman

ECDHE

Elliptic Curve Diffie-Hellman Ephemeral

ECDSA

Elliptic Curve Digital Signature Algorithm

EKT

エクスポートされたキートークン

ESK

暗号化されたセッションキー

GCM

Galois Counter Mode

HBK

HSM バックアップキー

HBKID

HSM バックアップキー識別子

HSM

ハードウェアセキュリティモジュール

RSA

Rivest Shamir and Adleman (暗号)

secp384r1

Standards for Efficient Cryptography 素数 384 ビットランダム曲線 1

SHA256

ダイジェスト長 256 ビットの Secure Hash Algorithm

キー

次のリストは、このドキュメントで参照されているキーの定義を示しています。

HBK

HSM バックアップキー: HSM バックアップキーは 256 ビットのルートキーで、そこから特定の用途のキーが派生します。

DK

ドメインキー: ドメインキーは、256 ビット AES-GCM キーです。これは、ドメインのすべてのメンバー間で共有され、HSM バックアップキーマテリアルと HSM サービスホストセッションキーを保護するために使用されます。

DKEK

ドメインキー暗号化キー: ドメインキー暗号化キーは、ホスト上で生成された AES-256-GCM キーで、HSM ホスト間でドメイン状態を同期するドメインキーの現在のセットを暗号化するために使用されます。

(dHAK,QHAK)

HSM 共有キーペア: 開始された HSM はすべて、secp384r1 (NIST-P384) 曲線上にローカルに生成された楕円曲線 Diffie-Hellman 共有キーペアを持ちます。

(dE, QE)

一時的共有キーペア: HSM とサービスホストは、一時的共有キーを生成します。これらは、secp384r1 (NIST-P384) 曲線上の楕円曲線 Diffie-Hellman キーです。これらは 2 つのユースケースで生成されます。ドメイントークンでドメインキー host-to-host 暗号化キーを転送するための暗号化キーを確立する場合と、機密性の高い通信を保護するための HSM サービスホストセッションキーを確立する場合です。

(dHSK,QHSK)

HSM 署名キーペア: 開始された HSM はすべて、secp384r1 (NIST-P384) 曲線上にローカルに生成された楕円曲線デジタル署名キーペアを持ちます。

(dOS,QOS)

オペレーター署名キーペア: サービスホストオペレーターと AWS KMS オペレーターには、他のドメイン参加者に対して自身を認証するために使用する ID 署名キーがあります。

K

データ暗号化キー: SHA256 で HMAC を使用したカウンタモードで NIST SP800-108 KDF を使用して HBK から派生した 256 ビットの AES-GCM キー。

SK

セッションキー: セッションキーは、認証済み楕円曲線 Diffie-Hellman キーがサービスホストオペレーターと HSM の間で交換された結果として作成されます。交換の目的は、サービスホストとドメインのメンバー間の通信を保護することです。

寄稿者

このドキュメントには、次の個人および組織が貢献しました。

- Ken Beer、ゼネラルマネージャー - KMS、AWS 暗号化
- Matthew Campagna、プリンシパルセキュリティエンジニア、AWS 暗号化

Bibliography

AWS Key Management Service HSM については、NIST Computer Security Resource Center の [Cryptographic Module Validation Program 検索ページ](#) で「AWS Key Management Service HSM」を検索してください。

Amazon Web Services、全般のリファレンス (バージョン 1.0)、 「AWS API リクエストの署名」、 http://docs.aws.amazon.com/general/latest/gr/signing_aws_api_requests.html。

Amazon Web Services、「What is the AWS Encryption SDK」、 <http://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/introduction.html>。

Federal Information Processing Standards Publications, FIPS PUB 180-4. Secure Hash Standard、2012 年 8 月 <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf> から入手可能。

Federal Information Processing Standards Publication 197、Announcing the Advanced Encryption Standard (AES)、2001 年 11 月。 <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf> から入手可能。

Federal Information Processing Standards Publication 198-1、The Keyed-Hash Message Authentication Code (HMAC)、2008 年 7 月。 http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf から入手可能。

NIST Special Publication 800-52 Revision 2、2019 年 8 月の Transport Layer Security (TLS) 実装の選択、設定、使用に関するガイドライン。 <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf>。

PKCS#1 v2.2: RSA Cryptography Standard (RFC 8017)、Internet Engineering Task Force (IETF)、2016 年 11 月。 <https://tools.ietf.org/html/rfc8017>。

Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC、NIST Special Publication 800-38D、2007 年 11 月。 <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf> から入手可能。

Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices、NIST Special Publication 800-38E、2010 年 1 月。 <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf> から入手可能。

Recommendation for Key Derivation Using Pseudorandom Functions、NIST Special Publication 800-108、2009 年 10 月、<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-108.pdf> から入手可能。

Recommendation for Key Management - Part 1: General (Revision 5)、NIST Special Publication 800-57A、2020 年 5 月、<https://doi.org/10.6028/NIST.SP.800-57pt1r5> から入手可能。

Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised)、NIST Special Publication 800-56A Revision 3、2018 年 4 月。<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf> から入手できます。

決定論的ランダムビットジェネレーターを使用した乱数生成のレコメンデーション、NIST Special Publication 800-90A Revision 1、2015 年 6 月 1、<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf> から入手可能。

SEC 2: Recommended Elliptic Curve Domain Parameters、Standards for Efficient Cryptography Group、Version 2.0、2010 年 1 月 27 日。

Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)、Brown, D.、Turner, S.、Internet Engineering Task Force、2010 年 7 月、<http://tools.ietf.org/html/rfc5753/>。

X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)、American National Standards Institute、2005。

AWS KMS 暗号化の詳細のドキュメント履歴

以下の表は AWS Key Management Service 暗号化の詳細ドキュメントの重要な変更点をまとめたものです。また、お客様からいただいたフィードバックに対応するために、ドキュメントを頻繁に更新しています。

変更	説明	日付
更新された内容	AWS KMS ReplicateKey オペレーションの実装に関する詳細を追加しました。	2021 年 10 月 28 日
ドキュメントの変更	カスタマーマスターキー (CMK) という条件が AWS KMS key および KMS キーに置き換えられています。	2021 年 8 月 30 日
初回リリース	このガイドは「KMS 暗号化の詳細」技術文書を基に作成	2020 年 12 月 30 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。