aws

ユーザーガイド

AWS Payment Cryptography

AWS Payment Cryptography: ユーザーガイド

Table of Contents

AWS Payment Cryptography とは？

AWS Payment Cryptography は、専用の決済 HSM インスタンスを調達しなくても、ペイメントカード業界 (PCI) 標準に従って支払い処理に使用される暗号化機能とキー管理にアクセスできるマネージド AWS サービスです。AWSPayment Cryptographyを使用すると、アクワイアラー、ペイメントファシリテーター、ネットワーク、スイッチ、プロセッサー、銀行などの決済機能を実行する顧客は、Payment Cryptography業務をクラウド内のアプリケーションに近づけ、専用の決済HSMを含む補助データセンターやコロケーション施設への依存を最小限に抑えることができます。

このサービスは、PCI PIN、PCI P2PE、PCI DSSなどの該当する業界ルールを満たすように設計されており、PCI PTS HSM V3およびFIPS 140-2レベル3認定を受けているハードウェアを活用しています。低レイテンシー、高レベルのアップタイム、耐障害性をサポートするように設計されています。AWSPayment Cryptography は柔軟性が高く、ハードウェアのプロビジョニング、キーマテリアルの安全な管理、安全な施設での緊急バックアップの維持など、オンプレミスHSMの運用要件の多くを排除します。AWSPayment Cryptography では、キーをパートナーと電子的に共有するオプションも用意されているため、ペーパークリアテキストコンポーネントを共有する必要がありません。

AWS Payment Cryptography コントロールプレーン API を使用してキーを作成および管理できます。

AWS Payment Cryptography データプレーン API を使用すると、支払い関連のトランザクション処理や関連する暗号化オペレーションに暗号化キーを使用できます。

AWS Payment Cryptography には、キーの管理に使用できる重要な機能があります。

- TDES、AES、RSA キーなどの対称型および非対称型の AWS Payment Cryptography キーを作成および管理し、CVV 生成や DUKPT キーの導出などの用途を指定します。
- AWS Payment Cryptography キーは、ハードウェアセキュリティモジュール (HSM) で保護された状態で自動的に安全に保存され、ユースケースごとにキーが分離されます。
- エイリアスを作成、削除、一覧表示、および更新します。エイリアスは、「フレンドリ名」であり、これを使用して、AWS Payment Cryptography キーへのアクセスまたはアクセスを制御できます。
- 識別、グループ化、オートメーション、アクセス制御、コスト追跡のために、AWS Payment Cryptography キーにタグ付けします。
- TR-31 (連携運用可能なセキュアキー交換キーブロック仕様) に従い、キー暗号化キー (KEK) を使用して、AWS Payment Cryptography と HSM (またはサードパーティ) 間の対称キーをインポートおよびエクスポートします。

- AWS非対称キーペアを使用する Payment Cryptography システムと他のシステムとの間で対称キー暗号キー (KEK) をインポートおよびエクスポートします。続いて、TR-34 (非対称技術を用いた対称キーの配布方法) などの電子的手段を使用します。

AWS Payment Cryptography キーは、次のような暗号化オペレーションに使用できます。

- 対称または非対称 AWS Payment Cryptography キーを使用して、データを暗号化、復号、再暗号化します。
- PCI PIN ルールに従い、クリアテキストを公開することなく、機密データ (カード所有者 PIN など) を暗号化キー間で安全に変換できます。
- CVV、CVV2、ARQC などのカード会員データを生成または検証します。
- カード会員ピンを生成して検証します。
- MAC 署名を生成または検証します。

概念

AWS Payment Cryptography で使用される基本的な用語と概念、およびそれらを使用してデータを保護する方法について説明します。

エイリアス

AWS Payment Cryptography キーに関連付けられているわかりやすい名前。エイリアスは、多くの AWS Payment Cryptography API オペレーションでキー ARN と互換的に使用できます。エイリアスを使用すると、アプリケーションコードに影響を与えることなく、キーをローテーションしたり変更したりできます。エイリアス名は、最大 256 文字の文字列です。アカウントとリージョン内の関連付けられた AWS Payment Cryptography キーを一意に識別します。 AWS Payment Cryptography では、エイリアス名は常に で始まりますalias/。

エイリアス名の形式は次のとおりです。

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

例:

```
alias/sampleAlias2
```

キー ARN

キー ARN は、 AWS Payment Cryptography のキーエントリの Amazon リソースネーム (ARN) です。これは Payment Cryptography AWS キーの一意の完全修飾識別子です。キー ARN には AWS アカウント、、リージョン、ランダムに生成された ID が含まれます。ARN はキーマテリアルとは関係がなく、キーマテリアルから派生したものでもありません。これらの値は作成またはインポートオペレーション中に自動的に割り当てられるため、これらの値は同等ではありません。同じキーを複数回インポートすると、独自のライフサイクルを持つ複数のキー ARN が生成されます。

キー ARN の形式は次のとおりです。

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

次に、 キー ARNの例を示します。

```
arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h
```

キー識別子

キー識別子はキーへの参照であり、そのうちの 1 つ (または複数) は AWS Payment Cryptography オペレーションへの一般的な入力です。有効なキー識別子は、キーエイリアス のキー ARN のいずれかです。

AWS Payment Cryptography キー

AWS Payment Cryptography キー (キー) は、すべての暗号化関数に使用されます。キーは [create key] コマンドを使用して直接生成するか、キーインポートを呼び出してシステムに追加します。キーのオリジンは、属性 を確認することで決定できます KeyOrigin。 AWS Payment Cryptography は、DUKPT で使用されるキーなど、暗号化オペレーション中に使用される派生キーまたは中間キーもサポートします。

これらのキーには、作成時に定義される不変属性と可変属性の両方があります。アルゴリズム、長さ、使用法などの属性は作成時に定義され、変更することはできません。発効日や有効期限など、その他のものは変更できます。AWS Payment Cryptography キー属性の完全なリストについては、「 Payment Cryptography API リファレンス」を参照してください。 AWS

AWS Payment Cryptography キーにはキータイプがあり、主に ANSI X9 TR 31 で定義され、PCI PIN v3.1 要件 19 で指定されている使用目的に制限されます。

属性は、PCI PIN v3.1 要件 18-3 の規定に従って保存、他のアカウントとの共有、またはエクスポート時に、キーブロックを使用してキーにバインドされます。

キーは、キー Amazon リソースネーム () と呼ばれる一意の値を使用して AWS Payment Cryptography プラットフォームで識別されますARN。

> **Note**
>
> キーARNは、キーが最初に作成または AWS Payment Cryptography サービスにインポートされたときに生成されます。そのため、キーのインポート機能を使用して同じキーマテリアルを複数回追加すると、同じキーマテリアルが複数のキーの下に置かれますが、それぞれのキーライフサイクルは異なります。

業界用語

トピック

一般的なキータイプ

AWK

アクワイアラーワーキングキー (AWK) は、アクワイアラー/アクワイアラープロセッサとネットワーク (VisaやMastercardなど) 間のデータ交換に通常使用されるキーです。歴史上、AWK は暗号化に 3DES を利用しており、これは「TR31_P0_PIN_ENCRYPTION_KEY」と表示されます。

BDK

基本派生キー (BDK) は、後続のキーを導出するために使用されるワーキングキーで、一般的に PCI PIN や PCI P2PE DUKPT プロセスの一部として使用されます。これは TR31_B0_BASE_DERIVATION_KEY と表記されます。

CMK

カードマスターキー (CMK) は、通常、発行者マスターキー、PAN、PSN から派生した 1 つ以上のカード固有のキーで、通常は 3DES キーです。これらのキーは、カスタマイズ時に EMV チップに保存されます。CMK の例としては、AC キー、SMI キー、SMC キーなどがあります。

CMK-AC

アプリケーション暗号文 (AC) キーは、EMV トランザクションの一部としてトランザクションクリプトグラムを生成するために使用され、カードマスターキーの一種です。

CMK-SMI

EMV の一部として、PIN 更新スクリプトなどの MAC を使用してカードに送信されるペイロードの整合性を検証するために、セキュアメッセージングインテグリティ (SMI) キーが使用されます。これはカードマスターキーの一種です。

CMK-SMC

EMV の一部として、暗証番号の更新など、カードに送信されるデータを暗号化するためにセキュアメッセージング機密保持 (SMC) キーが使用されます。これはカードマスターキーの一種です。

CVK

カード検証キー (CVK) は、定義されたアルゴリズムを使用して CVV や CVV2 などの値を生成したり、入力を検証したりするために使用されるキーです。これは TR31_C0_CARD_VERIFICATION_KEY と表記されます。

iCVV

iCVV は CVV2-likeた値ですが、EMV (チップ) カードに track2 に相当するデータが埋め込まれています。この値は 999 のサービスコードを使用して計算され、CVV1/CVV2 とは異なり、盗まれた情報が別のタイプの新しい支払い認証情報の作成に使用されるのを防ぎます。例えば、チップトランザクションデータを取得した場合、このデータを使用して磁気ストライプ (CVV1) を生成したり、オンライン購入 (CVV2) を行ったりすることはできません。

??? キーを使用する

IMK

発行者マスターキー (IMK) は EMV チップカードのパーソナライゼーションの一部として使用されるマスターキーです。通常、AC (暗号文)、SMI (完全性/署名用のスクリプトマスターキー)、および SMC (機密性/暗号化用のスクリプトマスターキー) の各キーに 1 つずつ、計 3 種類の IMK があります。

IK

初期キー (IK) は DUKPT プロセスで使用される最初のキーであり、基本導出キー (BDK) から取得されます。このキーではトランザクションは処理されませんが、トランザクションに使用される将来のキーを導出するために使用されます。IK を作成するための取得方法は、X9.24-1:2017 で

定義されています。TDES BDK を使用する場合、X9.24-1:2009 が適用可能な標準であり、IK は Initial Pin Encryption Key (IPEK) に置き換えられます。

IPEK

初期 PIN 暗号化キー (IPEK) は DUKPT プロセスで使用される初期キーで、ベース派生キー (BDK) から生成されます。このキーではトランザクションは処理されませんが、トランザクションに使用される将来のキーを導出するために使用されます。IPEK は、データ暗号化や mac キーの派生にも使用できるため、誤解があります。IPEK を作成するための取得方法は、X9.24-1:2009 で定義されています。AES BDK を使用する場合、X9.24-1:2017 が適用可能な標準であり、IPEK は初期キー (IK) に置き換えられます。

IWK

発行者ワーキングキー (IWK) は、発行者/発行者の処理者とネットワーク (Visa や Mastercard など) との間でデータを交換するために通常使用されるキーです。歴史上、IWK は暗号化に 3DES を利用しており、TR31_P0_PIN_ENCRYPTION_KEY と表示されます。

KEK

キー暗号化キー (KEK) は、送信時または保存時に他のキーを暗号化するために使用されるキーです。他のキーを保護するキーには通常、 KeyUsageTR-31標準に従って TR31_K0_KEY_ENCRYPTION_KEY の があります。

PEK

PIN 暗号化キー (PEK) は、2 者間での保存または送信を目的として PIN を暗号化するために使用される作業キーの一種です。IWK と AWK は PIN 暗号化キーの具体的な用途の 2 つの例です。これらのキーは TR31_P0_PIN_ENCRYPTION_KEY と表示されます。

PVK

PIN 検証キー (PVK) は PVV などの PIN 検証値を生成するために使用されるワーキングキーの一種です。IBM3624 オフセット値の生成に使用される TR31_V1_IBM3624_PIN_VERIFICATION_KEY と、VISA/ABA 検証値の生成に使用される TR31_V2_VISA_PIN_VERIFICATION_KEY が最も一般的な 2 つの種類です。

その他の用語

ARQC

オーソライゼーションリクエスト暗号文 (ARQC) は、EMV 標準チップカード (または同等の非接触型実装) によってトランザクション時に生成される暗号です。通常、ARQC はチップカードによって生成され、発行者またはその代理人に転送されて取引時に検証されます。

DUKPT

トランザクションごとの派生一意キー (DUKPT) は、物理的な POS/POI で一度だけ使用できる暗号キーの使用を定義するために一般的に使用されるキー管理標準です。歴史上、DUKPT は暗号化に 3DES を利用しています。DUKPT の業界標準は ANSI X9.24-3-2017 で定義されています。

EMV

EMV (当初はユーロペイ、Mastercard、Visa) は、支払いの利害関係者と協力して相互運用可能な支払い基準とテクノロジーを作成する技術団体です。標準例の 1 つは、チップ/コンタクトレスカードと、使用する暗号化など、それらが操作する支払いターミナルです。EMV キー取得とは、などのキーの初期セットに基づいて各支払いカードに一意のキーを生成する方法 (複数可) を指します。 IMK

HSM

ハードウェアセキュリティモジュール (HSM) は、暗号化オペレーション (暗号化、復号化、デジタル署名など) と、これらのオペレーションに使用される基盤となるキーを保護する物理デバイスです。

KCV

キーチェックバリュー (KCV) とは、実際のキーデータにアクセスせずにキー同士を比較するために主に使用されるさまざまなチェックサムメソッドを指します。KCV は整合性の検証 (特にキーの交換時) にも使用されてきましたが、現在ではこの役割は TR-31 などのキーブロック形式の一部として組み込まれています。TDES キーの場合、KCV は 8 バイト (各バイトの値が 0) を暗号化してキーをチェックし、暗号化された結果の上位 3 バイトを保持することで計算されます。AES キーの場合、KCV は CMAC アルゴリズムを使用して計算されます。このアルゴリズムでは、入力データは 16バイトで、暗号化された結果の上位 3 バイトは保持されます。

KDH

キー分散ホスト (KDH) は TR-34 などのキー交換プロセスでキーを送信するデバイスまたはシステムです。 AWS Payment Cryptography からキーを送信する場合、KDH と見なされます。

KIF

キーインジェクションファシリティ (KIF) は、決済ターミナルの初期化 (暗号化キーのロードなど) に使用される安全な機能です。

KRD

キー受信デバイス (KRD) は TR-34 などのキー交換プロセスでキーを受信するデバイスです。AWS Payment Cryptography にキーを送信する場合、KRD と見なされます。

KSN

キーシリアル番号 (KSN) は、DUKPT 暗号化/復号化の入力として使用される値で、トランザクションごとに一意の暗号化キーを作成します。KSN は通常、BDK 識別子、半一意のターミナル ID、および特定の決済ターミナルで処理されるたびに増加するトランザクションカウンターで構成されます。

PAN

プライマリアカウント番号 (PAN) は、クレジットカードやデビットカードなどの口座固有の識別子です。通常、長さは 13 ~ 19 桁です。最初の 6 ~ 8 桁はネットワークと発行銀行を識別します。

PIN ブロック

処理中または送信中の PIN とその他のデータ要素を含むデータブロック。PIN ブロック形式は PIN ブロックの内容と、PIN を取得するための処理方法を標準化します。ほとんどの PIN ブロックは PIN、PIN の長さで構成され、PAN の一部またはすべてが頻繁に含まれます。 AWS Payment Cryptography は ISO 9564-1 形式 0、1、3、4 をサポートしています。AES キーにはフォーマット 4 が必要です。PIN を検証または変換する場合、受信データまたは送信データの PIN ブロックを指定する必要があります。

POI

ポイントオブインタラクション (POI) は、POS (販売時点管理) と同義語としてもよく使われるハードウェアデバイスで、カード所有者が支払い認証情報を提示する際に使用するハードウェアデバイスです。POI の例としては、マーチャントロケーションの物理ターミナルがあります。認定済み PCI PTS POI ターミナルのリストについては、PCI ウェブサイトを参照してください。

PSN

PAN シーケンス番号 (PSN) は、同じ PAN で発行された複数のカードを区別するために使用される数値です。

パブリックキー

非対称暗号 (RSA) 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パブリックコンポーネントです。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) 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 AWS Payment Cryptography キーと同様に、プライベートキーは HSMs によって安全に作成されます。これらは、暗号化リクエストの処理に必要な期間、HSM の揮発性メモリにのみ復号化されます。

PVV

PIN 検証値 (PVV) は、カード番号や PIN などの一連の入力からアルゴリズム的に導出された値で、この値を生成してその後の検証に使用できます。このような方式の 1 つは Visa PVV として知られています(ABA 方式としても知られています) が、どのネットワークの PIN にも使用されます。

RSA ラップ/アンラップ

RSA ラップは非対称キーを使用して、別のシステムに送信するための対称キー (TDES キーなど) をラップします。一致するプライベートキーを持つシステムのみがペイロードを復号し、対称キーをロードできます。逆に、RSA アンラップは、RSA を使用して暗号化されたキーを安全に復号し、そのキーを AWS Payment Cryptography にロードします。RSA ラップは、キー交換の低レベルの方法であり、キーブロック形式でキーを送信せず、送信側によるペイロード署名も使用しません。代替コントロールを検討して、提供量とキー属性が変更されていないことを確認する必要があります。

TR-34 は内部的にも RSA を使用しますが、別の形式であり、相互運用できません。

TR-31

TR-31 (正式には ANSI X9 TR 31 と定義されます) は、米国規格協会 (ANSI) によって定義されているキーブロック形式で、キーデータ自体と同じデータ構造でのキー属性の定義をサポートします。TR-31 キーブロック形式は、キーに関連付けられたキー属性のセットを定義してまとめて保持します。 AWS Payment Cryptography は、可能な限り TR-31 標準化用語を使用して、キーの適切な分離とキーの目的を確保します。TR-31 は ANSI X9.143-2022 に取って代わられました。

TR-34

TR-34 は ANSI X9.24-2 の実装であり、非対称手法 (RSA など) を使用して対称キー (3DES や AES など) を安全に配布するプロトコルについて説明しています。 AWS Payment Cryptography は TR-34 メソッドを使用して、キーの安全なインポートとエクスポートを許可します。

関連サービス

AWS Key Management Service

AWS Key Management Service (AWS KMS) は、ユーザーのデータを保護するために使用される、暗号化キーの作成と制御を容易にするマネージドサービスです。AWSKMS では、ハードウェアセキュリティモジュール (HSM) を使用して AWS KMS キーを保護および検証します。

AWS CloudHSM

AWS CloudHSM は、AWS クラウド内の専用の汎用 HSM インスタンスを顧客に提供します。AWS CloudHSM はキーの作成、データ署名、データの暗号化と復号化など、さまざまな暗号化機能を提供できます。

詳細情報

- Payment Cryptography で使用される用語と概念については、「AWSAWS Payment Cryptography の概念」を参照してください。
- Payment Cryptography コントロールプレーン API の詳細については、「AWSAWS Payment Cryptography コントロールプレーン API リファレンス」を参照してください。
- Payment Cryptography データプレーン API の詳細については、「AWSAWS Payment Cryptography データプレーン API リファレンス」を参照してください。
- AWS が暗号化を使用し、KMS キーを保護する方法の詳細な技術情報については、「AWS暗号化の詳細」を参照してください。

のエンドポイント AWS Payment Cryptography

プログラムで に接続するには AWS Payment Cryptography、エンドポイントを使用します。エンドポイントは、サービスのエントリポイントの URL です。 AWS SDKs とコマンドラインツールでは、リクエストのリージョンコンテキスト AWS リージョン に基づいて のサービス用のデフォルト

のエンドポイントが自動的に使用されるため、通常、これらの値を明示的に設定する必要はありません。必要に応じて、API リクエストに別のエンドポイントを指定できます。

コントロールプレーンエンドポイント

リージョン名	リージョン	エンドポイント	プロトコル
米国東部 (バージニア北部)	us-east-1	controlplane.payment-cryptography.us-east-1.amazonaws.com	HTTPS
米国東部 (オハイオ)	us-east-2	controlplane.payment-cryptography.us-east-2.amazonaws.com	HTTPS
米国西部 (オレゴン)	us-west-2	controlplane.payment-cryptography.us-west-2.amazonaws.com	HTTPS

データプレーンエンドポイント

リージョン名	リージョン	エンドポイント	プロトコル
米国東部 (バージニア北部)	us-east-1	dataplane.payment-cryptography.us-east-1.amazonaws.com	HTTPS
米国東部 (オハイオ)	us-east-2	dataplane.payment-cryptography.us-east-2.amazonaws.com	HTTPS
米国西部 (オレゴン)	us-west-2	dataplane.payment-cryptography.us-west-2.amazonaws.com	HTTPS

AWS Payment Cryptography の開始方法

AWS Payment Cryptography を始めるには、まずキーを作成して、それをさまざまな暗号化オペレーションに使用する必要があります。以下のチュートリアルでは、CVV2 値の生成/検証に使用するキーを生成する簡単な使用例を紹介しています。他の例を試したり、AWS 内のデプロイパターンを調べたりするには、以下の AWS Payment Cryptography ワークショップを試してみるか、Github で公開されているサンプルプロジェクトをご覧ください。

このチュートリアルでは、1 つのキーを作成し、そのキーを使用して暗号化オペレーションを実行する手順について説明します。その後、不要になったキーを削除すると、キーのライフサイクルは完了します。

トピック

前提条件

開始する前に、以下を確認してください。

- サービスにアクセスする許可を得ていること。詳細については、「IAM ポリシー」を参照してください。
- AWS CLI がインストールされていること。AWS SDK や AWS API を使用して AWS Payment Cryptography にアクセスすることもできますが、このチュートリアルの説明では AWS CLI を使用しています。

ステップ 1: キーを作成する

最初のステップは、キー を作成することです。このチュートリアルでは、CVV/CVV2 値を生成および検証するための CVK 倍長 3DES (2KEY TDES) キーを作成します。

```
$ aws payment-cryptography create-key \
    --exportable
    --key-attributes KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,\
        KeyClass=SYMMETRIC_KEY,\
        KeyModesOfUse='{Generate=true,Verify=true}'
```

レスポンスには、後続の呼び出し用の ARN やキーチェック値 (KCV) などのリクエストパラメータがエコーバックされます。

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
        "KeyAttributes": {
            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyAlgorithm": "TDES_2KEY",
            "KeyModesOfUse": {
                "Encrypt": false,
                "Decrypt": false,
                "Wrap": false,
                "Unwrap": false,
                "Generate": true,
                "Sign": false,
                "Verify": true,
                "DeriveKey": false,
                "NoRestrictions": false
            }
        },
        "KeyCheckValue": "CADDA1",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": true,
        "Exportable": true,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
        "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
    }
}
```

キーを表す KeyArn を書き留めてください（例えば、arn: aws: payment-crypto: us-east-2:111122223333:key/tqv5yij6wtxx64pi）。これは次のステップで行います。

ステップ 2: キーを使用して CVV2 値を生成する

このステップでは、手順 1 のキーを使用して、所定の PAN と有効期限を表す CVV2 を生成します。

```
$  aws payment-cryptography-data generate-card-validation-data \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi \
    --primary-account-number=171234567890123 \
    --generation-attributes CardVerificationValue2={CardExpiryDate=0123}
```

```
{
    "CardDataGenerationKeyCheckValue": "CADDA1",
    "CardDataGenerationKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/tqv5yij6wtxx64pi",
    "CardDataType": "CARD_VERIFICATION_VALUE_2",
    "CardDataValue": "144"
}
```

cardDataValue を書き留めておきます (この場合は 3 桁の数字 144) 。これは次のステップで行います。

ステップ 3: 手順 2 で生成された値を確認する

この例では、ステップ 1 で作成したキーを使用して、ステップ 2 の CVV2 を検証します。

以下のコマンドを実行して CVV2 を検証します。

```
$  aws payment-cryptography-data  verify-card-validation-data \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi \
    --primary-account-number=171234567890123 \
    --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
    --validation-data 144
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "CADDA1"
}
```

このサービスは、CVV2 を検証したことを示す 200 の HTTP 応答を返します。

ステップ 4: ネガティブテストを実行する

このステップでは、CVV2 が正しくなく、検証もされないネガティブテストを作成します。ステップ 1 で作成したキーを使用して、誤った CVV2 を検証しようとします。これは予想されるオペレーションであり、例えば、カード所有者がチェックアウト時に間違った CVV2 を入力した場合などです。

```
$  aws payment-cryptography-data  verify-card-validation-data \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi \
    --primary-account-number=171234567890123 \
    --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
    --validation-data 999
```

```
Card validation data verification failed.
```

このサービスは、「カード検証データの検証に失敗しました」というメッセージと「INVALID_VALIDATION_DATA」という理由を含む 400 の HTTP レスポンスを返します。

ステップ 5: (オプション) クリーンアップする

これで、手順 1 で作成したキーを削除できます。回復不可能な変更を最小限に抑えるため、デフォルトのキー削除期間は 7 日間です。

```
$ aws payment-cryptography delete-key \
    --key-identifier=arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi
```

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T08:27:51.795000-07:00",
        "DeletePendingTimestamp": "2022-11-03T13:37:12.114000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
```

```
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY"
        },
        "KeyCheckValue": "CADDA1",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "DELETE_PENDING",
        "UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
    }
}
```


出力の 2 つのフィールドを書き留めておきます。deletePendingTimestamp は、デフォルトで 7 日後に設定されています。キーステートは DELETE_PENDING に設定されています。この削除は、予定されている削除時刻より前であればいつでも restore-key を呼び出してキャンセルできます。

キーの管理

AWS Payment Cryptography の使用を開始するには、 AWS Payment Cryptography キーを作成します。

このセクションのトピックでは、作成から削除まで、さまざまな AWS Payment Cryptography キータイプを作成して管理する方法について説明します。キーの作成、編集、表示、キーのタグ付け、キーエイリアスの作成、キーの有効化と無効化方法がトピックに含まれます。

トピック

- キーの生成
- キーのリスト
- キーの有効化と無効化
- キーの削除
- キーのインポートとエクスポート
- エイリアスの使用
- キーを取得する
- キーのタグ付け
- AWS Payment Cryptography キーのキー属性について

キーの生成

AWS Payment Cryptography キーは、 CreateKey API オペレーションを使用して作成できます。このプロセスでは、キーアルゴリズム (TDES_3KEY など)、 (TR31_P0_PIN_ENCRYPTION_KEY KeyUsage など)、許可されたオペレーション (暗号化、署名など)、エクスポート可能かどうかなど、キーまたは結果の出力のさまざまな属性を指定します。 AWS Payment Cryptography キーの作成後にこれらのプロパティを変更することはできません。

例

- 2KEY TDES キーの生成
- PIN 暗号化キーの生成
- 非対称 (RSA) キーを作成する
- PIN 検証値 (PVV) キーの生成

2KEY TDES キーの生成

Example

このコマンドは、CVV/CVV2 値の生成と検証を目的として 2KEY TDES キーを生成します。レスポンスは、後続の呼び出し用の ARN や KCV (Key Check Value) を含むリクエストパラメータをエコーバックします。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,\
KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY, \
KeyModesOfUse='{Generate=true,Verify=true}'
```

```
                        {
    "Key": {
        "CreateTimestamp": "2022-10-26T16:04:11.642000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
hjprdg5o4jtgs5tw",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_2KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": false,
                "Encrypt": false,
                "Generate": true,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
        },
        "KeyCheckValue": "B72F",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
```

```
        "UsageStartTimestamp": "2022-10-26T16:04:11.559000-07:00"
    }
}
```

PIN 暗号化キーの生成

Example PIN 暗号化キー (PEK) の生成

このコマンドは、PIN 値を暗号化するための 3KEY TDES キー (PIN 暗号化キーと呼ばれます) を生成します。このキーは、PIN を安全に保管したり、トランザクション中などの検証時に提供された PIN を復号化したりするために使用できます。レスポンスは、後続の呼び出し用の ARN や KCV (Key Check Value) を含むリクエストパラメータをエコーバックします。

```
$ aws payment-cryptography create-key --exportable --key-attributes \
                        KeyAlgorithm=TDES_3KEY,KeyUsage=TR31_P0_PIN_ENCRYPTION_KEY, \
                        KeyClass=SYMMETRIC_KEY,/

 KeyModesOfUse='{Encrypt=true,Decrypt=true,Wrap=true,Unwrap=true}'
```

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T08:27:51.795000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
```

```
            },
            "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY"
        },
        "KeyCheckValue": "9CA6",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
    }
}
```

非対称 (RSA) キーを作成する

Example

この例では、新しい非対称 RSA 2048 ビットキーペアを生成します。新しい秘密キーと、対応する公開キーが生成されます。パブリックキーは、getPublicCertificate API を使用して取得できます。

```
$ aws payment-cryptography create-key --exportable \
--key-attributes
 KeyAlgorithm=RSA_2048,KeyUsage=TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION, \
KeyClass=ASYMMETRIC_KEY_PAIR,KeyModesOfUse='{Encrypt=true,
 Decrypt=True,Wrap=True,Unwrap=True}'
```

```
{
    "Key": {
        "CreateTimestamp": "2022-11-15T11:15:42.358000-08:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
nsq2i3mbg6sn775f",
        "KeyAttributes": {
            "KeyAlgorithm": "RSA_2048",
            "KeyClass": "ASYMMETRIC_KEY_PAIR",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
```

```
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION"
        },
        "KeyCheckValue": "40AD487F",
        "KeyCheckValueAlgorithm": "CMAC",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-11-15T11:15:42.182000-08:00"
    }
}
```

PIN 検証値 (PVV) キーの生成

Example

このコマンドは、PVV 値 (PIN 検証値と呼ばれる) を生成するための 3KEY TDES キーを生成します。このキーを使用して PVV 値を生成できます。この値は、後で計算される PVV と比較できます。レスポンスは、後続の呼び出し用の ARN や KCV (Key Check Value) を含むリクエストパラメータをエコーバックします。

```
$ aws payment-cryptography create-key --exportable/
--key-attributes KeyAlgorithm=TDES_3KEY,KeyUsage=TR31_V2_VISA_PIN_VERIFICATION_KEY,/
KeyClass=SYMMETRIC_KEY,KeyModesOfUse='{Generate=true,Verify=true}'
```

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T10:22:59.668000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
j4u4cmnzkelhc6yb",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": false,
```

```
                "Encrypt": false,
                "Generate": true,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY"
        },
        "KeyCheckValue": "5132",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-27T10:22:59.614000-07:00"
    }
}
```

キーのリスト

キーのリスト は、このアカウントとリージョンの呼び出し元がアクセスできるキーのリストを表示します。

Example

```
$ aws payment-cryptography list-keys
```

```
 {"Keys": [
    {
        "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
        "Enabled": false,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": true,
```

```
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
        },
        "KeyCheckValue": "369D",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStopTimestamp": "2022-10-27T14:19:42.488000-07:00"
    }
    ]
}
```

キーの有効化と無効化

AWS Payment Cryptography キーを無効化および再有効化できます。キー を作成すると、そのキーはデフォルトで有効になります。キーを無効にすると、再度有効にするまで暗号化オペレーションで使用できなくなります。使用開始/停止コマンドはすぐに有効になるため、変更を加える前に使用状況を確認することをおすすめします。オプションの `timestamp` パラメータを使用して、変更 (使用開始または停止) を将来有効になるように設定することもできます。

Payment Cryptography キーは一時的で簡単に元に戻すことができるため、 AWS 破壊的で元に戻せないアクションである AWS Payment Cryptography キーを削除するよりも安全な代替手段です。AWS Payment Cryptography キーの削除を検討している場合は、最初に無効にし、そのキーを使用して将来データを暗号化または復号する必要がないことを確認します。

トピック

- キーの使用開始
- キーの使用停止

キーの使用開始

暗号オペレーションにキーを使用するには、キーの使用が有効になっている必要があります。キーが有効になっていない場合は、このオペレーションを使用してそのキーを使用可能にすることができます。フィールド UsageStartTimestamp は、キーがいつアクティブになったか、いつアクティブになるかを表します。これは、有効なトークンの場合は過去のもので、アクティベーションが保留になっている場合は将来のものになります。

Example

この例では、キーを使用できるようにキーを有効化するように要求しています。レスポンスにはキー情報が含まれており、有効フラグは true に遷移しています。これはキーのリストのレスポンスオブジェクトにも反映されます。

```
$ aws payment-cryptography start-key-usage --key-identifier "arn:aws:payment-
cryptography:us-east-2:111122223333:key/alsuwfxug3pgy6xh"
```

```
{
      "Key": {
      "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
      "Enabled": true,
      "Exportable": true,
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
      "KeyAttributes": {
          "KeyAlgorithm": "TDES_3KEY",
          "KeyClass": "SYMMETRIC_KEY",
          "KeyModesOfUse": {
              "Decrypt": true,
              "DeriveKey": false,
              "Encrypt": true,
              "Generate": false,
              "NoRestrictions": false,
              "Sign": false,
              "Unwrap": true,
              "Verify": false,
              "Wrap": true
          },
          "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
      },
      "KeyCheckValue": "369D",
```

```
      "KeyCheckValueAlgorithm": "ANSI_X9_24",
      "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
      "KeyState": "CREATE_COMPLETE",
      "UsageStartTimestamp": "2022-10-27T14:09:59.468000-07:00"
  }
}
```

キーの使用停止

キーを使用する予定がなくなった場合は、キーの使用を停止して、今後の暗号化オペレーションを防ぐことができます。このオペレーションは永続的なものではないため、キーの使用開始を使用することで元に戻すことができます。また、キーを将来無効にするように設定することもできます。フィールド UsageStopTimestamp は、キーがいつ無効になったか、または無効になるかを表します。

Example

この例では、キーの使用を将来停止するように要求されています。実行後、キーの使用開始によって再度有効化されない限り、このキーは暗号化オペレーションに使用できません。レスポンスにはキー情報が含まれており、有効化フラグは false に移行しています。これはキーのリストのレスポンスオブジェクトにも反映されます。

```
$ aws payment-cryptography stop-key-usage --key-identifier "arn:aws:payment-
cryptography:us-east-2:111122223333:key/alsuwfxug3pgy6xh"
```

```
{
  "Key": {
      "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
      "Enabled": false,
      "Exportable": true,
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
      "KeyAttributes": {
          "KeyAlgorithm": "TDES_3KEY",
          "KeyClass": "SYMMETRIC_KEY",
          "KeyModesOfUse": {
              "Decrypt": true,
              "DeriveKey": false,
              "Encrypt": true,
              "Generate": false,
              "NoRestrictions": false,
              "Sign": false,
```

```
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
        },
        "KeyCheckValue": "369D",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStopTimestamp": "2022-10-27T14:09:59.468000-07:00"
    }
}
```


キーの削除

AWS Payment Cryptography キーを削除すると、キーマテリアルとキーに関連付けられたすべてのメタデータが削除され、キーのコピーが AWS Payment Cryptography の外部で利用可能でない限り、元に戻せません。キーを削除すると、そのキーで暗号化されたデータを復号できなくなります。これは、そのデータが回復不能になることを意味します。キーの削除は、そのキーをもう使用しないことが確実である場合にのみ行ってください。不明な場合は、削除するのではなく、キーを無効化することを検討します。後で再度使用する必要がある場合は、無効化されたキーを再度有効にできますが、別のソースから再インポートしない限り、削除された AWS Payment Cryptography キーを復元することはできません。

キーを削除する前に、キーが不要になったことを確認する必要があります。 AWS Payment Cryptography は CVV2 などの暗号化オペレーションの結果を保存せず、永続的な暗号化マテリアルにキーが必要かどうかを判断できません。

AWS Payment Cryptography は、削除を明示的にスケジュールし、必須の待機期間が終了しない限り、アクティブな AWS アカウントに属するキーを削除しません。

ただし、次のいずれかの理由で AWS Payment Cryptography キーを削除することもできます。

- 不要になったきーのキーライフサイクルを完了する
- 未使用の AWS Payment Cryptography キーの管理オーバーヘッドを回避するには

> **Note**
>
> を閉じたり削除したりすると AWS アカウント、 AWS Payment Cryptography キーにアクセスできなくなります。Payment Cryptography AWS キーの削除は、アカウントの閉鎖とは別にスケジュールする必要はありません。

AWS Payment Cryptography は、 AWS Payment Cryptography キーの削除をスケジュールし、 AWS Payment Cryptography キーが実際に削除されたときに、AWS CloudTrailログにエントリを記録します。

待機期間について

キーの削除は元に戻せないため、 AWS Payment Cryptography では 3 ~ 180 日間の待機期間を設定する必要があります。デフォルトの待機時間は、7 日です。

ただし、実際の待機期間は、スケジュールした待機期間よりも最大 24 時間長くなる場合があります。 AWS Payment Cryptography キーが削除される実際の日時を取得するには、 GetKey オペレーションを使用します。必ずタイムゾーンをメモしておきます。

待機期間中、 AWS Payment Cryptography のキーステータスとキーステータスは削除保留中です。

> **Note**
>
> 削除保留中の AWS Payment Cryptography キーは、暗号化オペレーション では使用できません。

待機期間が終了すると、 AWS Payment Cryptography は AWS Payment Cryptography キー、そのエイリアス、および関連するすべての AWS Payment Cryptography メタデータを削除します。

待機期間を使用して、現在または将来に AWS Payment Cryptography キーが不要になるようにします。待機期間中にキーが必要になった場合は、待機期間の終了前にキーの削除をキャンセルできます。待機期間の終了後は、キーの削除はキャンセルできず、 サービスはキーを削除します。

Example

この例では、キーの削除をリクエストしています。基本的なキー情報に加えて、2 つの関連フィールドは、キーの状態が DELETE_PENDING に変更されていることと deletePendingTimestamp、キーが現在削除がスケジュールされているタイミングを表します。

```
$ aws payment-cryptography delete-key \
                    --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h
```


```
  {
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyAlgorithm": "TDES_3KEY",
            "KeyModesOfUse": {
                "Encrypt": false,
                "Decrypt": false,
                "Wrap": false,
                "Unwrap": false,
                "Generate": true,
                "Sign": false,
                "Verify": true,
                "DeriveKey": false,
                "NoRestrictions": false
            }
        },
        "KeyCheckValue": "",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": false,
        "Exportable": true,
        "KeyState": "DELETE_PENDING",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "CreateTimestamp": "2023-06-05T12:01:29.969000-07:00",
        "UsageStopTimestamp": "2023-06-05T14:31:13.399000-07:00",
        "DeletePendingTimestamp": "2023-06-12T14:58:32.865000-07:00"
    }
}
```

Example

この例では、保留中の削除がキャンセルされます。正常に完了すると、そのキーは以前のスケジュールに従って削除されなくなります。レスポンスには基本的なキー情報が含まれています。さらに、KeyState および deletePendingTimestamp の 2 つの関連フィールドが変更されました。KeyState は CREATE_COMPLETE の値に戻されるが、DeletePendingTimestamp は削除されます。

```
$ aws payment-cryptography restore-key --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h
```

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyAlgorithm": "TDES_3KEY",
            "KeyModesOfUse": {
                "Encrypt": false,
                "Decrypt": false,
                "Wrap": false,
                "Unwrap": false,
                "Generate": true,
                "Sign": false,
                "Verify": true,
                "DeriveKey": false,
                "NoRestrictions": false
            }
        },
        "KeyCheckValue": "",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": false,
        "Exportable": true,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "CreateTimestamp": "2023-06-08T12:01:29.969000-07:00",
        "UsageStopTimestamp": "2023-06-08T14:31:13.399000-07:00"
    }
}
```

キーのインポートとエクスポート

AWS Payment Cryptography キーは、他のソリューションからインポートすることも、他のソリューション (他の HSMs。インポートおよびエクスポート機能を使用してサービスプロバイダーとキーを交換するのが一般的な使用例です。クラウドサービスである AWS Payment Cryptography は、適用されるコンプライアンスとコントロールを維持しながら、キー管理に最新の電子的アプローチを採用しています。長期的な目標は、ペーパーベースの主要コンポーネントから標準ベースの電子的なキー交換手段に移行することです。

キー暗号化キー (KEK) 交換

AWS Payment Cryptography は、十分に確立された ANSI X9.24 TR-34 標準を使用して、初期キー交換にパブリックキー暗号化 (RSA) の使用を推奨します。この初期キータイプの共通名には、キー暗号化キー (KEK)、ゾーンマスターキー (ZMK)、ゾーンコントロールマスターキー (ZCMK) が含まれます。システムまたはパートナーが TR-34 をまだサポートしていない場合は、RSA ラップ/アンラップ の利用を検討することもできます。

すべてのパートナーが電子キー交換をサポートするまでペーパーキーコンポーネントの処理を続ける必要がある場合は、この目的でオフライン HSM を保持することを検討してください。

> **Note**
>
> 独自のテストキーをインポートしたい場合は、Github のサンプルプロジェクトをチェックしてください。他のプラットフォームからキーをインポート/エクスポートする方法については、そのプラットフォームのユーザーガイドを参照してください。

ワーキングキー (WK) 交換

AWS Payment Cryptography は、関連する業界標準 (ANSI X9.24 TR 31-2018) を使用して作業キーを交換します。TR-31 は KEK が既に交換されていることを前提としています。これは、キーの内容をキーのタイプと用途に常に暗号的に結び付けるという PCI PIN の要件と一致しています。作業キーには、アクワイアラー作業キー、発行者作業キー、BDK、IPEK など、さまざまな名前があります。

トピック

- キーのインポート

- キーのエクスポート

キーのインポート

> **Important**
>
> 例としては、AWS CLI V2 の最新バージョンが必要になる場合があります。開始する前に、最新バージョン にアップグレードしていることを確認してください。

トピック

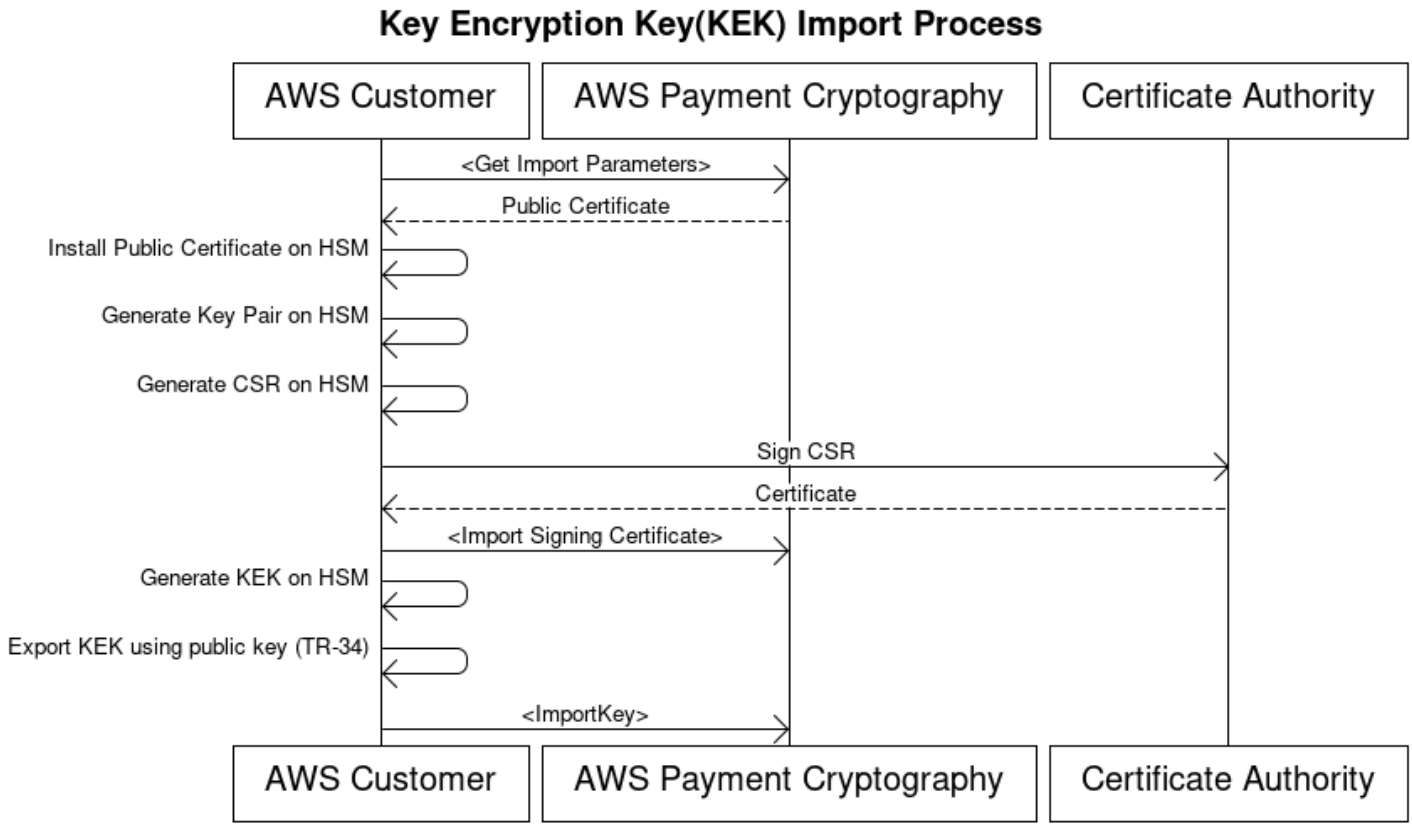
- 対称キーのインポート
- 非対称 (RSA) キーのインポート

対称キーのインポート

トピック

- 非対称手法 (TR-34) によるキーのインポート
- 非対称手法を使用したキーのインポート (RSA Unwrap)
- あらかじめ設定されているキー交換キー (TR-31) を使用して対称キーをインポートします。

非対称手法 (TR-34) によるキーのインポート



概要：TR-34 は RSA 非対称暗号を利用して、交換用の対称キーを暗号化し、データの出所を確認 (署名) します。これにより、ラップされたキーの機密性 (暗号化) と完全性 (署名) の両方が保証されます。

独自のキーをインポートしたい場合は、Github のサンプルプロジェクトをチェックしてください。他のプラットフォームからキーをインポート/エクスポートする方法については、そのプラットフォームのユーザーガイドを参照してください。

1. インポートの初期化コマンドを呼び出す

   `get-parameters-for-import` を呼び出して、インポートプロセスを初期化します。この API は、キーをインポートする目的でキーペアを生成し、キーに署名して、証明書と証明書ルートを返します。最終的には、エクスポートするキーはこのキーを使用して暗号化する必要があります。TR-34 の用語では、これは KRD 証明書と呼ばれています。これらの証明書は有効期間が短く、この目的のみを目的としていることに注意してください。

2. 公開証明書をキーソースシステムにインストールする

   多くの HSM では、ステップ 1 で生成した公開証明書を使用してキーをエクスポートするために、そのパブリック証明書をインストール/ロード/信頼する必要がある場合があります。

3. パブリックキーを生成し、証明書のルートを AWS Payment Cryptography に提供する

   送信されたペイロードの整合性を確保するために、送信側 (キー分散ホスト (KDH) と呼ばれる) によって署名されます。送信側はこの目的のためにパブリックキーを生成し、 AWS Payment Cryptography に返すことができるパブリックキー証明書 (X509) を作成します。 AWS Private CA は証明書を生成するオプションの 1 つですが、使用する認証局に制限はありません。

   証明書を取得したら、 の `importKey` コマンドと および を使用して、ルート証明書を AWS Payment Cryptography KeyMaterialType `ROOT_PUBLIC_KEY_CERTIFICATE` KeyUsageType にロードします`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`。

4. ソースシステムからキーをエクスポートする

   多くの HSMs および関連システムは、TR-34 標準を使用してキーをエクスポートする機能をサポートしています。ステップ 1 の公開キーを KRD (暗号化) 証明書として指定し、ステップ 3 の公開キーを KDH (署名) 証明書として指定する必要があります。 AWS Payment Cryptography にインポートするには、TR-34 Diebold 形式とも呼ばれる TR-34.2012 以外の 2 つのパス形式を指定します。

5. インポートキーを呼び出す

   最後のステップとして、 の を使用して importKey API KeyMaterialType を呼び出しますTR34_KEY_BLOCK。`certificate-authority-public-key-identifier` は、ステップ 3 でインポートしたルート CA の KeyArn で、`key-material` は、ステップ 4 でラップされたキーマテリアルになり、`signing-key-certificate` は、ステップ 3 のリーフ証明書です。また、ステップ 1 のインポートトークンを指定する必要があります。

6. 暗号化オペレーションやその後のインポートのためにインポートしたキーを使用する

   インポートされた KeyUsage が TR31_K0_KEY_ENCRYPTION_KEY の場合、このキーは TR-31 を使用した後続のキーインポートに使用できます。キータイプが他のタイプ (TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY など) だった場合は、そのキーを暗号オペレーションに直接使用できます。

非対称手法を使用したキーのインポート (RSA Unwrap)

概要： AWS Payment Cryptography は、TR-34 が不可能な場合、キー交換のための RSA ラップ/アンラップをサポートしています。TR-34 と同様に、この手法では RSA 非対称暗号化を使用して対称

キーを暗号化し、交換します。ただし、TR-34 とは異なり、このメソッドには送信側によって署名されたペイロードはありません。また、この RSA ラップ手法は、キーブロックを含めないことにより、転送中にキーメタデータの整合性を維持しません。

> **Note**
>
> RSA ラップを使用して、TDES および AES-128 キーをインポートまたはエクスポートできます。

1. インポートの初期化コマンドを呼び出す

   `get-parameters-for-import` を呼び出して、キーマテリアルタイプが KEY_CRYPTOGRAM のインポートプロセスを初期化します。TDES キーの交換時に RSA_2048 に WrappingKeyAlgorithm することができます。TDES または AES-128 キーを交換する場合は、RSA_3072 または RSA_4096 を使用できます。この API は、キーのインポートのためにキーペアを生成し、証明書ルートを使用してキーに署名し、証明書と証明書ルートの両方を返します。最終的には、エクスポートするキーはこのキーを使用して暗号化する必要があります。これらの証明書は有効期間が短く、この目的のみを目的としていることに注意してください。

   ```
   $ aws payment-cryptography get-parameters-for-import --key-material-type
    KEY_CRYPTOGRAM --wrapping-key-algorithm RSA_4096
   ```

   ```
   {
       "ImportToken": "import-token-bwxli6ocftypneu5",
       "ParametersValidUntilTimestamp": 1698245002.065,
       "WrappingKeyCertificateChain": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0....",
       "WrappingKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0....",
       "WrappingKeyAlgorithm": "RSA_4096"
   }
   ```

2. 公開証明書をキーソースシステムにインストールする

   多くの HSMs、それを使用してキーをエクスポートするために、ステップ 1 で生成されたパブリック証明書 (およびそのルート) をインストール/ロード/信頼する必要がある場合があります。

3. ソースシステムからキーをエクスポートする

   多くの HSMsおよび関連システムは、RSA ラップを使用してキーをエクスポートする機能をサポートしています。ステップ 1 のパブリックキーを (暗号化) 証明書 (WrappingKey証明書) として指定します。信頼チェーンが必要な場合は、 WrappingKeyCertificateChain ステップ 1 のレスポンスフィールドに含まれています。HSM からキーをエクスポートするときは、RSA、パディングモード = PKCS#1 v2.2 OAEP (SHA 256 または SHA 512 を使用) の形式を指定します。

4. インポートキーを呼び出す

   最後のステップとして、 の を使用して importKey API KeyMaterialType を呼び出しますKeyMaterial。ステップ 1 のインポートトークンと、ステップ 3 の `key-material` (ラップされたキーマテリアル) が必要です。RSA ラップはキーブロックを使用しないため、キーパラメータ (キーの使用法など) を指定する必要があります。

```
$ cat import-key-cryptogram.json
              {
                  "KeyMaterial": {
                      "KeyCryptogram": {
                          "Exportable": true,
                          "ImportToken": "import-token-bwxli6ocftypneu5",
                          "KeyAttributes": {
                              "KeyAlgorithm": "AES_128",
                              "KeyClass": "SYMMETRIC_KEY",
                              "KeyModesOfUse": {
                                  "Decrypt": true,
                                  "DeriveKey": false,
                                  "Encrypt": true,
                                  "Generate": false,
                                  "NoRestrictions": false,
                                  "Sign": false,
                                  "Unwrap": true,
                                  "Verify": false,
                                  "Wrap": true
                              },
                              "KeyUsage": "TR31_K0_KEY_ENCRYPTION_KEY"
                          },
                          "WrappedKeyCryptogram": "18874746731....",
                          "WrappingSpec": "RSA_OAEP_SHA_256"
                      }
                  }
              }
```

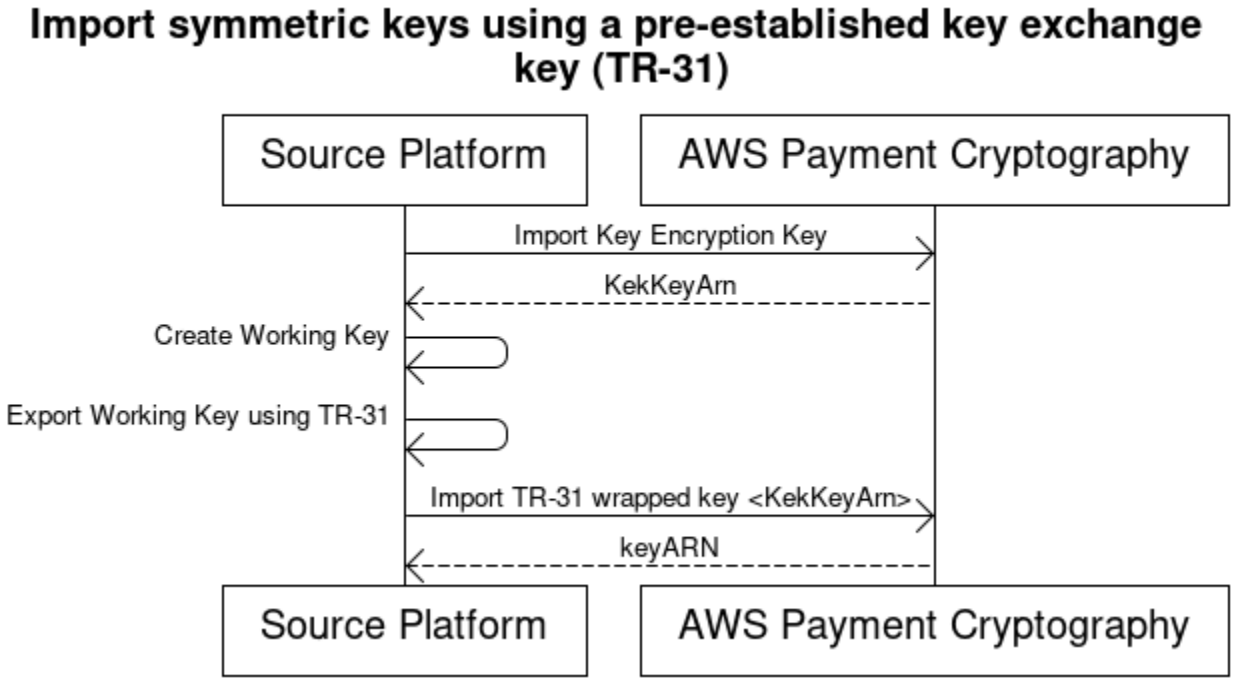
```
$ aws payment-cryptography import-key --cli-input-json file://import-key-
cryptogram.json
```

```
{
      "Key": {
          "KeyOrigin": "EXTERNAL",
          "Exportable": true,
          "KeyCheckValue": "DA1ACF",
          "UsageStartTimestamp": 1697643478.92,
          "Enabled": true,
          "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
          "CreateTimestamp": 1697643478.92,
          "KeyState": "CREATE_COMPLETE",
          "KeyAttributes": {
              "KeyAlgorithm": "AES_128",
              "KeyModesOfUse": {
                  "Encrypt": true,
                  "Unwrap": true,
                  "Verify": false,
                  "DeriveKey": false,
                  "Decrypt": true,
                  "NoRestrictions": false,
                  "Sign": false,
                  "Wrap": true,
                  "Generate": false
              },
              "KeyUsage": "TR31_K0_KEY_ENCRYPTION_KEY",
              "KeyClass": "SYMMETRIC_KEY"
          },
          "KeyCheckValueAlgorithm": "CMAC"
      }
  }
```

5. 暗号化オペレーションやその後のインポートのためにインポートしたキーを使用する

インポートされた KeyUsage が TR31_K0_KEY_ENCRYPTION_KEY の場合、このキーは TR-31 を使用した後続のキーインポートに使用できます。キータイプが他のタイプ (TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY など) だった場合は、そのキーを暗号オペレーションに直接使用できます。

あらかじめ設定されているキー交換キー (TR-31) を使用して対称キーをインポートします。



パートナーが複数のキーを交換する場合 (またはキーローテーションをサポートする場合) 、まずペーパーキーコンポーネントなどの手法を使用して、または Payment Cryptography AWS の場合は TR-34 を使用して、初期キー暗号化キー (KEK) を交換するのが一般的です。

KEK が確立されたら、このキーを使用して後続のキー (他の KEKs) を転送できます。 AWS Payment Cryptography は、HSM ベンダーによって広く使用およびサポートされている ANSI TR-31 を使用したこの種のキー交換をサポートしています。

1. キー暗号化キー (KEK) をインポートする

   KEK を既にインポートしていて、KeyArn (または KeyAlias) が使用可能であることを前提としています。
2. ソースプラットフォームでキーを作成する

   キーがまだ存在しない場合は、ソースプラットフォームでキーを作成します。逆に、 AWS Payment Cryptography でキーを作成し、代わりに `export` コマンドを使用することもできます。
3. ソースプラットフォームからキーをエクスポートする

   エクスポートするときは、エクスポート形式を必ず TR-31 として指定してください。ソースプラットフォームでは、エクスポートするキーと使用するキー暗号化キーの入力も求められます。

4. AWS Payment Cryptography にインポートする

importKey コマンドを呼び出す場合、 WrappingKeyIdentifier はキー暗号化キーの keyARN (またはエイリアス) で、 WrappedKeyBlock はソースプラットフォームからの出力です。

Example

```
$ aws payment-cryptography import-key \
            --key-material="Tr31KeyBlock={WrappingKeyIdentifier="arn:aws:payment-
cryptography:us-east-2:111122223333:key/ov6icy4ryas4zcza",\

 WrappedKeyBlock="D0112B0AX00E00002E0A3D58252CB67564853373D1EBCC1E23B2ADE7B15E967CC27B85D599
```

```
      {
        "Key": {
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
            "KeyAttributes": {
                "KeyUsage": "TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY",
                "KeyClass": "SYMMETRIC_KEY",
                "KeyAlgorithm": "AES_128",
                "KeyModesOfUse": {
                    "Encrypt": true,
                    "Decrypt": true,
                    "Wrap": true,
                    "Unwrap": true,
                    "Generate": false,
                    "Sign": false,
                    "Verify": false,
                    "DeriveKey": false,
                    "NoRestrictions": false
                }
            },
            "KeyCheckValue": "0A3674",
            "KeyCheckValueAlgorithm": "CMAC",
            "Enabled": true,
            "Exportable": true,
            "KeyState": "CREATE_COMPLETE",
            "KeyOrigin": "EXTERNAL",
            "CreateTimestamp": "2023-06-02T07:38:14.913000-07:00",
            "UsageStartTimestamp": "2023-06-02T07:38:14.857000-07:00"
```

```
        }
      }
```

非対称 (RSA) キーのインポート

RSA 公開キーのインポート

AWS Payment Cryptography は、X.509 証明書の形式のパブリック RSA キーのインポートをサポートしています。証明書をインポートするには、まずそのルート証明書をインポートする必要があります。すべての証明書は、インポート時に有効期限が切れていない必要があります。証明書は PEM 形式で、base64 でエンコードされている必要があります。

1. ルート証明書を AWS Payment Cryptography にインポートする

   Example

```
$ aws payment-cryptography import-key \
  --key-material='{"RootCertificatePublicKey":{"KeyAttributes":
{"KeyAlgorithm":"RSA_2048", \
  "KeyClass":"PUBLIC_KEY", "KeyModesOfUse":{"Verify":
 true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"}, \

 "PublicKeyCertificate":"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURKVENDQWcyZ0F3SUJBZ0lCWkR
```

```
  {
    "Key": {
        "CreateTimestamp": "2023-08-08T18:52:01.023000+00:00",
        "Enabled": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
zabouwe3574jysdl",
        "KeyAttributes": {
            "KeyAlgorithm": "RSA_2048",
            "KeyClass": "PUBLIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": false,
                "Encrypt": false,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
```

```
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
        },
        "KeyOrigin": "EXTERNAL",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2023-08-08T18:52:01.023000+00:00"
    }
}
```

2. パブリックキー証明書を AWS Payment Cryptography にインポートする

公開キーをインポートできるようになりました。公開キーのインポートには 2 つのオプションがあり、キーの目的が署名の検証である場合 (TR-34を使用してインポートする場合)、TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE を使用できます。別のシステムで使用するデータを暗号化する場合、TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION を使用できます。

Example

```
$ aws payment-cryptography import-key \
    --key-material='{"TrustedCertificatePublicKey":
{"CertificateAuthorityPublicKeyIdentifier":"arn:aws:payment-cryptography:us-
east-2:111122223333:key/zabouwe3574jysdl", \
    "KeyAttributes":
{"KeyAlgorithm":"RSA_2048","KeyClass":"PUBLIC_KEY","KeyModesOfUse":
{"Verify":true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"},\
    "PublicKeyCertificate":"LS0tLS1CRUdJTiB..."}}'
```

```
{
  "Key": {
      "CreateTimestamp": "2023-08-08T18:55:46.815000+00:00",
      "Enabled": true,
      "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/4kd6xud22e64wcbk",
      "KeyAttributes": {
          "KeyAlgorithm": "RSA_4096",
          "KeyClass": "PUBLIC_KEY",
          "KeyModesOfUse": {
```

```
                "Decrypt": false,
                "DeriveKey": false,
                "Encrypt": false,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
        },
        "KeyOrigin": "EXTERNAL",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2023-08-08T18:55:46.815000+00:00"
    }
}
```


キーのエクスポート

トピック

- 対称キーのエクスポート
- 非対称 (RSA) キーのエクスポート

対称キーのエクスポート

> **Important**
>
> 例としては、AWS CLI V2 の最新バージョンが必要になる場合があります。開始する前に、最新バージョン にアップグレードしていることを確認してください。

トピック

- 非対称技術 (TR-34) を使用してキーをエクスポートする
- 非対称手法を使用したキーのエクスポート (RSA ラップ)
- あらかじめ設定されているキー交換キー (TR-31) を使用して対称キーをエクスポートする
- DUKPT 初期キーのエクスポート (IPEK/IK)

非対称技術 (TR-34) を使用してキーをエクスポートする

概要：TR-34 は RSA 非対称暗号を利用して、交換用の対称キーを暗号化し、データの出所を確認 (署名) します。これにより、ラップされたキーの機密性 (暗号化) と完全性 (署名) の両方が保証されます。エクスポート時に、 AWS Payment Cryptography がキー分散ホスト (KDH) になり、ターゲットシステムがキー受信デバイス (KRD) になります。

1. エクスポートの初期化コマンドを呼び出す

   `get-parameters-for-export` を呼び出してエクスポートプロセスを初期化します。この API は、キーをエクスポートする目的でキーペアを生成し、キーに署名して、証明書と証明書ルートを返します。最終的には、このコマンドで生成された秘密キーがエクスポートペイロードの署名に使用されました。TR-34 の用語では、これは KDH 署名証明書として知られています。これらの証明書は有効期間が短く、この目的のみを目的としていることに注意してください。パラメータ ParametersValidUntilTimestampは期間を指定します。

   注：証明書はすべて Base64 でエンコードされた形式で返されます。

   Example

```
$ aws payment-cryptography get-parameters-for-export \
                    --signing-key-algorithm RSA_2048 --key-material-type
 TR34_KEY_BLOCK
```

```
{
               "SigningKeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2RENDQXFTZ0F3SUJBZ0lRZFAzSzNHNEFKT0I4WTNpTmUvYl
               "SigningKeyCertificateChain":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThtZ0F3SUJBZ0lSQUtlN2piaHFKZjJPd3FGUWI5c3
               "SigningKeyAlgorithm": "RSA_2048",
               "ExportToken": "export-token-au7pvkbsq4mbup6i",
               "ParametersValidUntilTimestamp": "2023-06-13T15:40:24.036000-07:00"
}
```

2. AWS Payment Cryptography 証明書を受信システムにインポートする

   必要に応じて、ステップ 1 で提供した証明書チェーンを受信側のシステムにインポートします。

3. キーペアを生成し、パブリック証明書を作成し、証明書のルートを AWS Payment Cryptography に提供します。

送信されるペイロードの機密性を確保するために、ペイロードは送信側 (キー分散ホスト (KDH) と呼ばれる) によって暗号化されます。受信側 (通常は HSM またはパートナーの HSM) はこの目的のためにパブリックキーを生成し、 AWS Payment Cryptography に返すことができるパブリックキー証明書 (x.509) を作成する必要があります。 AWS Private CA は証明書を生成するための 1 つのオプションですが、使用する認証局に制限はありません。

証明書を取得したら、 の `ImportKey` コマンドと および を使用して、ルート証明書を AWS Payment Cryptography KeyMaterialType `ROOT_PUBLIC_KEY_CERTIFICATE` KeyUsageType にロードします`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`。

この証明書 KeyUsageType の は、ルートキーであり、リーフ証明書の署名に使用されるため、TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE です。インポート/エクスポート用のリーフ証明書は AWS Payment Cryptography にインポートされませんが、インラインで渡されます。

> **Note**
>
> ルート証明書が以前にインポートされたことがある場合は、このステップは省略できます。

4. エクスポートキーを呼び出す

最後のステップとして、 KeyMaterialType の を使用して ExportKey API を呼び出します`TR34_KEY_BLOCK`。`certificate-authority-public-key-identifier` はステップ 3 のルート CA インポートの KeyArn になり、`WrappingKeyCertificate` はステップ 3 のリーフ証明書になり、`export-key-identifier` はエクスポートする KeyArn (またはエイリアス) になります。また、ステップ 1 のエクスポートトークンを指定する必要があります。

非対称手法を使用したキーのエクスポート (RSA ラップ)

概要： AWS Payment Cryptography は、TR-34 がカウンターパーティが利用できるオプションでない場合、キー交換のための RSA ラップ/アンラップをサポートします。TR-34 と同様に、この手法では RSA 非対称暗号化を使用して対称キーを暗号化し、交換します。ただし、TR-34 とは異なり、このメソッドには送信側によって署名されたペイロードはありません。また、この RSA ラップ手法に

は、転送中のキーメタデータの整合性を維持するために使用されるキーブロックは含まれていません。

> **Note**
>
> RSA ラップを使用して TDES キーと AES-128 キーをエクスポートできます。

1. 受信システムで RSA キーと証明書を生成する

   ラップされたキーの受信に使用される RSA キーを作成 (または識別) します。 AWS Payment Cryptography では、X.509 証明書形式のキーが必要です。証明書は、 AWS Payment Cryptography にインポートされた (またはインポートできる) ルート証明書によって署名する必要があります。

2. AWS Payment Cryptography にルートパブリック証明書をインストールする

```
$ aws payment-cryptography  import-key --key-material='{"RootCertificatePublicKey":
{"KeyAttributes":{"KeyAlgorithm":"RSA_4096","KeyClass":"PUBLIC_KEY","KeyModesOfUse":
{"Verify":
 true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"},"PublicKeyCertificate":"LS
```

```
    {
        "Key": {
        "CreateTimestamp": "2023-09-14T10:50:32.365000-07:00",
        "Enabled": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
nsq2i3mbg6sn775f",
        "KeyAttributes": {
        "KeyAlgorithm": "RSA_4096",
        "KeyClass": "PUBLIC_KEY",
        "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,
        "Encrypt": false,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
        "Verify": true,
        "Wrap": false
```

```
        },
        "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
        },
        "KeyOrigin": "EXTERNAL",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2023-09-14T10:50:32.365000-07:00“
        }
       }
```

3. コールエクスポートキー

次に、リーフ証明書を使用してキーをエクスポートするように AWS Payment Cryptography に指示します。以前にインポートしたルート証明書の ARN、エクスポートに使用するリーフ証明書、およびエクスポートする対称キーを指定します。出力は、対称キーの 16 進エンコードされたバイナリラップ (暗号化) バージョンになります。

```
$ cat export-key.json
```

```
  {
      "ExportKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/tqv5yij6wtxx64pi",
      "KeyMaterial": {
          "KeyCryptogram": {
              "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-
cryptography:us-east-2:111122223333:key/zabouwe3574jysdl",
              "WrappingKeyCertificate": "LS0tLS1CRUdJTiBD...",
              "WrappingSpec": "RSA_OAEP_SHA_256"
          }
      }
  }
```

```
$ aws payment-cryptography export-key --cli-input-json file://export-key.json
```

```
{
      "WrappedKey": {
          "KeyMaterial":
 "18874746731E9E1C4562E4116D1C2477063FCB08454D757D81854AEAEE0A52B1F9D303FA29C02DC82AE7853
          "WrappedKeyMaterialFormat": "KEY_CRYPTOGRAM"
      }
```

```
}
```

4. 受信システムにキーをインポートする

   多くの HSMs および関連システムは、RSA アンラップ (AWS Payment Cryptography を含む) を使用してキーをインポートする機能をサポートしています。そのためには、ステップ 1 のパブリックキーを (暗号化) 証明書として指定します。形式は RSA、パディングモード = PKCS#1 v2.2 OAEP (SHA 256 を使用) として指定する必要があります。正確な用語は HSM によって異なる場合があります。

   > **Note**
   >
   > AWS Payment Cryptography は、hexBinaryでラップされたキーを出力します。システムで base64 のような別のバイナリ表現が必要な場合は、インポート前に 形式を変換する必要がある場合があります。

あらかじめ設定されているキー交換キー (TR-31) を使用して対称キーをエクスポートする

パートナーが複数のキーを交換する場合 (またはキーローテーションをサポートする場合) 、まずペーパーキーコンポーネントなどの手法を使用して、または Payment Cryptography AWS の場合は TR-34 を使用して、初期キー暗号化キー (KEK) を交換するのが一般的です。KEK が確立されたら、このキーを使用して後続のキー (他の KEK を含む) を転送できます。 AWS Payment Cryptography は、HSM ベンダーによって広く使用およびサポートされている ANSI TR-31 を使用したこの種のキー交換をサポートしています。

1. キー暗号化キー (KEK) を交換する

   既に KEK を交換していて、KeyArn (または KeyAlias) を使用できることを前提としています。

2. AWS Payment Cryptography でキーを作成する

   このキーが存在していない場合は、作成します。逆に、他のシステムでキーを作成し、代わりにインポートコマンドを使用することもできます。

3. AWS Payment Cryptography からキーをエクスポートする

   エクスポート時の形式は TR-31 になります。API を呼び出す際には、エクスポートするキーと使用するラップキーを指定します。

```
$ aws payment-cryptography export-key --key-material='{"Tr31KeyBlock":
 {"WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
```

```
ov6icy4ryas4zcza"}}' --export-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp
```

```
{
            "WrappedKey": {
                "KeyCheckValue": "73C263",
                "KeyCheckValueAlgorithm": "ANSI_X9_24",
                "KeyMaterial":
 "D0144K0AB00E0000A24D3ACF3005F30A6E31D533E07F2E1B17A2A003B338B1E79E5B3AD4FBF7850FACF9A37844
                "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK"
            }
     }
```

4. システムにインポートする

   ユーザーまたはパートナーは、システム上のインポートキー実装を使用してキーをインポートします。

DUKPT 初期キーのエクスポート (IPEK/IK)

DUKPT を使用する場合、ターミナルのフリートに対して 1 つの基本導出キー (BDK) を生成できます。ただし、ターミナルは元の BDK にはアクセスできませんが、それぞれ IPEK または初期キー (IK) と呼ばれる一意の初期ターミナルキーが挿入されます。各 IPEK は BDK から派生したキーであり、ターミナルごとに一意であることが意図されていますが、元の BDK から派生しています。この計算の派生データは、キーシリアル番号 (KSN) と呼ばれます。X9.24 に従い、TDES の場合、10 バイトの KSN は通常、キーセット ID に 24 ビット、ターミナル ID に 19 ビット、トランザクションカウンターに 21 ビットで構成されます。AES の場合、12 バイトの KSN は通常、BDK ID の場合は 32 ビット、取得識別子 (ID) の場合は 32 ビット、トランザクションカウンターの場合は 32 ビットで構成されます。

AWS Payment Cryptography は、これらの初期キーを生成してエクスポートするメカニズムを提供します。生成されたキーは、TR-31, TR-34RSA ラップメソッドを使用してエクスポートできます。IPEK キーは保持されず、 AWS Payment Cryptography での以降のオペレーションには使用できません。

AWS Payment Cryptography では、KSN の最初の 2 つの部分間の分割は強制されません。取得識別子を BDK とともに保存する場合は、この目的で AWS タグ機能を使用できます。

> **Note**
>
> KSN のカウンター部分 (AES DUKPT の場合は 32 ビット) は、IPEK/IK の取得には使用されません。したがって、12345678901234560001 と 12345678901234569999 の入力は同じ IPEK を出力します。

```
$ aws payment-cryptography export-key --key-material='{"Tr31KeyBlock":
 {"WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza"}}' --export-key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --export-attributes
 'ExportDukptInitialKey={KeySerialNumber=12345678901234560001}'
```

```
{
    "WrappedKey": {
        "KeyCheckValue": "73C263",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyMaterial":
 "B0096B1TX00S000038A8A06588B9011F0D5EEF1CCAECFA6962647A89195B7A98BDA65DDE7C57FEA507559AF2A5D60
        "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK"
    }
}
```

非対称 (RSA) キーのエクスポート

get-public-key-certificate を呼び出して、パブリックキーを証明書形式でエクスポートします。この API は、証明書と base64 形式でエンコードされたルート証明書をエクスポートします。

注：この API は同等性ではありません。基になるキーが同じであっても、それ以降の呼び出しでは異なる証明書が生成される可能性があります。

Example

```
$ aws payment-cryptography get-public-key-certificate \
        –key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5dza7xqd6soanjtb
```

```
{
        "KeyCertificate": "LS0tLS1CRUdJTi...",
```

```
        "KeyCertificateChain": "LS0tLS1CRUdJT...“
}
```

エイリアスの使用

エイリアスは Payment Cryptography AWS キーのわかりやすい名前です。例えば、エイリアスを使用すると、arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h ではなく、KMS キーを alias/test-key として参照できます。

エイリアスを使用すると、ほとんどのキー管理 (コントロールプレーン) オペレーションと暗号化 (データプレーン) オペレーションでキーを識別できます。

ポリシーを編集したり、許可を管理したりすることなく、エイリアスに基づいて AWS Payment Cryptography キーへのアクセスを許可または拒否することもできます。この機能は、属性ベースのアクセス制御 (ABAC)の sa-bisuno サポートの一部です。

エイリアスの機能の強みは、エイリアスに関連付けられているキーをいつでも変更できることです。エイリアスを使用すると、コードの記述と保守が容易になります。例えば、エイリアスを使用して特定の AWS Payment Cryptography キーを参照し、 AWS Payment Cryptography キーを変更するとします。この場合は、単にエイリアスを別のキーに関連付けます。コードやアプリケーションの設定を変更する必要はありません。

エイリアスを使用すると、別の AWS リージョンで同じコードを再利用することも容易になります。複数のリージョンに同じ名前のエイリアスを作成し、各エイリアスをそのリージョンの AWS Payment Cryptography キーに関連付けます。コードが各リージョンで実行されると、エイリアスはそのリージョン内の関連付けられた AWS Payment Cryptography キーを参照します。

CreateAlias API を使用して AWS Payment Cryptography キーのエイリアスを作成できます。

AWS Payment Cryptography API は、各アカウントとリージョンのエイリアスを完全に制御します。API には、エイリアスの作成 (CreateAlias) 、エイリアス名とリンクされた keyARN の表示 (リストエイリアス) 、エイリアスに関連付けられた AWS Payment Cryptography キーの変更 (更新エイリアス) 、エイリアスの削除 (削除エイリアス) を行うオペレーションが含まれています。

トピック

- エイリアスについて
- アプリケーションでのエイリアスの使用
- 関連 API

エイリアスについて

AWS Payment Cryptography でのエイリアスの動作について説明します。

エイリアスは独立した AWS リソースです。

エイリアスは AWS Payment Cryptography キーのプロパティではありません。エイリアスに対して実行するアクションは、エイリアスに関連付けられた キーには影響しません。 AWS Payment Cryptography キーのエイリアスを作成し、エイリアスを更新して、別の AWS Payment Cryptography キーに関連付けることができます。関連付けられた AWS Payment Cryptography キーに影響を与えずにエイリアスを削除することもできます。 AWS Payment Cryptography キーを削除すると、そのキーに関連付けられているすべてのエイリアスが割り当てられなくなります。

IAM ポリシーでリソースとしてエイリアスを指定すると、ポリシーは関連する AWS Payment Cryptography キーではなくエイリアスを参照します。

各エイリアスにはわかりやすい名前が付いています

エイリアスを作成するときは、`alias/` によりプレフィックスが付いたエイリアス名を指定します。`alias/test_1234` の例

各エイリアスは、一度に 1 つの AWS Payment Cryptography キーに関連付けられます。

エイリアスとその AWS Payment Cryptography キーは、同じアカウントとリージョンに存在する必要があります。

AWS Payment Cryptography キーは複数のエイリアスに同時に関連付けることができますが、各エイリアスは 1 つのキーにのみマッピングできます。

例えば、この `list-aliases` 出力では、`alias/sampleAlias1` エイリアスが正確に 1 つのターゲット AWS Payment Cryptography キーに関連付けられていることが示されています。これは、`KeyArn` プロパティによって表されます。

```
$ aws payment-cryptography list-aliases
```

```
{
  "Aliases": [
      {
```

```
            "AliasName": "alias/sampleAlias1",
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
        }
    ]
}
```

複数のエイリアスを同じ AWS Payment Cryptography キーに関連付けることができます

例えば、alias/sampleAlias1; と alias/sampleAlias2 のエイリアスを同じキーに関連付けることができます。

```
$ aws payment-cryptography list-aliases
```

```
{
        "Aliases": [
            {
                "AliasName": "alias/sampleAlias1",
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
            },
            {
                "AliasName": "alias/sampleAlias2",
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
            }
        ]
    }
```

エイリアスは、アカウントとリージョン内で一意である必要があります

例えば、各アカウントとリージョンに alias/sampleAlias1 エイリアスを 1 つだけ持つことができます。エイリアスでは大文字と小文字が区別されますが、大文字と小文字だけが異なるエイリアスは使用しないことをお勧めします。エイリアス名は変更できません。ただし、エイリアスを削除して、目的の名前で新しいエイリアスを作成することはできます。

異なるリージョンに同じ名前のエイリアスを作成することができます

例えば、米国東部 (バージニア北部) に alias/sampleAlias2 エイリアスを持つことができ、米国西部 (オレゴン) に alias/sampleAlias2 エイリアスを持つことができます。各エイリ

アスは、そのリージョンの AWS Payment Cryptography キーに関連付けられます。コードが alias/finance-key のようなエイリアス名を参照している場合は、複数のリージョンで実行できます。各リージョンでは、異なるエイリアス/サンプルエイリアス2 が使用されます。詳細については、「アプリケーションでのエイリアスの使用」を参照してください。

エイリアスに関連付けられた AWS Payment Cryptography キーを変更できます。

UpdateAlias オペレーションを使用して、エイリアスを別の AWS Payment Cryptography キーに関連付けることができます。例えば、alias/sampleAlias2エイリアスが arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h AWS Payment Cryptography キーに関連付けられている場合は、arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64piキーに関連付けられるように更新できます。

> **Warning**
>
> AWS Payment Cryptography は、古いキーと新しいキーの属性がすべてキーの使用など同じであることを検証しません。別のキータイプで更新すると、アプリケーションで問題が発生する可能性があります。

一部のキーにはエイリアスがない

エイリアスはオプション機能であり、このような方法で環境を運用しない限り、すべてのキーにエイリアスがあるわけではありません。キーは create-alias コマンドを使用してエイリアスに関連付けることができます。また、update-alias オペレーションを使用してエイリアスに関連付けられている AWS Payment Cryptography キーを変更することや、delete-alias オペレーションを使用してエイリアスを削除することもできます。その結果、一部の AWS Payment Cryptography キーには複数のエイリアスがあり、一部のエイリアスにはない場合があります。

キーをエイリアスにマッピングする

create-alias コマンドを使用して、キー (ARN で表される) を 1 つ以上のエイリアスにマップできます。このコマンドは同一ではありません。エイリアスを更新するには、update-alias コマンドを使用してください。

```
$ aws payment-cryptography create-alias --alias-name alias/sampleAlias1 \
            --key-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h
```

```
{
    "Alias": {
        "AliasName": "alias/alias/sampleAlias1",
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
    }
}
```

アプリケーションでのエイリアスの使用

エイリアスを使用して、アプリケーションコード内の AWS Payment Cryptography キーを表すことができます。 AWS Payment Cryptography データオペレーションの `key-identifier`パラメータ、およびリストキーなどの他のオペレーションは、エイリアス名またはエイリアス ARN を受け入れます。

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier alias/
BIN_123456_CVK --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue2={CardExpiryDate=0123}
```

エイリアス ARN を使用する場合、 AWS Payment Cryptography キーへのエイリアスマッピングは AWS Payment Cryptography キーを所有するアカウントで定義され、リージョンごとに異なる場合があることに注意してください。

エイリアスの最も強力な使用法の 1 つは、アプリケーションを複数の AWS リージョンで実行する場合です。

リージョンごとに異なるバージョンのアプリケーションを作成することも、ディクショナリ、設定、またはスイッチステートメントを使用して、リージョンごとに適切な AWS Payment Cryptography キーを選択することもできます。ただし、各リージョンで同じエイリアス名を持つエイリアスを作成する方がはるかに簡単かもしれません。。エイリアス名では、大文字と小文字が区別されます。

関連 API

タグ

タグは、 AWS Payment Cryptography キーを整理するためのメタデータとして機能するキーと値のペアです。これらを使用すると、キーを柔軟に識別したり、1 つ以上のキーをグループ化したりできます。

キーを取得する

AWS Payment Cryptography キーは暗号化マテリアルの 1 つのユニットを表し、このサービスの暗号化オペレーションにのみ使用できます。 GetKeys API は を入力 KeyIdentifier として受け取り、キーのイミュータブル属性とミュータブル属性を返しますが、暗号化マテリアルは含まれません。

Example

```
$ aws payment-cryptography get-key --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h
```


```
{
  "Key": {
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
      "KeyAttributes": {
          "KeyUsage": "TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY",
          "KeyClass": "SYMMETRIC_KEY",
          "KeyAlgorithm": "AES_128",
          "KeyModesOfUse": {
              "Encrypt": true,
              "Decrypt": true,
              "Wrap": true,
              "Unwrap": true,
              "Generate": false,
              "Sign": false,
              "Verify": false,
              "DeriveKey": false,
              "NoRestrictions": false
          }
      },
      "KeyCheckValue": "0A3674",
      "KeyCheckValueAlgorithm": "CMAC",
      "Enabled": true,
      "Exportable": true,
      "KeyState": "CREATE_COMPLETE",
      "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
      "CreateTimestamp": "2023-06-02T07:38:14.913000-07:00",
      "UsageStartTimestamp": "2023-06-02T07:38:14.857000-07:00"
```

```
  }
}
```

キーペアに関連付けられた公開キー/証明書を取得する

[Get Public Key/Certificate] は、KeyArn で示される公開キーを返します。これは、 AWS Payment Cryptography で生成されたキーペアのパブリックキー部分でも、以前にインポートされたパブリックキーでもかまいません。最も一般的な使用例は、データを暗号化する外部サービスに公開キーを提供することです。その後、そのデータは AWS Payment Cryptography を利用するアプリケーションに渡され、そのデータは AWS Payment Cryptography 内で保護されたプライベートキーを使用して復号化できます。

このサービスは公開キーを公開証明書として返します。API 結果には、CA とパブリックキー証明書が含まれます。両方のデータ要素は base64 エンコードされています。

> **Note**
>
> 返される公開証明書は有効期間が短いもので、同等性を意図したものではありません。パブリックキー自体が変更されていなくても、API 呼び出しごとに異なる証明書を受け取る場合があります。

Example

```
$ aws payment-cryptography get-public-key-certificate  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/nsq2i3mbg6sn775f
```

```
{
  "KeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2VENDQXFXZ0F3SUJBZ0lSQUo1OWd2VkpDd3dlYldMNldYZEpYY
  "KeyCertificateChain":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThtZ0F3SUJBZ0lSQUtlN2piaHFKZjJPd3FGUWI5c3VuQ
}
```

キーのタグ付け

AWS Payment Cryptography では、キー の作成時に AWS Payment Cryptography キーにタグを追加し、削除保留中でない限り、既存のキーにタグを付けるかタグを解除できます。 ???タグはオプションですが、非常に便利です。

ベストプラクティス、タグ付け戦略、タグの形式と構文など、タグに関する一般的な情報については、「」の「 AWS リソースのタグ付け」を参照してくださいAmazon Web Services 全般のリファレンス。

トピック

- AWS Payment Cryptography のタグについて
- コンソールでキータグを表示する
- API オペレーションでキータグを管理する
- タグへのアクセスを制御する
- タグを使用してキーへのアクセスを制御する

AWS Payment Cryptography のタグについて

タグは、 AWS リソースに割り当てる (または割り当てることができる) AWS オプションのメタデータラベルです。各タグは、 タグキーとタグ値で構成され、どちらも大文字と小文字が区別される文字列です。タグ値には、空の (null) 文字列を指定できます。リソースの各タグには異なるタグキーが必要ですが、同じタグを複数の AWS リソースに追加できます。各リソースには、最大 50 個のユーザーが作成したタグを含めることができます。

タグキーまたはタグ値には、機密情報や重要情報を含めないでください。タグには AWS のサービス、請求を含む多くの がアクセスできます。

AWS Payment Cryptography では、キー の作成時にキーにタグを追加し、削除保留中でない限り、既存のキーにタグを付けるかタグを解除できます。エイリアスにタグを付けることはできません。タグはオプションですが、非常に便利です。

例えば、Alpha プロジェクトに使用するすべての AWS Payment Cryptography キーと Amazon S3 バケットに"Project"="Alpha"タグを追加できます。もう 1 つの例は、特定の銀行識別番号 (BIN) に関連付けられているすべてのキーに "BIN"="20130622" タグを追加することです。

```
[
  {
    "Key": "Project",
    "Value": "Alpha"
  },
  {
    "Key": "BIN",
    "Value": "20130622"
  }
]
```

形式や構文などのタグに関する一般的な情報については、「」の「 AWS リソースのタグ付け」を参照してくださいAmazon Web Services 全般のリファレンス。

タグは、以下のことに役立ちます。

- AWS リソースを特定して整理します。多くの AWS のサービスではタグ付けがサポートされているため、異なる サービスのリソースに同じタグを割り当てて、リソースが関連していることを示すことができます。例えば、 AWS Payment Cryptography キーと Amazon Elastic Block Store (Amazon EBS) ボリュームまたは AWS Secrets Manager シークレットに同じタグを割り当てることができます。タグを使用して、オートメーションのためにキーを識別することもできます。
- AWS コストを追跡します。 AWS リソースにタグを追加すると、 は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。この機能を使用して、プロジェクト、アプリケーション、またはコストセンターの AWS Payment Cryptography コストを追跡できます。

  タグを使用したコスト配分の詳細については、「AWS Billing ユーザーガイド」の「コスト配分タグの使用」を参照してください。タグキーとタグ値に適用されるルールの詳細については、「AWS Billing ユーザーガイド」の「ユーザー定義タグの制限」を参照してください。
- AWS リソースへのアクセスを制御します。タグに基づいてキーへのアクセスを許可および拒否することは、属性ベースのアクセスコントロール (ABAC) の AWS Payment Cryptography サポートの一部です。タグに基づく AWS へのアクセス制御の詳細については、「AWS Payment Cryptography タグに基づく承認」を参照してください。タグを使用してリソースへのアクセスを

制御する方法の詳細については、IAM ユーザーガイドの AWS「リソースタグを使用した AWS リソースへのアクセスの制御」を参照してください。

AWS Payment Cryptography は、 TagResource、 UntagResource、または ListTagsForResource オペレーションを使用するときに、 AWS CloudTrail ログにエントリを書き込みます。

コンソールでキータグを表示する

コンソールでタグを表示するには、キーを含む IAM ポリシーのキーに対するタグ付け許可が必要です。コンソールでキーを表示するための許可に加えて、これらの許可が必要です。

API オペレーションでキータグを管理する

AWS Payment Cryptography API を使用して、管理するキーのタグを追加、削除、一覧表示できます。以下の例では AWS Command Line Interface (AWS CLI) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。にタグを付けることはできません AWS マネージドキー。

キーのタグを追加、編集、表示、削除するには、所定の許可が必要です。詳細については、「タグへのアクセスを制御する」を参照してください。

トピック

- CreateKey: 新しいキーにタグを追加する
- TagResource: キーのタグを追加または変更する
- ListResourceタグ: キーのタグを取得する
- UntagResource: キーからタグを削除する

CreateKey: 新しいキーにタグを追加する

キーを作成するときのみタグを追加できます。タグを指定するには、 CreateKeyオペレーションの Tagsパラメータを使用します。

キーの作成時にタグを追加するには、発信者が IAM ポリシーで payment-cryptography:TagResource 許可を取得する必要があります。少なくとも、この許可はアカウントとリージョン内のすべてのキーを対象にする必要があります。詳細については、「タグへのアクセスを制御する」を参照してください。

CreateKey の Tags パラメータ値は、大文字と小文字を区別するタグキーとタグ値のペアのコレクションです。キーのそれぞれのタグは、異なるタグ名を持つ必要があります。タグ値は、NULL または空の文字列にすることができます。

例えば、次の AWS CLI コマンドは、 Project:Alpha タグを使用して対称暗号化キーを作成します。複数のキーと値のペアを指定する場合は、スペースを使用して各ペアを区切ります。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY, \
       KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY, \
       KeyModesOfUse='{Generate=true,Verify=true}' \
       --tags '[{"Key":"Project","Value":"Alpha"},{"Key":"BIN","Value":"123456"}]'
```

このコマンドが成功すると、新しいキーに関する情報を含む Key オブジェクトが返されます。ただし、Key にはタグは含まれません。タグを取得するには、ListResourceタグオペレーションを使用します。

TagResource: キーのタグを追加または変更する

TagResource オペレーションは、キーに 1 つ以上のタグを追加します。このオペレーションを使用して、別の AWS アカウントのタグを追加または編集することはできません。

タグを追加するには、新しいタグキーとタグ値を指定します。タグを編集するには、既存のタグキーと新しいタグ値を指定します。キーのそれぞれのタグは、異なるタグキーを持つ必要があります。タグ値は、NULL または空の文字列にすることができます。

例えば、次のコマンドでは、サンプルのキーに **UseCase** タグおよび **BIN** タグを追加します。

```
$ aws payment-cryptography tag-resource --resource-arn arn:aws:payment-
cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h --tags
 '[{"Key":"UseCase","Value":"Acquiring"},{"Key":"BIN","Value":"123456"}]'
```

このコマンドが成功した場合、出力を返しません。キーのタグを表示するには、ListResourceタグオペレーションを使用します。

TagResource を使用して、既存のタグのタグ値を変更することもできます。タグ値を置き換えるには、同じタグキーを異なる値に指定します。修正コマンドにリストされていないタグは変更も削除もされません。

例えば、このコマンドは Project タグの値 Alpha をからに変更します Noe。

このコマンドは、内容のない http/200 を返します。変更内容を確認するため
に、ListTagsForResource を使用する

```
$ aws payment-cryptography tag-resource --resource-arn arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h \
        --tags '[{"Key":"Project","Value":"Noe"}]'
```

ListResourceタグ: キーのタグを取得する

ListResourceタグオペレーションは、キーのタグを取得します。ResourceArn (KeyArn または KeyAlias) パラメータは必須です。このオペレーションを使用して、別の AWS アカウントのキーのタグを表示することはできません。

例えば、次のコマンドでは、サンプルのキーのタグを取得します。

```
$ aws payment-cryptography list-tags-for-resource --resource-arn arn:aws:payment-
cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h

  {
    "Tags": [
        {
            "Key": "BIN",
            "Value": "20151120"
        },
        {
            "Key": "Project",
            "Value": "Production"
        }
    ]
}
```

UntagResource: キーからタグを削除する

UntagResource オペレーションはキーからタグを削除します。削除するタグを識別するには、タグキーを指定します。このオペレーションを使用して、別の AWS アカウントのキーからタグを削除することはできません。

成功すると、 UntagResource オペレーションは出力を返しません。また、指定したタグキーがキーで見つからない場合、例外をスローしたり、レスポンスを返したりすることもありません。オペレーションが機能したことを確認するには、ListResourceタグオペレーションを使用します。

例えば、このコマンドでは、指定したキーから **Purpose** タグとその値を削除します。

```
$ aws payment-cryptography untag-resource \
        --resource-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h --tag-keys Project
```

タグへのアクセスを制御する

API を使用して、タグを追加、表示、削除するには、 コンソールまたは API を使用して、プリンシパルがIAM ポリシーでタグ付けされている必要があります。

タグ AWS のグローバル条件キーを使用して、これらのアクセス許可を制限することもできます。AWS Payment Cryptography では、これらの条件により、 TagResourceや などのタグ付けオペレーションへのアクセスを制御できますUntagResource。

サンプルポリシーおよび詳細については、「IAM ユーザーガイド」の「タグキーに基づいたアクセス制御」を参照してください。

タグを作成および管理するためのアクセス許可は、次のように機能します。

payment-cryptography:TagResource

プリンシパルにタグの追加または編集を許可します。キーの作成中にタグを追加するには、プリンシパルが特定のキーに制限されない IAM ポリシーでアクセス許可を持っている必要があります。

payment-cryptography:ListTagsForResource

プリンシパルがキーのタグを表示できるようにします。

payment-cryptography:UntagResource

プリンシパルがキーからタグを削除できるようにします。

ポリシーのタグ付け許可

キーポリシーまたは IAM ポリシーでタグ付け許可を付与できます。例えば、次のキーポリシーの例では、選択したユーザーにキーに対するタグ付け許可が付与されます。これにより、サンプルの管理者ロールまたはデベロッパーロールを引き受けることができるすべてのユーザーにタグを表示する許可が付与されます。

```
{
```

```
  "Version": "2012-10-17",
  "Id": "example-key-policy",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
      "Action": "payment-cryptography:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow all tagging permissions",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/LeadAdmin",
        "arn:aws:iam::111122223333:user/SupportLead"
      ]},
      "Action": [
          "payment-cryptography:TagResource",
          "payment-cryptography:ListTagsForResource",
          "payment-cryptography:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow roles to view tags",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:role/Administrator",
        "arn:aws:iam::111122223333:role/Developer"
      ]},
      "Action": "payment-cryptography:ListResourceTags",
      "Resource": "*"
    }
  ]
}
```

プリンシパルに複数のキーに対するタグ付け許可を付与するには、IAM ポリシーを使用します。このポリシーを有効にするには、各キーのキーポリシーで、アカウントが IAM ポリシーを使用してキーへのアクセスを制御することを許可する必要があります。

例えば、次の IAM ポリシーではプリンシパルがキーを作成することを許可します。指定したアカウントのすべてのキーでタグを作成および管理することもできます。この組み合わせにより、プリンシ

パルは CreateKeyオペレーションの tags パラメータを使用して、キーの作成中にキーにタグを追加できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKeys",
      "Effect": "Allow",
      "Action": "payment-cryptography:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyTags",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:TagResource",
        "payment-cryptography:UntagResource",
        "payment-cryptography:ListTagsForResource"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
    }
  ]
}
```

タグ付け許可を制限する

ポリシー条件を使用して、タグ付け許可を制限できます。次のポリシー条件を payment-cryptography:TagResource および payment-cryptography:UntagResource 許可に適用できます。例えば、aws:RequestTag/*tag-key* 条件を使用して、プリンシパルが特定のタグのみを追加できるようにするか、プリンシパルが特定のタグキーを持つタグを追加しないように許可できます。

- aws:RequestTag
- aws:ResourceTag/tag-key (IAM ポリシーのみ)
- aws:TagKeys

ベストプラクティスとして、タグを使用してキーへのアクセスを制御する場合は、aws:RequestTag/*tag-key*またはaws:TagKeys条件キーを使用して、許可するタグ (またはタグキー) を決定します。

例えば、次の IAM ポリシーは前述のものと似ています。ただしこのポリシーでは、プリンシパルはタグ (TagResource) の作成とタグ UntagResource の削除を、Project タグキーを持つタグに対してのみ実行できます。

TagResource および UntagResourceリクエストには複数のタグを含めることができるため、aws:TagKeys 条件で ForAllValuesまたは ForAnyValue集合演算子を指定する必要があります。ForAnyValue 演算子では、リクエスト内のタグキー 1 つ以上が、ポリシーのタグキーの 1 つと一致する必要があります。ForAllValues 演算子では、リクエスト内のタグキーすべてが、ポリシーのタグキーの 1 つと一致する必要があります。ForAllValues 演算子trueは、リクエストにタグがない場合も を返しますが、タグが指定されていない場合は TagResource と UntagResource が失敗します。集合演算子の詳細については、「IAM ユーザーガイド」の「複数のキーと値の使用」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKey",
      "Effect": "Allow",
      "Action": "payment-cryptography:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyViewAllTags",
      "Effect": "Allow",
      "Action": "payment-cryptography:ListResourceTags",
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
    },
    {
      "Sid": "IAMPolicyManageTags",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:TagResource",
        "payment-cryptography:UntagResource"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
      "Condition": {
          "ForAllValues:StringEquals": {"aws:TagKeys": "Project"}
      }
    }
  ]
```

```
}
```

タグを使用してキーへのアクセスを制御する

AWS Payment Cryptography へのアクセスは、キーのタグに基づいて制御できます。例えば、プリンシパルが特定のタグを持つキーのみを有効または無効にすることを許可する IAM ポリシーを書き込むことができます。または、IAM ポリシーを使用して、キーに特定のタグがない限り、プリンシパルが暗号化オペレーションでキーを使用しないようにすることもできます。

この機能は、属性ベースのアクセスコントロール (ABAC) の AWS Payment Cryptography サポートの一部です。タグを使用してリソースへのアクセス AWS を制御する方法については、IAM ユーザーガイドの「 の ABAC とは AWS」および「リソースタグを使用した AWS リソースへのアクセスの制御」を参照してください。

> **Note**

AWS Payment Cryptography は aws:ResourceTag/tag-key グローバル条件コンテキストキーをサポートしています。これにより、キーのタグに基づいてキーへのアクセスを制御できます。複数のキーに同じタグを付けることができるため、この機能を使用すると、キーの選択したセットにアクセス許可を適用できます。それらのタグを変更することで、セット内のキーを簡単に変更することもできます。

AWS Payment Cryptography では、 `aws:ResourceTag/`*`tag-key`* 条件キーは IAM ポリシーでのみサポートされています。1 つのキーのみに適用されるキーポリシー、または や オペレーションなど、特定のキーを使用しないListKeysListAliasesオペレーションではサポートされていません。

タグを使用してアクセスを制御すると、シンプルでスケーラブルかつ柔軟な方法でアクセス許可を管理できます。ただし、適切に設計および管理されていない場合、キーへのアクセスを誤って許可または拒否する可能性があります。タグを使用してアクセスを制御する場合は、次の方法を検討します。

- タグを使用して、最小特権アクセスのベストプラクティスを強化します。IAM プリンシパルで、使用または管理する必要があるキーのみに対して、必要なアクセス許可のみを付与します。例えば、タグを使用して、プロジェクトに使用するキーにラベルを付けます。次に、プロジェクトタグでキーのみを使用する許可をプロジェクトチームに付与します。
- プリンシパルにタグを追加、編集、削除できる `payment-cryptography:TagResource` および `payment-cryptography:UntagResource` 許可を付与する際は注意してください。タグを使用

してキーへのアクセスを制御する場合、タグを変更すると、使用許可のないキーに対する使用許可をプリンシパルに付与してしまう可能性があります。他のプリンシパルがジョブを実行するために必要なキーへのアクセスを拒否することもできます。キーポリシーを変更したり、許可を作成したりする許可を持たないキー管理者も、タグを管理する許可があれば、キーへのアクセスを制御できます。

可能な限り、ポリシー条件 (`aws:RequestTag/`*`tag-key`*または`aws:TagKeys`など) を使用して、特定のタグパターンまたは特定のキーのタグパターンにプリンシパルのタグ付け許可を制限します。

- 現在タグ付けおよびタグ付け解除のアクセス許可 AWS アカウント を持っている のプリンシパルを確認し、必要に応じて調整します。IAM ポリシーでは、すべてのキーに対してタグ付けおよびタグ解除を許可する場合があります。例えば、Admin マネージドポリシーは、プリンシパルがすべてのキーで、タグ付け、タグ解除、タグの一覧表示を行うことを許可します。
- タグに依存するポリシーを設定する前に、 のキーのタグを確認してください AWS アカウント。含めるタグにのみポリシーが適用されることを確認します。CloudTrail ログと CloudWatch アラームを使用して、キーへのアクセスに影響する可能性のある変更にタグを付けるよう警告します。
- タグベースのポリシー条件では、パターンマッチングを使用します。タグの特定のインスタンスには関連付けられません。タグベースの条件キーを使用するポリシーは、パターンに一致するすべての新規および既存のタグに影響します。ポリシー条件に一致するタグを削除して再作成すると、古いタグの場合と同様に、新しいタグに条件が適用されます。

例えば、次の IAM ポリシーの例を考えてみます。これにより、プリンシパルは、アカウント内の米国東部 (バージニア北部) リージョンで `"Project"="Alpha"` タグ付きのキーに対してのみ復号化オペレーションを呼び出すことができます。このポリシーは、サンプルの Alpha プロジェクトでロールにアタッチできます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:DecryptData"
      ],
      "Resource": "arn:aws::us-east-1:111122223333:key/*",
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceTag/Project": "Alpha"
        }
      }
    }
  ]
}
```

次の IAM ポリシーの例では、プリンシパルが、特定の暗号化オペレーションのために、アカウントで任意のキーを使用することを許可します。ただし、プリンシパルが "Type"="Reserved" タグのある、または "Type" タグのないキーにこれらの暗号化オペレーションを使用することを禁止します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAllowCryptographicOperations",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:EncryptData",
        "payment-cryptography:DecryptData",
        "payment-cryptography:ReEncrypt*"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
    },
    {
      "Sid": "IAMDenyOnTag",
      "Effect": "Deny",
      "Action": [
        "payment-cryptography:EncryptData",
        "payment-cryptography:DecryptData",
        "payment-cryptography:ReEncrypt*"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Type": "Reserved"
        }
      }
    },
    {
      "Sid": "IAMDenyNoTag",
      "Effect": "Deny",
```

```
      "Action": [
        "payment-cryptography:EncryptData",
        "payment-cryptography:DecryptData",
        "payment-cryptography:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "Null": {
          "aws:ResourceTag/Type": "true"
        }
      }
    }
  ]
}
```

AWS Payment Cryptography キーのキー属性について

適切なキー管理の原則は、キーの範囲は適切に設定され、許可されたオペレーションにのみ使用できるということです。そのため、特定のキーは特定のキー使用モードでしか作成できない場合があります。可能な限り、TR-31 で定義されている使用可能な使用モードと一致するようにしてください。

AWS Payment Cryptography では無効なキーを作成できませんが、ここでは便利な有効な組み合わせが提供されています。

対称キー

- TR31_B0_BASE_DERIVATION_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: {= true },{ NoRestrictions = true } DeriveKey
- TR31_C0_CARD_VERIFICATION_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Generate = true } 、{ Verify = true } 、{ Generate = true、Verify= true } 、{ NoRestrictions = true }
- TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } 、{ Encrypt = true, Wrap = true } 、{ Decrypt = true, Unwrap = true } 、{ NoRestrictions = true }

- TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: {= true }, { NoRestrictions = true } DeriveKey
- TR31_E1_EMV_MKEY_CONFIDENTIALITY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: {= true }, { NoRestrictions = true } DeriveKey
- TR31_E2_EMV_MKEY_INTEGRITY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: {= true }, { NoRestrictions = true } DeriveKey
- TR31_E4_EMV_MKEY_DYNAMIC_NUMBER
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: {= true }, { NoRestrictions = true } DeriveKey
- TR31_E5_EMV_MKEY_CARD_PERSONALIZATION
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: {= true }, { NoRestrictions = true } DeriveKey
- TR31_E6_EMV_MKEY_OTHER
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: {= true }, { NoRestrictions = true } DeriveKey
- TR31_K0_KEY_ENCRYPTION_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } 、{ Encrypt = true, Wrap = true } 、{ Decrypt = true, Unwrap = true } 、{ NoRestrictions = true }
- TR31_K1_KEY_BLOCK_PROTECTION_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } 、{ Encrypt = true, Wrap = true } 、{ Decrypt = true, Unwrap = true } 、{ NoRestrictions = true }
- TR31_M1_ISO_9797_1_MAC_KEY
  - 許可されるキーアルゴリズム：TDES_2KEY、TDES_3KEY

  - 使用できるキーモードの組み合わせ: { Generate = true } 、{ Verify = true } 、{ Generate = true、Verify= true } 、{ NoRestrictions = true }
- TR31_M3_ISO_9797_3_MAC_KEY
  - 許可されるキーアルゴリズム： TDES_2KEY、TDES_3KEY
  - 使用できるキーモードの組み合わせ: { Generate = true } 、{ Verify = true } 、{ Generate = true、Verify= true } 、{ NoRestrictions = true }
- TR31_M6_ISO_9797_5_CMAC_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Generate = true } 、{ Verify = true } 、{ Generate = true、Verify= true } 、{ NoRestrictions = true }
- TR31_M7_HMAC_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Generate = true } 、{ Verify = true } 、{ Generate = true、Verify= true } 、{ NoRestrictions = true }
- TR31_P0_PIN_ENCRYPTION_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } ,{ Encrypt = true, Wrap = true } ,{ Decrypt = true, Unwrap = true } ,{ NoRestrictions = true }
- TR31_V1_IBM3624_PIN_VERIFICATION_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Generate = true } 、{ Verify = true } 、{ Generate = true、Verify= true } 、{ NoRestrictions = true }
- TR31_V2_VISA_PIN_VERIFICATION_KEY
  - 使用可能なキーアルゴリズム: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 使用できるキーモードの組み合わせ: { Generate = true } 、{ Verify = true } 、{ Generate = true、Verify= true } 、{ NoRestrictions = true }

非対称キー

- TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION
  - 許可されているキーアルゴリズム: RSA_2048 ,RSA_3072 ,RSA_4096

  - 使用可能なキーモードの組み合わせ: { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } ,{ Encrypt = true, Wrap = true } ,{ Decrypt = true, Unwrap = true }
  - 注：{ Encrypt = true, Wrap = true } は、データの暗号化またはキーのラップを目的としたパブリックキーをインポートする場合に唯一の有効なオプションです。
- TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE
  - 許可されているキーアルゴリズム: RSA_2048 ,RSA_3072 ,RSA_4096
  - 使用できるキーモードの組み合わせ: { Sign = true } 、{ Verify = true }
  - 注：{ Verify = true } は、ルート証明書、中間証明書、TR-34 の署名証明書など、署名用のキーをインポートするときの唯一の有効なオプションです。

データオペレーション

AWS Payment Cryptography 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。さまざまなオペレーションが、暗号化、ハッシュ、CVV2 生成などのドメイン固有のアルゴリズムなど、さまざまなタイプのアクティビティを実行します。

暗号化されたデータは、対応する復号キー (暗号化タイプによって対称キーまたは秘密キー) がないと復号化できません。ハッシュやドメイン固有のアルゴリズムも同様に、対称キーまたは公開キーがないと検証できません。

特定のオペレーションに有効なキータイプについては、「暗号オペレーションに有効なキー」セクションを参照してください。

> **Note**
>
> 本番稼働以外の環境では、テストデータを使用することをお勧めします。本番稼働以外の環境でプロダクションキーとデータ (PAN、BDK ID など) を使用すると、PCI DSS や PCI P2PE などのコンプライアンス範囲に影響する可能性があります。

トピック

データの暗号化、復号化、再暗号化

暗号化と復号化の方法を使用して、TDES、AES、RSAなどのさまざまな対称および非対称技術を使用してデータを暗号化または復号化できます。これらのメソッドは、DUKPT および EMV 技術を使用して生成されたキーもサポートします。基になるデータを公開せずに新しいキーでデータを保護したいユースケースでは、 ReEncrypt このコマンドを使用することもできます。

> **Note**
>
> 暗号化/復号化機能を使用する場合、すべての入力は HexBinary であると想定されます。たとえば、値 1 は 31 (16進数) と入力され、小文字の t は 74 (16進数) と表されます。出力もすべて HexBinary 形式になります。

使用可能なすべてのオプションの詳細については、『暗号化、復号化、再暗号化用 API ガイド』を参照してください。

トピック

- [データの暗号化]
- データの復号化

[データの暗号化]

`Encrypt Data`この API は、DUKPT や EMV の派生キーだけでなく、対称データ暗号化キーと非対称データ暗号化キーを使用してデータを暗号化するのにも使用されます。TDES や RSA、AES など、さまざまなアルゴリズムとバリエーションをサポートしています。

主な入力は、データの暗号化に使用される暗号化キー、暗号化対象の HexBinary 形式のプレーンテキストデータ、TDES などのブロック暗号の初期化ベクトルやモードなどの暗号化属性です。プレーンテキストデータは、は 8 バイト、は 16 バイトTDES、の場合はキーの長さの倍数である必要があります。AES RSA入力データがこれらの要件を満たさない場合は、対称キー入力 (TDES、AES、DUKPT、EMV) をパディングする必要があります。次の表は、各キータイプのプレーンテキストの最大長と、RSA キーに定義するパディングタイプを示しています。`EncryptionAttributes`

パディングタイプ	RSA_2048	RSA_3072	RSA_4096
`OAEP_SHA1`	428	684	940
`OAEP_SHA256`	380	636	892
`OAEP_SHA512`	252	508	764
`PKCS1`	488	744	1,000

パディングタイプ	RSA_2048	RSA_3072	RSA_4096
None	488	744	1,000

主な出力には、HexBinary 形式の暗号文として暗号化されたデータと、暗号化キーのチェックサム値が含まれます。使用可能なすべてのオプションの詳細については、『Encrypt 用 API ガイド』を参照してください。

例

- AES 対称キーを使用したデータの暗号化
- DUKPT キーを使用したデータの暗号化
- EMV 由来の対称鍵を使用してデータを暗号化します。
- RSA キーを使用したセッションデータの暗号化

AES 対称キーを使用したデータの暗号化

> **Note**
>
> どの例も、関連するキーがすでに存在することを前提としています。キーはオペレーションを使用して作成することも、CreateKeyオペレーションを使用してインポートすることもできます。ImportKey

Example

この例では、オペレーションを使用して作成された、CreateKeyまたはオペレーションを使用してインポートされた対称キーを使用して、プレーンテキストデータを暗号化します。ImportKeyこの操作では、キーがに設定され、 KeyModesOfUse `Encrypt`に設定されている必要があります。KeyUsage `TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY`その他のオプションについては、「暗号化オペレーション用キー」を参照してください。

```
$ aws payment-cryptography-data encrypt-data --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --plain-text
 31323334313233343132333431323334 --encryption-attributes 'Symmetric={Mode=CBC}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

DUKPT キーを使用したデータの暗号化

Example

この例では、DUKPT キーを使用してプレーンテキストデータを暗号化します。 AWS 支払い暗号化は DUKPT キーをサポートします。TDES AESこの操作では、DeriveKeyキーがに設定され KeyUsage 、 KeyModesOfUse がに設定されている必要があります。TR31_B0_BASE_DERIVATION_KEYその他のオプションについては、「暗号化オペレーション用キー」を参照してください。

```
$ aws payment-cryptography-data encrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --plain-text 31323334313233343132333431323334 --encryption-attributes
 'Dukpt={KeySerialNumber=FFFF9876543210E00001}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

EMV 由来の対称鍵を使用してデータを暗号化します。

Example

この例では、すでに作成されている EMV 由来の対称鍵を使用してクリアテキストデータを暗号化します。このようなコマンドを使用して EMV カードにデータを送信できます。この操作では、 KeyModesOfUse `Derive`キーがに設定され、 KeyUsage `TR31_E1_EMV_MKEY_CONFIDENTIALITY`または`TR31_E6_EMV_MKEY_OTHER`に設定されている必要があります。詳細については、「暗号化操作用のキー」を参照してください。

```
$ aws payment-cryptography-data encrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --plain-text 33612AB9D6929C3A828EB6030082B2BD --encryption-attributes
 'Emv={MajorKeyDerivationMode=EMV_OPTION_A,PanSequenceNumber=27,PrimaryAccountNumber=1000000000
 InitializationVector=1500000000000999,Mode=CBC}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

RSA キーを使用したセッションデータの暗号化

Example

この例では、操作を使用してインポートされた RSA 公開鍵を使用してプレーンテキストデータを暗号化します。ImportKeyこの操作では、キーがに設定され、 KeyModesOfUse `Encrypt`がに設定されている必要があります。 KeyUsage `TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION`その他のオプションについては、「暗号化オペレーション用キー」を参照してください。

PKCS #7 または現在サポートされているないその他のパディングスキームについては、サービスを呼び出す前に申請し、パディングインジケータ「'Asymmetric={}'」を省略してパディングなしを選択してください。

```
$ aws payment-cryptography-data encrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/thfezpmsalcfwmsg
 --plain-text 31323334313233343132333431323334 --encryption-attributes
 'Asymmetric={PaddingType=OAEP_SHA256}'
```

```
{
 "CipherText":
 "12DF6A2F64CC566D124900D68E8AFEAA794CA819876E258564D525001D00AC93047A83FB13 \
 E73F06329A100704FA484A15A49F06A7A2E55A241D276491AA91F6D2D8590C60CDE57A642BC64A897F4832A3930
 \
 0FAEC7981102CA0F7370BFBF757F271EF0BB2516007AB111060A9633D1736A9158042D30C5AE11F8C5473EC70F067
 \
 72590DEA1638E2B41FAE6FB1662258596072B13F8E2F62F5D9FAF92C12BB70F42F2ECDCF56AADF0E311D4118FE3591
 \
 FB672998CCE9D00FFFE05D2CD154E3120C5443C8CF9131C7A6A6C05F5723B8F5C07A4003A5A6173E1B425E2B5E42AD
 \
 7A2966734309387C9938B029AFB20828ACFC6D00CD1539234A4A8D9B94CDD4F23A",
 "KeyArn": "arn:aws:payment-cryptography:us-east-1:529027455495:key/5dza7xqd6soanjtb",
 "KeyCheckValue": "FF9DE9CE"
}
```

データの復号化

`Decrypt Data`API は、DUKPT や EMV の派生キーだけでなく、対称データ暗号化キーと非対称データ暗号化キーを使用してデータを復号化するために使用されます。TDES や RSA、AES など、さまざまなアルゴリズムとバリエーションをサポートしています。

主な入力は、データの復号化に使用される復号化キー、復号化対象の HexBinary 形式の暗号文データ、および初期化ベクトルやブロック暗号モードなどの復号化属性です。主な出力には、HexBinary 形式のプレーンテキストとして復号化されたデータと、復号キーのチェックサム値が含まれます。使用可能なすべてのオプションの詳細については、「復号用 API ガイド」を参照してください。

例

- AES 対称キーを使用したデータの復号化
- DUKPT キーを使用したデータの復号化
- EMV から派生した対称鍵を使用してデータを復号化します。
- RSA キーを使用してデータを復号化します。

AES 対称キーを使用したデータの復号化

Example

この例では、対称鍵を使用して暗号文データを復号化します。AESこの例では鍵を示していますが、ともサポートされています。TDES_2KEY TDES_3KEYこの操作では、Decryptキーがに設定され、 KeyModesOfUse KeyUsage がに設定されている必要がありますTR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY。その他のオプションについては、「暗号化オペレーション用キー」を参照してください。

```
$ aws payment-cryptography-data decrypt-data --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --cipher-text
 33612AB9D6929C3A828EB6030082B2BD --decryption-attributes 'Symmetric={Mode=CBC}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "PlainText": "31323334313233343132333431323334"
}
```

DUKPT キーを使用したデータの復号化

> **Note**
>
> DUKPT で復号データを P2PE トランザクションに使用すると、PCI DSS の範囲を決定する際に考慮する必要があるクレジットカード PAN やその他のカード会員データがアプリケーションに返される可能性があります。

Example

この例では、オペレーションを使用して作成された、またはオペレーションを使用してインポートされた DUKPT キーを使用して暗号文データを復号化します。CreateKeyImportKeyこの操作では、キーがに設定され、がに設定されている必要があります。 KeyModesOfUse DeriveKey KeyUsage

TR31_B0_BASE_DERIVATION_KEYその他のオプションについては、「暗号化オペレーション用キー」を参照してください。TDESアルゴリズムを使用する場合DUKPT、暗号文データ長は 16 バイトの倍数でなければなりません。AESアルゴリズムの場合、暗号文データ長は 32 バイトの倍数でなければなりません。

```
$ aws payment-cryptography-data decrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --cipher-text 33612AB9D6929C3A828EB6030082B2BD --decryption-attributes
 'Dukpt={KeySerialNumber=FFFF9876543210E00001}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "PlainText": "31323334313233343132333431323334"
}
```

EMV から派生した対称鍵を使用してデータを復号化します。

Example

この例では、操作を使用して作成された、または操作を使用してインポートされた EMV 由来の対称鍵を使用して暗号文データを復号化します。CreateKeyImportKeyこの操作では、キーがまたはに設定され、かに設定されている必要があります。 KeyModesOfUse Derive KeyUsage TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER詳細については、「暗号化操作用のキー」を参照してください。

```
$ aws payment-cryptography-data decrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --cipher-text 33612AB9D6929C3A828EB6030082B2BD --decryption-attributes
 'Emv={MajorKeyDerivationMode=EMV_OPTION_A,PanSequenceNumber=27,PrimaryAccountNumber=1000000000
 InitializationVector=1500000000000999,Mode=CBC}'
```

```
{
```

```
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi",
"KeyCheckValue": "71D7AE",
"PlainText": "313233343132333431323334313233334"
}
```

RSA キーを使用してデータを復号化します。

Example

この例では、オペレーションを使用して作成された RSA key pair を使用して暗号文データを復号化します。CreateKeyこの操作では、キーが有効に設定され、 KeyModesOfUse に設定されている必要があります。Decrypt KeyUsage TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTIONその他のオプションについては、「暗号化オペレーション用キー」を参照してください。

PKCS #7 または現在サポートされているないその他のパディングスキームについては、パディングインジケータ 「'Asymmetric= {}'」 を省略してパディングなしを選択し、サービスを呼び出したらパディングを削除してください。

```
$ aws payment-cryptography-data decrypt-data \
        --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5dza7xqd6soanjtb --cipher-text
 8F4C1CAFE7A5DEF9A40BEDE7F2A264635C... \
        --decryption-attributes 'Asymmetric={PaddingType=OAEP_SHA256}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-
east-1:529027455495:key/5dza7xqd6soanjtb",
    "KeyCheckValue": "FF9DE9CE",
    "PlainText": "313233343132333431323334313233334"
}
```

カードデータの生成と検証

カードデータにカードデータから派生したデータ (CVV、CVV2、CVC、DCVV など) が組み込まれていることを生成して検証します。

トピック

- カードデータの生成

- カードデータの検証

カードデータの生成

Generate Card Data API は CVV、CVV2、ダイナミック CVV2 などのアルゴリズムを使用してカードデータを生成するために使用されます。このコマンドで使用できるキーについては、「暗号化オペレーションに有効なキー」セクションを参照してください。

CVV、CVV2、iCVV、CAVV V8 などの多くの暗号化値は同じ暗号化アルゴリズムを使用しますが、入力値は異なります。例えば、CardVerificationValue1 には ServiceCode、、カード番号、有効期限の入力があります。CardVerificationValue2 にはこれらの入力が 2 つしかありませんが、CVV2/CVC2 の場合、 ServiceCode は 000 に固定されているためです。同様に、iCVV の場合、ServiceCode は 999 に固定されます。一部のアルゴリズムでは、CAVV V8 などの既存のフィールドを再利用する場合があります。その場合は、プロバイダーマニュアルを参照して正しい入力値を確認する必要があります。

> **Note**
>
> 正しい結果を生成するには、有効期限を同じ形式 (MMY と YYMM など) で入力する必要があります。

CVV2 の生成

Example

この例では、 PANとカードの有効期限を入力して、特定の PAN の CVV2 を生成します。これは、カード認証キーが生成済みであることを前提としています。

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue2={CardExpiryDate=0123}
```

```
  {
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
```

```
    "KeyCheckValue": "CADDA1",
    "ValidationData": "801"
}
```

iCVV の生成

Example

この例では、入力が 、サービスコードが 999PAN、カードの有効期限が切れた特定の PAN の iCVV を生成します。これは、カード認証キーが生成済みであることを前提としています。

使用可能なすべてのパラメータについては、 API リファレンスガイドのCardVerificationValue1」を参照してください。

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "CADDA1",
    "ValidationData": "801"
}
```

カードデータの検証

Verify Card Data は、DISCOVER_DYNAMIC_CARD_VERIFICATION_CODE などの暗号化原理に依存する支払いアルゴリズムを使用して作成されたデータを検証するために使用されます。

入力値は通常、インバウンドトランザクションの一部として発行者またはサポートされているプラットフォームパートナーに提供されます。ARQC 暗号文 (EMV チップカードに使用) を検証するには、「ARQC の検証」を参照してください。

詳細については、 API ガイドVerifyCardValidationDataの「」を参照してください。

値が検証されると、API は http/200 を返します。値が検証されないと、API は http/400 を返します。

CVV2 の検証

Example

この例では、特定の PAN の CVV/CVV2 を検証します。CVV2 は通常、カード所有者またはユーザーがトランザクション中に検証のために提供します。入力内容を検証するために、ランタイム時に検証に使用するキー (CVK)、PAN、カードの有効期限、CVV2 の入力値が提供されます。カードの有効期限の形式は、初期値の生成に使用した形式と一致する必要があります。

使用可能なすべてのパラメータについては、 API リファレンスガイドのCardVerificationValue2」を参照してください。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue2={CardExpiryDate=0123} --validation-data 801
```

```
  {
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
      "KeyCheckValue": "CADDA1"
  }
```

iCVV の検証

Example

この例では、検証に使用するキー (CVK)、、999 のサービスコードPAN、カードの有効期限、検証するトランザクションによって提供される iCVV の入力を含む特定の PAN の iCVV を検証します。

iCVV はユーザーが入力した値 (CVV2 など) ではなく、EMV カードに埋め込まれています。提供されたときに常に検証する必要があるかどうかを考慮する必要があります。

使用可能なすべてのパラメータについては、 API リファレンスガイドのCardVerificationValue1」を参照してください。

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
```

```
tqv5yij6wtxx64pi --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "CADDA1",
    "ValidationData": "801"
}
```

PIN データの生成、変換、検証

PIN データ関数を使用すると、ランダムなピンや PIN 検証値 (PVV) を生成したり、受信した暗号化ピンを PVV や PIN オフセットと照合して検証したりできます。

ピン変換を使用すると、PCI PIN 要件 1 で規定されているクリアテキストでピンを公開しなくても、1 つの動作キーから別のキーに PIN を変換できます。

> **Note**
>
> PIN の生成と検証は一般に発行者の機能であり、PIN 変換は一般的な取得者の機能であるため、最低特権アクセスを検討し、システムのユースケースに適したポリシーを設定することをお勧めします。

トピック

- PIN データ変換
- PIN データ生成
- PIN データ検証

PIN データ変換

PIN データ変換機能を使用すると、暗号化された PIN データを HSM から送信することなく、あるキーセットから別のキーセットに変換できます。P2PE 暗号化に使用されます。P2PE 暗号化では、作業キーは変更されるはずなのに、処理システムがデータを復号する必要も許可もされません。主な入力は、暗号化されたデータ、データの暗号化に使用される暗号化キー、入力値の生成に使用される

パラメーターです。その他の入力セットは、出力の暗号化に使用するキーや出力の作成に使用されるパラメーターなど、要求された出力パラメーターです。主な出力は、新しく暗号化されたデータセットと、その生成に使用されたパラメーターです。

> **Note**
>
> AES キータイプは ISO Format 4 ピンブロックのみをサポートしています。

トピック

- PEK から DUKPT へのピン
- DUKPT から AWK への PIN

PEK から DUKPT へのピン

Example

この例では、PIN を ISO 0 PIN ブロックを使用する PEK TDES 暗号化から DUKPT アルゴリズムを使用する AES ISO 4 ピンブロックに変換します。一般的には、支払いターミナルが ISO 4 で PIN を暗号化し、それを TDES に変換して下流処理を行うという逆の方法で行われます。

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-block
 "AC17DC148BDA645E" --incoming-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}' --incoming-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt  --outgoing-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/4pmyquwjs3yj4vwe --outgoing-translation-attributes
 IsoFormat4="{PrimaryAccountNumber=171234567890123}" --outgoing-dukpt-attributes
 KeySerialNumber="FFFF9876543210E00008"
```

```
    {
            "PinBlock": "1F4209C670E49F83E75CC72E81B787D9",
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
            "KeyCheckValue": "7CC9E2"
        }
```

DUKPT から AWK への PIN

Example

この例では、PIN を AES DUKPT で暗号化された PIN から AWK で暗号化されたピンに変換します。これは機能的には前の例と逆です。

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-
block "1F4209C670E49F83E75CC72E81B787D9" --outgoing-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}' --outgoing-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt   --incoming-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/4pmyquwjs3yj4vwe --incoming-translation-attributes
 IsoFormat4="{PrimaryAccountNumber=171234567890123}" --incoming-dukpt-attributes
 KeySerialNumber="FFFF9876543210E00008"
```

```
        {
            "PinBlock": "AC17DC148BDA645E",
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
            "KeyCheckValue": "FE23D3"
        }
```

PIN データ生成

PINデータ生成機能は、PVV やピンブロックオフセットなどの PIN 関連の値を生成するために使用されるPINデータ生成機能を使用して、トランザクションまたは認証時にユーザーが PIN を入力したことを検証します。この API では、さまざまなアルゴリズムを使用して新しいランダム PIN を生成することもできます。

PIN の Visa PVV を生成する

Example

この例では、出力が暗号化された (.) と `PIN block` (pinDataPinData.Offset) となる新しい PVV (ランダムPinBlock) ピンを生成します。pinData.Offset). 主な入力は、`PAN`、`Pin Verification Key`、`Pin Encryption Key`、`PIN block format` です。

このコマンドでは、キーがタイプ である必要がありま
すTR31_V2_VISA_PIN_VERIFICATION_KEY。

```
$ aws payment-cryptography-data generate-pin-data --generation-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 --encryption-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt
 --primary-account-number 171234567890123 --pin-block-format ISO_FORMAT_0 --generation-
attributes VisaPin={PinVerificationKeyIndex=1}
```

```
{
            "GenerationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
            "GenerationKeyCheckValue": "7F2363",
            "EncryptionKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt",
            "EncryptionKeyCheckValue": "7CC9E2",
            "EncryptedPinBlock": "AC17DC148BDA645E",
            "PinData": {
                "VerificationValue": "5507"
            }
        }
```

ピンの IBM3624 ピンオフセットを生成する

IBM 3624 PIN オフセットは、IBM メソッドとも呼ばれます。この方法では、検証データ (通常は PAN) と PIN キー (PVK) を使用して自然/中間 PIN を生成します。自然ピンは実質的に派生値であり、カード所有者レベルでピンデータを保存する必要がないため、発行者にとって決定論的であることは非常に効率的です。最も明らかなのは、このスキームはカード所有者が選択できるピンやランダムなピンを考慮していないことです。これらのタイプのピンを許可するため、オフセットアルゴリズムがスキームに追加されました。オフセットは、選択したユーザー (またはランダム) ピンと自然キーの差を表します。オフセット値は、カード発行者またはカードプロセッサによって保存されます。トランザクション時に、 AWS Payment Cryptography サービスは内部的に自然ピンを再計算し、オフセットを適用してピンを見つけます。次に、これをトランザクション認証によって提供される値と比較します。

IBM3624 にはいくつかのオプションがあります。

- `Ibm3624NaturalPin` は、自然ピンと暗号化されたピンブロックを出力します。
- `Ibm3624PinFromOffset` は、オフセットを指定して暗号化されたピンブロックを生成します。

- `Ibm3624RandomPin` はランダムなピンを生成し、次に一致するオフセットと暗号化されたピンブロックを生成します。
- `Ibm3624PinOffset` は、ユーザーが選択したピンを指定してピンオフセットを生成します。

AWS Payment Cryptography の内部では、次の手順が実行されます。

- 提供されたパンに 16 文字のパッドを付けます。<16 が指定されている場合は、指定されたパディング文字を使用して右側にパディングします。
- PIN 生成キーを使用して検証データを暗号化します。
- 小数化テーブルを使用して、暗号化されたデータを小数化します。これは、16 進数桁を 10 進数にマッピングします。インスタンス「A」は 9 に、1 は 1 にマッピングできます。
- 出力の 16 進表現から最初の 4 桁を取得します。これは自然なピンです。
- ユーザーが選択した またはランダムなピンを生成した場合は、モジュロから顧客ピンで自然ピンを減算します。結果はピンオフセットです。

例

- ピンの IBM3624 ピンオフセットを生成する

ピンの IBM3624 ピンオフセットを生成する

この例では、出力が暗号化された (.) とIBM3624オフセット値 PIN block (pinDataPinData.Offset) となる新しい (ランダムPinBlock) ピンを生成します。pinData.Offset). 入力は、PAN、検証データ (通常はパン) 、パディング文字、Pin Verification Key、、 Pin Encryption KeyですPIN block format。

このコマンドでは、ピン生成キーが タイプTR31_V1_IBM3624_PIN_VERIFICATION_KEYで、暗号化キーが タイプである必要があります TR31_P0_PIN_ENCRYPTION_KEY

Example

次の例は、ランダムピンを生成し、Ibm3624 を使用して暗号化されたピンブロックと IBM3624 オフセット値を出力する例です。 Ibm3624RandomPin

```
$ aws payment-cryptography-data generate-pin-data --generation-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2
 --encryption-key-identifier arn:aws:payment-cryptography:us-
```

```
east-2:111122223333:key/ivi5ksfsuplneuyt --primary-account-number
 171234567890123 --pin-block-format ISO_FORMAT_0 --generation-attributes
 Ibm3624RandomPin="{DecimalizationTable=9876543210654321,PinValidationDataPadCharacter=D,PinVal
```

```
{
              "GenerationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
              "GenerationKeyCheckValue": "7F2363",
              "EncryptionKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt",
              "EncryptionKeyCheckValue": "7CC9E2",
              "EncryptedPinBlock": "AC17DC148BDA645E",
              "PinData": {
                  "PinOffset": "5507"
              }
          }
```

PIN データ検証

PIN データ検証機能は PIN が正しいかどうかの検証に使用されます。この検証には通常、以前に保存した暗証番号の値と、カード所有者が POI で入力した暗証番号の値との照合が含まれます。この機能によって、いずれのソースの基になる値も表示されずに 2 つの値が照合されます。

PVV メソッドを使用して暗号化された PIN を検証する

Example

この例では、特定の PAN の PIN を検証します。PIN は通常、検証のために取引時にカード所有者またはユーザーによって提供され、ファイル上の値と比較されます (カード所有者からの入力は、ターミナルまたはその他のアップストリームプロバイダーから暗号化された値として提供されます) 。この入力を検証するために、ランタイム時には、入力 PIN (通常は `IWK`) の暗号化に使用されるキー、PAN と検証対象の値 (PVVまたは `PIN offset`) の値も提供されます。

AWS Payment Cryptography が PIN を検証できる場合、http/200 が返されます。PIN が検証されない場合、http/400 が返されます。

```
$ aws payment-cryptography-data verify-pin-data --verification-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 --encryption-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt
 --primary-account-number 171234567890123 --pin-block-format ISO_FORMAT_0  --
```

```
verification-attributes VisaPin="{PinVerificationKeyIndex=1,VerificationValue=5507}" --
encrypted-pin-block AC17DC148BDA645E
```

```
        {
            "VerificationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
            "VerificationKeyCheckValue": "7F2363",
            "EncryptionKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt",
            "EncryptionKeyCheckValue": "7CC9E2",

        }
```

以前に保存した IBM3624 ピンオフセットに対して PIN を検証する

この例では、カード発行者/プロセッサにファイルに保存されているピンオフセットと照らし合わせて、カード所有者が提供した PIN を検証します。入力は、支払いターミナル (またはカードネットワークなどの他のアップストリームプロバイダー) によって提供される暗号化された PIN の追加???と似ています。PIN が一致すると、API は http 200 を返します。ここで、出力は暗号化された PIN block (PinData.PinBlock) とIBM3624オフセット値 (pinData.Offset).

このコマンドでは、ピン生成キーが タイプTR31_V1_IBM3624_PIN_VERIFICATION_KEYで、暗号化キーが タイプである必要があります TR31_P0_PIN_ENCRYPTION_KEY

Example

```
$ aws payment-cryptography-data generate-pin-data --generation-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2
 --encryption-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt --primary-account-number
 171234567890123 --pin-block-format ISO_FORMAT_0 --generation-attributes
 Ibm3624RandomPin="{DecimalizationTable=9876543210654321,PinValidationDataPadCharacter=D,PinVal
```

```
{
"GenerationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
"GenerationKeyCheckValue": "7F2363",
```

```
"EncryptionKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
"EncryptionKeyCheckValue": "7CC9E2",
"EncryptedPinBlock": "AC17DC148BDA645E",
"PinData": {
  "PinOffset": "5507"
}
}
```

認証リクエスト (ARQC) 暗号文の検証

認証リクエスト暗号検証 API は ARQC の検証に使用されます。ARQC の生成は AWS Payment Cryptography の範囲外であり、通常、トランザクション認証時に EMV チップカード (またはモバイルウォレットなどのデジタル同等のカード) で実行されます。ARQC は各取引に固有のもので、カードの有効性を暗号的に示すとともに、取引データが現在の (予想される) 取引と完全に一致することを保証することを目的としています。

AWS Payment Cryptography には、ARQC を検証し、EMV 4.4 Book 2 で定義されているものや Visa や Mastercard で使用されるその他のスキームなど、オプションの ARPC 値を生成するためのさまざまなオプションが用意されています。使用可能なすべてのオプションの完全なリストについては、「 API ガイド」の VerifyCardValidationData 「」セクションを参照してください。

ARQC 暗号文には通常、次の入力が必要です (ただし、実装によって異なる場合があります) 。

- PAN - PrimaryAccountNumber フィールドで指定
- PAN シーケンス番号 (PSN) - PanSequenceNumber フィールドで指定
- 共通セッションキー (CSK) などのキー取得方法 - で指定 SessionKeyDerivationAttributes
- マスターキー取得モード (EMV オプション A など) - で指定 MajorKeyDerivationMode
- トランザクションデータ - 金額や日付などのさまざまなトランザクション、ターミナル、カードデータの文字列 - TransactionData フィールドで指定
- 発行者マスターキー - 個々のトランザクションを保護し、 KeyIdentifier フィールドで指定された暗号文 (AC) キーを取得するために使用されるマスターキー

トピック

- トランザクションデータの作成
- トランザクションデータパディング
- 例

トランザクションデータの作成

トランザクションデータフィールドの正確な内容 (および順序) は実装とネットワークスキームによって異なりますが、推奨される最小フィールド (および連結シーケンス) は EMV 4.4 Book 2 Section 8.1.1 - Data Selection で定義されています。最初の 3 つのフィールドが金額 (17.00)、その他の金額 (0.00)、購入国の場合、トランザクションデータは次のように始まります。

- 000000001700 - 金額 - 12 桁の小数点以下 2 桁を暗示します
- 000000000000 - その他の金額 - 12 桁の数字は 2 桁の十進法で表記されます
- 0124 - 4 桁の国コード
- アウトプット (一部) トランザクションデータ (一部) - 00000000170000000000000000124

トランザクションデータパディング

トランザクションデータは、サービスに送信する前にパディングする必要があります。ほとんどのスキームでは、ISO 9797 メソッド 2 のパディングを使用します。このパディングでは、フィールドが暗号化ブロックサイズの倍数になるまで、16 進数文字列に 16 進数 80 の後に 00 が続きます。TDES の場合は 8 バイトまたは 16 文字、AES の場合は 16 バイトまたは 32 文字です。代替方法 (方法 1) はそれほど一般的ではなく、パディング文字として 00 だけを使用します。

ISO 9797 メソッド 1: パディング

パディングなし:
00000000170000000000000008400080008000084016051700000000093800000B03011203 (74 文字または 37 バイト)

パディング付き:
00000000170000000000000008400080008000084016051700000000093800000B03011203 000000 (80 文字または 40 バイト)

ISO 9797 メソッド 2: パディング

パディングなし:ing.17000000084000800080000840160517923093800000B1F220103000000 (80 文字または 40 バイト)

パディング付き:17000000084000800080000840160517093800000B1F220103000000 80000000 (88 文字または 44 バイト)

例

VISA CVN (10)

Example

この例では、Visa CVN10 を使用して生成された ARQC の検証を行います。

AWS Payment Cryptography が ARQC を検証できる場合、http/200 が返されます。arqc が検証されない場合、http/400 レスポンスが返されます。

```
$ aws payment-cryptography-data verify-auth-request-cryptogram --auth-request-
cryptogram D791093C8A921769 \
--key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
--major-key-derivation-mode EMV_OPTION_A \
--transaction-data
 00000000170000000000000008400080008000084016051700000000093800000B03011203000000 \
--session-key-derivation-attributes='{"Visa":{"PanSequenceNumber":"01" \
,"PrimaryAccountNumber":"9137631040001422"}}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk",
  "KeyCheckValue": "08D7B4"
}
```

ビザ CVN18 とビザ CVN22

Example

この例では、Visa CVN18 または CVN22 を使用して生成された ARQC を検証します。CVN18 と CVN22 の暗号化オペレーションは同じですが、トランザクションデータに含まれるデータは異なります。CVN10 と比較すると、同じ入力でもまったく異なる暗号文が生成されます。

AWS Payment Cryptography が ARQC を検証できる場合、http/200 が返されます。arqc が検証されない場合、http/400 が返されます。

```
$ aws payment-cryptography-data verify-auth-request-cryptogram \
--auth-request-cryptogram 61EDCC708B4C97B4
--key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
```

```
--major-key-derivation-mode EMV_OPTION_A
--transaction-data
 00000000170000000000000084000800080000840160517000000000093800000B1F22010300000000000
 \
0000000000000000000000000000000000000000000800000000000000
--session-key-derivation-attributes='{"EmvCommon":
{"ApplicationTransactionCounter":"000B", \
"PanSequenceNumber":"01","PrimaryAccountNumber":"9137631040001422"}}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk",
  "KeyCheckValue": "08D7B4"
}
```

MAC の生成と検証

メッセージ認証コード (MAC) は通常、メッセージの完全性 (変更されているかどうか) を認証するために使用されます。さらに、HMAC (ハッシュベースのメッセージ認証コード)、CBC-MAC、CMAC (暗号ベースのメッセージ認証コード) などの暗号ハッシュは、暗号を利用して MAC の送信者をさらに保証します。HMAC はハッシュ関数をベースにしていますが、CMAC はブロック暗号に基づいています。

すべての MAC アルゴリズムは、暗号化ハッシュ関数と共有シークレットキーを組み合わせます。これらはメッセージとシークレットキー (キーのキーマテリアルなど) を使用して、一意のタグまたはmacを返します。メッセージの文字が 1 字でも変更されている場合、またはシークレットキーが同一ではない場合、結果として得られるタグはまったく異なるものになります。暗号化MAC は、シークレットキーをリクエストすることによって信頼性も提供するため、シークレットキーがなければ、同一の HMAC タグを生成することは不可能となります。Cryptographic MAC は対称署名と呼ばれることもあります。これらはデジタル署名のように機能しますが、署名と検証の両方に単一のキーを使用するからです。

AWS Payment Cryptography はいくつかのタイプの MAC をサポートしています。

ISO9797 アルゴリズム 1

ISO9797_ALGORITHM1の KeyUsage で示されます。

ISO9797 アルゴリズム 3 (リテールマック)

ISO9797_ALGORITHM3の KeyUsage で示されます。

ISO9797 アルゴリズム 5 (CMAC)

TR31_M6_ISO_9797_5_CMAC_KEYの KeyUsage で示されます。

HMAC

TR31_M7_HMAC_KEY including
HMAC_SHA224、HMAC_SHA256、HMAC_SHA384、HMAC_SHA512の KeyUsage で示されます。

トピック

- MAC の生成
- MAC の検証

MAC の生成

Generate MAC API は、既知のデータ値を使用して送信側と受信側間のデータ検証用の MAC (メッセージ認証コード) を生成することにより、カードの磁気ストライプからのトラックデータなど、カード関連のデータを認証するために使用されます。MAC の生成に使用されるデータには、メッセージデータ、秘密 MAC 暗号化キー、および送信用の固有の MAC 値を生成する MAC アルゴリズムが含まれます。MAC の受信側は、同じ MAC メッセージデータ、MAC 暗号化キー、およびアルゴリズムを使用して、比較とデータ認証のために別の MAC 値を再現します。メッセージの文字が 1 字でも変わっている場合、またはシークレットキーが同一ではない場合、結果として得られる MAC の値はまったく異なるものになります。API は、このオペレーション用の DUPKT MAC、HMAC、および EMV MAC 暗号化キーをサポートしています。

`message-data` の入力値は HexBinary データでなければなりません。

この例では、HMAC `HMAC_SHA256` アルゴリズムと HMAC 暗号化キーを使用してカードデータ認証用の HMAC (ハッシュベースのメッセージ認証コード) を生成します。キーは、KeyUsage が `TR31_M7_HMAC_KEY` に設定され、KeyModesOfUse が `Generate` に設定されている必要があります。MAC キーは、CreateKey を呼び出してAWS Payment Cryptography を使用して作成することも、ImportKey を呼び出してインポートすることもできます。

Example

```
$ aws payment-cryptography-data generate-mac \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6 \
```

```
    --message-data
 "3b3130383834393030313037333934313533d32343038323236303030373030303f33" \
    --generation-attributes Algorithm=HMAC_SHA256
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6,
    "KeyCheckValue": "2976E7",
    "Mac": "ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C"
}
```

MAC の検証

MAC (メッセージ認証コード) を検証するための MAC (メッセージ認証コード) の検証には MAC (MAC) API が使用されます。認証用の MAC 値を再生成するには、MAC の生成時に使用したのと同じ暗号キーを使用する必要があります。MAC 暗号化キーは、CreateKey を呼び出して AWS Payment Cryptography を使用して作成することも、ImportKey を呼び出してインポートすることもできます。API は、このオペレーション用の DUPKT MAC、HMAC、および EMV MAC 暗号化キーをサポートしています。

値が検証されると、レスポンスパラメータ `MacDataVerificationSuccessful` は `Http/200` を返し、検証されなかった場合は、`Http/400` を返すと共に、`Mac verification failed` を示すメッセージを表示します。

この例では、HMAC `HMAC_SHA256` アルゴリズムと HMAC 暗号化キーを使用してカードデータ認証用の HMAC (ハッシュベースのメッセージ認証コード) を検証します。キーは、KeyUsage が `TR31_M7_HMAC_KEY` に設定され、KeyModesOfUse が `Verify` に設定されている必要があります。

Example

```
$ aws payment-cryptography-data verify-mac \
     --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6 \
     --message-data
 "3b3430383834393030313037333934313533d32343038323236303030373030303f33" \
     --verification-attributes='Algorithm=HMAC_SHA256' \
     --mac ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6,
    "KeyCheckValue": "2976E7",
}
```

暗号化オペレーション用の検証キー

特定のキーは特定のオペレーションにのみ使用できます。また、オペレーションによってはキーを使用できるキーモードが制限される場合があります。以下の許可される組み合わせの説明をご覧ください。

> **Note**
>
> 組み合わせによっては許可されているものの、CVV コード (`generate`) を生成しても検証できないなど、使用できない状況が生じる場合があります。(`verify`)

トピック

- GenerateCardデータ
- VerifyCardデータ
- GeneratePinData (VISA/ABA スキームの場合)
- GeneratePinData (の場合IBM3624)
- VerifyPinData (VISA/ABA スキームの場合)
- VerifyPinData (の場合IBM3624)
- データを復号化する
- データを暗号化する
- PIN データ変換
- MAC の生成/検証
- VerifyAuthRequestCryptogram
- インポート/エクスポートキー
- 使用されていないキーのタイプ

GenerateCardデータ

API エンドポイント	暗号化オペレーションまたはアルゴリズム	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
GenerateCardデータ	• AMEX_CARD_SECURITY_CODE_VERSION_1 • AMEX_CARD_SECURITY_CODE_VERSION_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	{ Generate = true },{ Generate = true, Verify = true }
GenerateCardデータ	• CARD_VERIFICATION_VALUE_1 • CARD_VERIFICATION_VALUE_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY	{ Generate = true },{ Generate = true, Verify = true }
GenerateCardデータ	• CARDHOLDER_AUTHENTICATION_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true }
GenerateCardデータ	• DYNAMIC_CARD_VERIFICATION_CODE	TR31_E4_EMV_MKEY_DYNAMIC_NUMBER	• TDES_2KEY	{ DeriveKey = true }
GenerateCardデータ	• DYNAMIC_CARD_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true }

VerifyCardデータ

暗号化オペレーションまたはアルゴリズム	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
• AMEX_CARD_SECURITY_CODE_VERSION_1 • AMEX_CARD_SECURITY_CODE_VERSION_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	{ Generate = true }, { Generate = true, Verify = true }
• CARD_VERIFICATION_VALUE_1 • CARD_VERIFICATION_VALUE_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY	{ Generate = true }, { Generate = true, Verify = true }
• CARDHOLDER_AUTHENTICATION_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true }
• DYNAMIC_CARD_VERIFICATION_CODE	TR31_E4_EMV_MKEY_DYNAMIC_NUMBER	• TDES_2KEY	{ DeriveKey = true }
• DYNAMIC_CARD_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true }

GeneratePinData (VISA/ABA スキームの場合)

`VISA_PIN or VISA_PIN_VERIFICATION_VALUE`

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
PIN 暗号化キー	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	• { Encrypt = true, Wrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true }
PIN 生成キー	TR31_V2_VISA_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true }

GeneratePinData (の場合**IBM3624**)

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
PIN 暗号化キー	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	IBM 3624_NATURAL_PIN、IBM 3624_RANDOM_PIN、IBM 3624_PIN_FROM_OFFSET 用

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
			• { Encrypt = true, Wrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true } IBM 3624_PIN_OFFSET 用 • { Encrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true }
PIN 生成キー	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true }

VerifyPinData (VISA/ABA スキームの場合)

`VISA_PIN`

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
PIN 暗号化キー	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	• { Decrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true }
PIN 生成キー	TR31_V2_VISA_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Verify = true } • { Generate = true, Verify = true }

VerifyPinData (の場合**IBM3624**)

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
PIN 暗号化キー	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	IBM 3624_NATURAL_PIN、IBM 3624_RANDOM_PIN、IBM 3624_PIN_FROM_OFFSET 用 • { Decrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true,

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
			Wrap = true, Unwrap = true } • { NoRestrictions = true }
PIN 検証キー	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Verify = true } • { Generate = true, Verify = true }

データを復号化する

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
DUKPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true } • { NoRestrictions = true }
EMV	TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	• { DeriveKey = true }
RSA	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• { Decrypt = true, Unwrap=true} • {Encrypt=true, Wrap=true,Decrypt = true, Unwrap=true}

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
対称キー	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Decrypt = true, Unwrap=true} • {Encrypt=true, Wrap=true,Decrypt = true, Unwrap=true} • { NoRestrictions = true }

データを暗号化する

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
DUKPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true } • { NoRestrictions = true }
EMV	TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	• { DeriveKey = true }
RSA	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• { Encrypt = true, Wrap=true} • {Encrypt=true, Wrap=true,Decrypt = true, Unwrap=true}

キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
対称キー	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Encrypt = true, Wrap=true} • {Encrypt=true, Wrap=true,Decrypt = true, Unwrap=true} • { NoRestrictions = true }

PIN データ変換

[Direction] (方向)	キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
インバウンドデータソース	DUKPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true } • { NoRestrictions = true }
インバウンドデータソース	DUKPT以外 (PEK、AWK、IWK など)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Decrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true }

[Direction] (方向)	キータイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
アウトバウンドデーターゲット	DUKPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true } • { NoRestrictions = true }
アウトバウンドデーターゲット	DUKPT以外 (PEK、IWK、AWK など)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Encrypt = true, Wrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true }

MAC の生成/検証

MAC キーは、メッセージ/データ本文の暗号化ハッシュを作成するために使用されます。一致するオペレーションを実行できなくなるため、キーモードが制限されたキーを作成することはお勧めしません。ただし、他のシステムがオペレーションペアの残りの半分を実行することを意図している場合、1 つのオペレーションのみでキーをインポート/エクスポートできます。

許可されたキーの使用	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
MAC キー	TR31_M1_ISO_9797_1_MAC_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true }

許可されたキーの使用	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
			• { Generate = true }
MAC キー (小売 MAC)	TR31_M1_ISO_9797_3_MAC_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true } • { Generate = true }
MAC キー (CMAC)	TR31_M6_ISO_9797_5_CMAC_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true } • { Generate = true }
MAC キー (HMAC)	TR31_M7_HMAC_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true } • { Generate = true }

VerifyAuthRequestCryptogram

許可されたキーの使用	EMV オプション	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
• オプション A • オプション B	TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS	• TDES_2KEY	• { DeriveKey = true }

インポート/エクスポートキー

オペレーションのタイプ	許可されたキーの使用	許可されたキーアルゴリズム	許可されたキーモードの組み合わせ
TR-31 ラップキー	TR31_K1_KEY_BLOCK_PROTECTION_KEY TR31_K0_KEY_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128	• { 暗号化 = true、ラップ = true } (エクスポートのみ) • { Decrypt = true、Unwrap = true } (インポートのみ) • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true }
信頼できる CA のインポート	TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE	• RSA_2048 • RSA_3072 • RSA_4096	• { Verify = true }
非対称暗号化用のパブリックキー証明書のインポート	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• { Encrypt=true,Wrap=true }

使用されていないキーのタイプ

以下のキータイプは現在 AWS Payment Cryptography では使用されていません

- TR31_P1_PIN_GENERATION_KEY
- TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT

AWS Payment Cryptography のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ — クラウドで AWS サービスを実行するインフラストラクチャを保護する責任AWS は AWS にあります。 AWS また、 では、安全に使用できるサービスも提供しています。コンプライアンスAWS プログラムコンプライアンスプログラム の一環として、サードパーティーの監査者は定期的にセキュリティの有効性をテストおよび検証。 AWS Payment Cryptography に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」「コンプライアンスプログラム」を参照してください。
- クラウドのセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このトピックは、 AWS Payment Cryptography を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。セキュリティとコンプライアンスの目的を達成するために AWS Payment Cryptography を設定する方法を示します。また、 AWS Payment Cryptography リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

トピック

- AWS Payment Cryptography でのデータ保護
- AWS Payment Cryptography の耐障害性
- のインフラストラクチャセキュリティ AWS Payment Cryptography
- VPC エンドポイントを介した AWS Payment Cryptography への接続
- AWS Payment Cryptography のセキュリティのベストプラクティス

AWS Payment Cryptography でのデータ保護

責任 AWS 共有モデル、 AWS Payment Cryptography でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「AWS 責任共有モデルおよび GDPR」を参照してください。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または SDK を使用して AWS Payment Cryptography AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

AWS Payment Cryptography は暗号化キーを保存して保護し、高可用性を実現すると同時に、強力で柔軟なアクセス制御を提供します。

トピック

キーマテリアルの保護

デフォルトでは、AWS Payment Cryptography は、サービスによって管理される支払いキーの暗号化キーマテリアルを保護します。さらに、AWS Payment Cryptography には、サービスの外部で作成されたキーマテリアルをインポートするオプションもあります。KMS キーとキーマテリアルの技術的な詳細については、「AWS Payment Cryptography 暗号化の詳細」を参照してください。

データ暗号化

AWS Payment Cryptographyのデータは、AWS Payment Cryptography キー、それらが表す暗号化キーマテリアル、およびそれらの使用属性で構成されます。このキーマテリアルは、 AWS Payment Cryptography ハードウェアセキュリティモジュール (HSM) 内でのみ、使用中の場合にのみ、プレーンテキストで存在します。それ以外の場合、キー素材は暗号化され、属性は耐久性のある永続ストレージに保存されます。

AWS Payment Cryptography が支払いキー用に生成またはロードするキーマテリアルは、AWS Payment Cryptography HSM の境界を暗号化されずに残ることはありません。AWS Payment Cryptography API オペレーションによって暗号化されてエクスポートできます。

保管中の暗号化

AWS Payment Cryptographyは、PCI PTS HSM に登録されている HSM 内の支払いキーのキーマテリアルを生成します。使用されていない場合、キーマテリアルは HSM キーによって暗号化され、耐久性のある永続的なストレージに書き込まれます。Payment Cryptography キーのキーマテリアルおよびキーマテリアルを保護する暗号化キーは、HSM をプレーンテキスト形式のままにしません。

Payment Cryptography キーのキーマテリアルの暗号化と管理は、サービスによって完全に処理されます。

詳細については、「AWS キーマネジメントサービス暗号化の詳細」を参照してください。

転送中の暗号化

AWS Payment Cryptography が支払いキー用に生成またはロードするキーマテリアルは、AWS Payment Cryptography API オペレーションでクリアテキストでエクスポートまたは送信されることはありません。AWS Payment Cryptography は、API オペレーションの CMK を表すために キー識別子 を使用します。

ただし、AWS Payment Cryptography API オペレーションの中には、以前に共有されたキーまたは非対称キー交換キーによって暗号化されたキーを エクスポートするものがあります。お客様は API オペレーションを使用して、選択したキーのキーマテリアルをインポートすることもできます。

AWS Payment Cryptography API コールはすべて、署名し、Transport Layer Security (TLS) を使用して送信する必要があります。AWS Payment Cryptography には、PCI が「強力な暗号化」として定義した TLS バージョンと暗号スイートが必要です。すべてのサービスエンドポイントは TLS 1.0—1.3 とハイブリッドポストクォンタム TLS をサポートしています。

詳細については、「AWS キーマネジメントサービス暗号化の詳細」を参照してください。

インターネットトラフィックのプライバシー

AWS Payment Cryptography は、AWS マネジメントコンソールおよび一連の API オペレーションをサポートしています。これにより、決済キーを作成および管理し、暗号化オペレーションで使用することができます。

AWS Payment Cryptography は、プライベートネットワークから AWS への 2 つのネットワーク接続オプションをサポートしています。

- インターネット経由の IPsec VPN 接続
- AWS Direct Connect, ：標準イーサネット光ファイバケーブルを介して、内部ネットワークを AWS Direct Connect ロケーションにリンクします。

API 呼び出しすべて、署名し、Transport Layer Security (TLS) を使用して送信する必要があります。呼び出しには、 完全な転送秘密をサポートする最新の暗号スイートも必要です。キーのキーマテリアルを保存するハードウェアセキュリティモジュール (HSM) へのトラフィックは、 の内部ネットワーク経由で既知の API ホストからのみ許可されます。

パブリックインターネット経由でトラフィックを送信せずに Virtual Private Cloud (VPC) から AWS Payment Cryptography に直接接続するには、AWS を搭載した VPC エンドポイントを使用します

PrivateLink。詳細については、「VPC エンドポイントを介した AWS Payment Cryptographyへの接続」をご参照ください。

AWS Payment Cryptography は、Transport Layer Security (TLS)ネットワーク暗号化プロトコル用の ハイブリッドポスト量子キー交換オプションもサポートしています。このオプションは、 AWS Payment Cryptography API エンドポイントへの接続時に TLS で使用できます。

AWS Payment Cryptography の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立および隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

リージョンの隔離

AWS Payment Cryptography は、複数のリージョンで利用できるリージョンサービスです。

AWS Payment Cryptography は地域的に分離された設計になっているため、ある AWS リージョンで発生した可用性の問題が他のリージョンの AWS Payment Cryptography の運用に影響を与えることはありません。AWS Payment Cryptography は、すべてのソフトウェアアップデートとスケーリングオペレーションがシームレスかつ気付かないうちに実行されるため、計画的なダウンタイムがゼロになるように設計されています。

AWS Payment Cryptography サービスレベルアグリーメント (SLA) には、すべての Payment Cryptography API に対して 99.99% のサービスコミットメントが含まれています。このコミットメントを実現するために、AWS Payment Cryptography は API リクエストの実行に必要なすべてのデータと認証情報が、リクエストを受信するすべてのリージョンのホストで利用可能であることを確認します。

AWS Payment Cryptography インフラストラクチャは、各リージョンの 3 つ以上のアベイラビリティーゾーン (AZ) にレプリケートされます。複数のホスト障害による のパフォーマンスへの影

響を受けないように、リージョンのどの AZ からの顧客トラフィックでも処理するように、AWS Payment Cryptography は設計されています。

決済キーのプロパティまたは許可に加えた変更は、リージョン内のすべてのホストにレプリケートされ、後続のリクエストがリージョン内の任意のホストで正しく処理されるようにします。決済キーを使用した暗号化オペレーションのリクエストは、 ハードウェアセキュリティモジュール (HSM) のフリートに転送され、そのいずれもがキーを使用してオペレーションを実行できます。

マルチテナント設計

AWS Payment Cryptography のマルチテナント設計により、SLA の可用性を満たし、高いリクエストレートを維持しながら、キーとデータの機密性を保護できます。

暗号化オペレーションに指定したキーが常に使用される決済キーであることを保証するために、複数の整合性強制メカニズムがデプロイされます。

Payment Cryptography キーのプレーンテキストキーマテリアルは、広範囲に保護されています。キーマテリアルは作成後すぐに HSM で暗号化され、暗号化されたキーマテリアルはセキュアなストレージに即座に移動されます。暗号化されたキーは、HSM 内で取得され、使用に間に合うように復号されます。プレーンテキストキーは、暗号化オペレーションを完了するのに必要な時間だけ HSM メモリに残ります。プレーンテキストのキーマテリアルが HSM を離れることはありません。永続ストレージに書き込まれることもありません。

AWS Payment Cryptography がキーを保護するために使用するメカニズムの詳細は、「AWS Payment Cryptography 暗号化の詳細」を参照してください。

のインフラストラクチャセキュリティ AWS Payment Cryptography

マネージドサービスである AWS Payment Cryptography は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている AWS グローバルネットワークセキュリティの手順で保護されています。

が AWS 公開した API コールを使用して、ネットワーク AWS Payment Cryptography 経由で にアクセスします。クライアントは、Transport Layer Security (TLS) 1.2 以降をサポートする必要があります。クライアントは、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用する暗号スイートもサポートする必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token ServiceAWS STSを使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

物理ホストの分離

AWS Payment Cryptography が使用する物理的インフラストラクチャのセキュリティは、Amazon Web Services: セキュリティプロセスの概要の物理的および環境的セキュリティのセクションで説明されている制御に支配されます。詳細については、前のセクションにリストされたコンプライアンスレポートとサードパーティーの監査結果を参照してください。

AWS Payment Cryptography は、専用の commercial-off-the-shelf PCI PTS HSM 認定ハードウェアセキュリティモジュール (HSMs。AWS Payment Cryptography キーのキーマテリアルは、Payment Cryptography キーの使用中にのみ、HSM の揮発性メモリにのみ保存されます。HSM は Amazon データセンター内のアクセス制御ラックに設置されており、あらゆる物理的アクセスを二重に制御します。AWS Payment Cryptography HSM のオペレーションの詳細については、「AWS Payment Cryptography 暗号化の詳細」を参照してください。

VPC エンドポイントを介した AWS Payment Cryptography への接続

Virtual Private Cloud (VPC) のプライベートインターフェイスエンドポイントを介して AWS Payment Cryptography に直接接続できます。インターフェイス VPC エンドポイントを使用すると、VPC と AWS Payment Cryptography 間の通信は AWS 、ネットワーク内で完全に行われます。

AWS Payment Cryptography は、 を利用した Amazon Virtual Private Cloud (Amazon VPC) エンドポイントをサポートしていますAWS PrivateLink。各 VPC エンドポイントは、VPC サブネット内のプライベート IP アドレスを持つ 1 つ以上の Elastic Network Interfaces (ENI) で表されます。

インターフェイス VPC エンドポイントは、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続なしで VPC を AWS Payment Cryptography に直接接続します。VPC 内のインスタンスは、パブリック IP アドレスがなくても AWS Payment Cryptography と通信できます。

リージョン

AWS Payment Cryptography は、 Payment Cryptography がサポートされているすべての AWS リージョン で VPC エンドポイントと VPC エンドポイントポリシーをサポートします。 AWS

トピック

AWS Payment Cryptography VPC エンドポイントに関する考慮事項

AWS Payment Cryptography のインターフェイス VPC エンドポイントを設定する前に、「 AWS PrivateLink ガイド」の「インターフェイスエンドポイントのプロパティと制限」トピックを確認してください。

AWS VPC エンドポイントの Payment Cryptography サポートには、以下が含まれます。

- VPC エンドポイントを使用して、VPC AWS からすべての Payment Cryptography Controlplane オペレーションと AWS Payment Cryptography Dataplane オペレーションを呼び出すことができます。
- AWS Payment Cryptography リージョンエンドポイントに接続するインターフェイス VPC エンドポイントを作成できます。
- AWS Payment Cryptography は、コントロールプレーンとデータプレーンで構成されます。1 つまたは両方のサブサービスを設定できますが、それぞれが個別に設定されます。
- AWS CloudTrail ログを使用して、VPC エンドポイントを介した AWS Payment Cryptography キーの使用を監査できます。詳細については、「VPC エンドポイントのログ記録」を参照してください。

AWS Payment Cryptography 用の VPC エンドポイントの作成

AWS Payment Cryptography の VPC エンドポイントは、Amazon VPC コンソールまたは Amazon VPC API を使用して作成できます。詳細については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントを作成」を参照してください。

- AWS Payment Cryptography 用の VPC エンドポイントを作成するには、次のサービス名を使用します。

```
com.amazonaws.region.payment-cryptography.controlplane
```

```
com.amazonaws.region.payment-cryptography.dataplane
```

例えば、米国西部 (オレゴン) リージョン (us-west-2) では、サービス名は次のようになります。

```
com.amazonaws.us-west-2.payment-cryptography.controlplane
```

```
com.amazonaws.us-west-2.payment-cryptography.dataplane
```

VPC エンドポイントを使いやすくするために、VPC エンドポイントに対してプライベート DNS 名を有効にすることができます。DNS 名を有効にするオプションを選択すると、標準の AWS Payment Cryptography DNS ホスト名が VPC エンドポイントに解決されます。例えば、`https://controlplane.payment-cryptography.us-west-2.amazonaws.com` はサービス名 `com.amazonaws.us-west-2.payment-cryptography.controlplane` に接続された VPC エンドポイントに解決されます。

このオプションにより VPC エンドポイントが使いやすくなります。 AWS SDKsと はデフォルトで標準の AWS Payment Cryptography DNS ホスト名 AWS CLI を使用するため、アプリケーションとコマンドで VPC エンドポイント URL を指定する必要はありません。

詳細については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントを介したサービスへアクセスする」を参照してください。

AWS Payment Cryptography VPC エンドポイントへの接続

AWS SDK、、 AWS CLI または を使用して、VPC エンドポイントを介して AWS Payment Cryptography に接続できます AWS Tools for PowerShell。VPC エンドポイントを指定するには、DNS 名を使用します。

例えば、この list-keys コマンドは、 `endpoint-url` パラメータを使用して VPC エンドポイントを指定します。こうしたコマンドを使用するには、サンプルの VPC エンドポイント ID を、ご自身のアカウントのものに置き換えてください。

```
$ aws payment-cryptography list-keys --endpoint-url
```

VPC エンドポイントの作成時にプライベートホスト名を有効にした場合は、CLI コマンドまたはアプリケーションの設定で VPC エンドポイント URL を指定する必要はありません。標準の AWS Payment Cryptography DNS ホスト名は VPC エンドポイントに解決されます。 AWS CLI および SDKs はデフォルトでこのホスト名を使用するため、VPC エンドポイントを使用して AWS Payment Cryptography リージョンエンドポイントに接続し始めることができます。スクリプトやアプリケーションに変更を加える必要はありません。

プライベートホスト名を使用するには、VPC の `enableDnsHostnames` 属性と `enableDnsSupport` 属性を `true` に設定する必要があります。これらの属性を設定するには、ModifyVpc属性オペレーションを使用します。詳細については、「Amazon VPC ユーザーガイド」の「VPC の DNS 属性の表示と更新」を参照してください。

VPC エンドポイントへのアクセスの制御

AWS Payment Cryptography の VPC エンドポイントへのアクセスを制御するには、VPC エンドポイントポリシーを VPC エンドポイントにアタッチします。エンドポイントポリシーは、プリンシパルが VPC エンドポイントを使用して、特定の AWS Payment Cryptography リソースで AWS Payment Cryptography オペレーションを呼び出すことができるかどうかを決定します。

エンドポイントの作成時に VPC エンドポイントポリシーを作成できます。また、VPC エンドポイントポリシーはいつでも変更できます。VPC マネジメントコンソール、またはCreateVpcエンドポイントまたはModifyVpcエンドポイントオペレーションを使用します。AWS CloudFormation テンプレート を使用して VPC エンドポイントポリシーを作成および変更することもできます。VPC マネジメントコンソールの使用方法については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントの作成」および「インターフェイスエンドポイントの変更」を参照してください。

トピック

- VPC エンドポイントポリシーについて
- デフォルトの VPC エンドポイントポリシー
- VPCエンドポイントポリシーの作成
- VPC エンドポイントポリシーの表示

VPC エンドポイントポリシーについて

VPC エンドポイントを使用する AWS Payment Cryptography リクエストを正常に実行するには、プリンシパルに次の 2 つのソースからのアクセス許可が必要です。

- アイデンティティベースのポリシーは、プリンシパルにリソース (AWS Payment Cryptography キーまたはエイリアス) で オペレーションを呼び出すアクセス許可を付与する必要があります。
- VPC エンドポイントポリシーは、エンドポイントを使用してリクエストを実行するためのアクセス権限をプリンシパルに付与する必要があります。

例えば、キーポリシーは、プリンシパルに特定の AWS Payment Cryptography キーで Decrypt を呼び出すアクセス許可を付与する場合があります。ただし、DecryptVPC エンドポイントポリシーでは、そのプリンシパルがエンドポイントを使用して AWS Payment Cryptography キーで を呼び出すことを許可しない場合があります。

または、VPC エンドポイントポリシーにより、プリンシパルがエンドポイントを使用して特定の AWS Payment Cryptography キーで StopKeyUsage を呼び出すことが許可される場合があります。ただし、プリンシパルに IAM ポリシーからのアクセス許可がない場合、リクエストは失敗します。

デフォルトの VPC エンドポイントポリシー

すべての VPC エンドポイントには VPC エンドポイントポリシーがありますが、ポリシーを指定する必要はありません。ポリシーを指定しない場合、デフォルトのエンドポイントポリシーでは、エンドポイント上のすべてのリソースのすべてのプリンシパルによるすべてのオペレーションが許可されます。

ただし、 AWS Payment Cryptography リソースの場合、プリンシパルには IAM ポリシー から オペレーションを呼び出すアクセス許可も必要です。したがって、実際には、デフォルトポリシーでは、プリンシパルがリソースに対してオペレーションを呼び出す権限を持っている場合、エンドポイントを使用してオペレーションを呼び出すこともできます。

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

許可されたオペレーションのサブセットのみに VPC エンドポイントを使用することをプリンシパルに許可するには、VPC エンドポイントポリシーを作成または変更します。

VPCエンドポイントポリシーの作成

VPC エンドポイントポリシーは、プリンシパルに VPC エンドポイントを使用してリソースに対してオペレーションを実行するアクセス許可があるかどうかを決定します。 AWS Payment Cryptography リソースの場合、プリンシパルには IAM ポリシー からのオペレーションを実行するアクセス許可も必要です。

各 VPC エンドポイントポリシーステートメントには、次の要素が必要です。

- アクションを実行できるプリンシパル
- 実行可能なアクション
- アクションを実行できるリソース

ポリシーステートメントは VPC エンドポイントを指定しません。代わりに、ポリシーがアタッチされているすべての VPC エンドポイントに適用されます。詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

AWS Payment Cryptography の VPC エンドポイントポリシーの例を次に示します。VPC エンドポイントにアタッチされると、このポリシーは、VPC エンドポイントExampleUserを使用して、指定された AWS Payment Cryptography キーで指定されたオペレーションを呼び出すことを許可します。このようなポリシーを使用する前に、サンプルプリンシパルとキー識別子をアカウントの有効な値に置き換えてください。

```
{
   "Statement":[
      {
         "Sid": "AllowDecryptAndView",
         "Principal": {"AWS": "arn:aws:iam::111122223333:user/ExampleUser"},
         "Effect":"Allow",
         "Action": [
             "payment-cryptography:Decrypt",
             "payment-cryptography:GetKey",
             "payment-cryptography:ListAliases",
             "payment-cryptography:ListKeys",
             "payment-cryptography:GetAlias"
          ],
         "Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
      }
   ]
```

```
}
```

AWS CloudTrail は、VPC エンドポイントを使用するすべてのオペレーションを記録します。ただし、 CloudTrail ログには、他のアカウントのプリンシパルによってリクエストされたオペレーションや、他のアカウントの AWS Payment Cryptography キーのオペレーションは含まれません。

そのため、外部アカウントのプリンシパルが VPC エンドポイントを使用して、ローカルアカウントの任意のキーで AWS Payment Cryptography オペレーションを呼び出すことを禁止する VPC エンドポイントポリシーを作成できます。

次の例では、aws:PrincipalAccount グローバル条件キーを使用して、プリンシパルがローカルアカウントにある場合を除き、すべての AWS Payment Cryptography キーに対するすべてのオペレーションのすべてのプリンシパルへのアクセスを拒否します。このようなポリシーを使用する前に、サンプルアカウント ID を有効なものに置き換えてください。

```
{
  "Statement": [
    {
      "Sid": "AccessForASpecificAccount",
      "Principal": {"AWS": "*"},
      "Action": "payment-cryptography:*",
      "Effect": "Deny",
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": "111122223333"
        }
      }
    }
  ]
}
```

VPC エンドポイントポリシーの表示

エンドポイントの VPC エンドポイントポリシーを表示するには、VPC 管理コンソールまたはDescribeVpcエンドポイントオペレーションを使用します。

次の AWS CLI コマンドは、指定された VPC エンドポイント ID を持つエンドポイントのポリシーを取得します。

このコマンドを使用する前に、サンプルのエンドポイント ID をアカウントの有効なものに置き換えてください。

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==``].[PolicyDocument]'
--output text
```

ポリシーステートメントでの VPC エンドポイントの使用

リクエストが VPC から送信されたとき、または VPC エンドポイントを使用する場合に、 AWS Payment Cryptography のリソースとオペレーションへのアクセスを制御できます。これを行うには、IAM ポリシーを 1 つ使用します。

- `aws:sourceVpce` 条件キーを使用して、VPC エンドポイントに基づいてアクセスを許可または制限します。
- `aws:sourceVpc` 条件キーを使用して、プライベートエンドポイントをホストする VPC に基づいてアクセスを許可または制限します。

> **Note**
>
> リクエストが Amazon VPC エンドポイント から送信された場合、 `aws:sourceIP`条件キーは有効ではありません。リクエストを VPC エンドポイントに制限するには、`aws:sourceVpce` または `aws:sourceVpc` 条件キーを使用します。詳細については、「AWS PrivateLink ガイド」の「VPC エンドポイントおよび VPC エンドポイントサービスの Identity and Access Management」を参照してください。

これらのグローバル条件キーを使用して、 AWS Payment Cryptography キー、エイリアス、および特定のリソースに依存しCreateKeyない などのオペレーションへのアクセスを制御できます。

例えば、次のサンプルキーポリシーでは、リクエストが指定された VPC エンドポイントを使用している場合にのみ、ユーザーが AWS Payment Cryptography キーを使用して特定の暗号化オペレーションを実行し、インターネットと接続の両方からのアクセスをブロックできます (設定されている場合) 。ユーザーが AWS Payment Cryptography にリクエストを行うと、リクエストの VPC エンドポイント ID がポリシー`aws:sourceVpce`の条件キー値と比較されます。一致しない場合、要求は拒否されます。

このようなポリシーを使用するには、プレースホルダー AWS アカウント ID と VPC エンドポイント IDsをアカウントの有効な値に置き換えます。

```
{
```

```
    "Id": "example-key-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM policies",
            "Effect": "Allow",
            "Principal": {"AWS":["111122223333"]},
            "Action": ["payment-cryptography:*"],
            "Resource": "*"
        },
        {
            "Sid": "Restrict usage to my VPC endpoint",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "payment-cryptography:Encrypt",
                "payment-cryptography:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": ""
                }
            }
        }

    ]
}
```

aws:sourceVpc 条件キーを使用して、VPC エンドポイントが存在する VPC に基づいて AWS Payment Cryptography キーへのアクセスを制限することもできます。

次のサンプルキーポリシーでは、 AWS Payment Cryptography キーが から取得された場合にのみキーを管理するコマンドを許可しますvpc-12345678。さらに、 AWS Payment Cryptography キーを暗号化オペレーションに使用するコマンドは、 から取得した場合にのみ許可されますvpc-2b2b2b2b。ある VPC でアプリケーションが実行されていれば、このようなポリシーを使用できますが、管理機能のために 2 番目の切り離された VPC を使用します。

このようなポリシーを使用するには、プレースホルダー AWS アカウント ID と VPC エンドポイント IDsをアカウントの有効な値に置き換えます。

```
{
```

```
    "Id": "example-key-2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow administrative actions from vpc-12345678",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:Create*","payment-
cryptography:Encrypt*","payment-cryptography:ImportKey*","payment-
cryptography:GetParametersForImport*",
                "payment-cryptography:TagResource", "payment-
cryptography:UntagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-12345678"
                }
            }
        },
        {
            "Sid": "Allow key usage from vpc-2b2b2b2b",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:Encrypt","payment-cryptography:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-2b2b2b2b"
                }
            }
        },
        {
            "Sid": "Allow list/read actions from everywhere",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:List*","payment-cryptography:Get*"
            ],
            "Resource": "*",
        }
```

```
    ]
}
```

VPC エンドポイントのログ記録

AWS CloudTrail は、VPC エンドポイントを使用するすべてのオペレーションを記録します。 AWS Payment Cryptography へのリクエストが VPC エンドポイントを使用する場合、VPC エンドポイント ID はリクエストを記録するAWS CloudTrail ログエントリに表示されます。エンドポイント ID を使用して、 AWS Payment Cryptography VPC エンドポイントの使用を監査できます。

VPC を保護するために、VPC エンドポイントポリシー によって拒否されたが、それ以外の場合は許可されていたリクエストは に記録されませんAWS CloudTrail。

例えば、このサンプルログエントリは、VPC エンドポイントを使用したGenerateMacリクエストを記録します。vpcEndpointId フィールドは、ログエントリの最後に表示されます。

```
{
      "eventVersion": "1.08",
      "userIdentity": {
          "principalId": "TESTXECZ5U9M4LGF2N6Y5:",
          "arn": "arn:aws:sts::111122223333:assumed-role//",
          "accountId": "111122223333",
          "accessKeyId": "TESTXECZ5U2ZULLHHMJG",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "TESTXECZ5U9M4LGF2N6Y5",
                  "arn": "arn:aws:iam::111122223333:role/",
                  "accountId": "111122223333",
                  "userName": ""
              },
              "webIdFederationData": {},
              "attributes": {
                  "creationDate": "2024-05-27T19:34:10Z",
                  "mfaAuthenticated": "false"
              },
              "ec2RoleDelivery": "2.0"
          }
      },
      "eventTime": "2024-05-27T19:49:54Z",
      "eventSource": "payment-cryptography.amazonaws.com",
      "eventName": "CreateKey",
```

```
      "awsRegion": "us-east-1",
      "sourceIPAddress": "172.31.85.253",
      "userAgent": "aws-cli/2.14.5 Python/3.9.16 Linux/6.1.79-99.167.amzn2023.x86_64
 source/x86_64.amzn.2023 prompt/off command/payment-cryptography.create-key",
      "requestParameters": {
          "keyAttributes": {
              "keyUsage": "TR31_M1_ISO_9797_1_MAC_KEY",
              "keyClass": "SYMMETRIC_KEY",
              "keyAlgorithm": "TDES_2KEY",
              "keyModesOfUse": {
                  "encrypt": false,
                  "decrypt": false,
                  "wrap": false,
                  "unwrap": false,
                  "generate": true,
                  "sign": false,
                  "verify": true,
                  "deriveKey": false,
                  "noRestrictions": false
              }
          },
          "exportable": true
      },
      "responseElements": {
          "key": {
              "keyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
              "keyAttributes": {
                  "keyUsage": "TR31_M1_ISO_9797_1_MAC_KEY",
                  "keyClass": "SYMMETRIC_KEY",
                  "keyAlgorithm": "TDES_2KEY",
                  "keyModesOfUse": {
                      "encrypt": false,
                      "decrypt": false,
                      "wrap": false,
                      "unwrap": false,
                      "generate": true,
                      "sign": false,
                      "verify": true,
                      "deriveKey": false,
                      "noRestrictions": false
                  }
              },
              "keyCheckValue": "A486ED",
```

```
                "keyCheckValueAlgorithm": "ANSI_X9_24",
                "enabled": true,
                "exportable": true,
                "keyState": "CREATE_COMPLETE",
                "keyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                "createTimestamp": "May 27, 2024, 7:49:54 PM",
                "usageStartTimestamp": "May 27, 2024, 7:49:54 PM"
            }
        },
        "requestID": "f3020b3c-4e86-47f5-808f-14c7a4a99161",
        "eventID": "b87c3d30-f3ab-4131-87e8-bc54cfef9d29",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "vpcEndpointId": "",
        "eventCategory": "Management",
        "tlsDetails": {
            "tlsVersion": "TLSv1.3",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "clientProvidedHostHeader": "-oo28vrvr.controlplane.payment-cryptography.us-
east-1.vpce.amazonaws.com"
        }
    }
```


AWS Payment Cryptography のセキュリティのベストプラクティス

AWS Payment Cryptography は、暗号化キーの保護を強化し、意図した目的で使用されるように、組み込みまたはオプションで実装できる多くのセキュリティ機能をサポートしています。これには、IAM ポリシー、キーポリシーと IAM ポリシーを改良するための広範なポリシー条件キーのセット、およびキーブロックに関する PCI PIN ルールの組み込み適用が含まれます。

> **Important**
>
> これらの一般的なガイドラインは、完全なセキュリティソリューションを提供するものではありません。すべてのベストプラクティスがあらゆる状況に適しているわけではないため、これらは規範的なものではありません。

- キーの使用方法と使用方法： AWS Payment Cryptography は、ANSI X9 TR 31-2018 Interoperable Secure Key Exchange Key Block Specification で説明され、PCI PIN セキュリティ要件 18-3 に準拠しているように、キーの使用方法と使用方法の制限に従い、適用します。これにより、1 つのキーを複数の目的で使用できなくなり、キーメタデータ (許可されたオペレーションなど) がキーマテリアル自体に暗号的にバインドされます。 AWS Payment Cryptography は、キー暗号化キー (TR31_K0_KEY_ENCRYPTION_KEY) をデータ復号化に使用できないなど、これらの制限を自動的に適用します。詳細については、「AWS Payment Cryptography キーのキー属性について」を参照してください。
- 対称キーマテリアルの共有を制限する：対称キーマテリアル (PIN 暗号化キーやキー暗号化キーなど) は、多くても他の 1 つのエンティティとのみ共有できます。機密性の高いマテリアルをより多くのエンティティまたはパートナーに転送する必要がある場合は、追加のキーを作成します。AWS Payment Cryptography は、対称キーマテリアルまたは非対称プライベートキーマテリアルをクリアに公開しません。
- エイリアスやタグを使用して、キーを特定のユースケースやパートナーに関連付ける：エイリアスを使用すると、キーに関連するユースケースを簡単に示すことができます。例えば、Alias/BIN_12345_CVK は BIN 12345 に関連するカード検証キーを表すのに便利です。より高い柔軟性が必要な場合は、bin=12345、use_case=acquiring、country=us、partner=foo などのタグを作成することを検討してください。エイリアスやタグは、ユースケースの発行と取得の間にアクセス制御を強制するなど、アクセスを制限するためにも使用できます。
- 最小許可アクセスを実践する：IAM を使用すると、個々のユーザーがキーを作成したり、暗号化オペレーションを実行したりすることを禁止するなど、本番環境へのアクセスを個人ではなくシステムに制限できます。IAM は、取得者による PIN の生成や検証を制限するなど、ユースケースには当てはまらないコマンドとキーの両方へのアクセスを制限するためにも使用できます。最小特権を使用するもう 1 つの方法は、機密性の高いオペレーション (キーのインポートなど) を特定のサービスアカウントに制限することです。例については、「AWS Payment Cryptography のアイデンティティベースのポリシーの例」を参照してください。

以下の資料も参照してください。

- AWS Payment Cryptography の Identity and Access Management
- 「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」

AWS Payment Cryptography のコンプライアンス検証

AWS Payment Cryptography のセキュリティとコンプライアンスは、複数の AWS コンプライアンスプログラムの一環として、サードパーティーの監査機関によって評価されます。このプログラムには、SOC、PCI などが含まれます。

AWS Payment Cryptography は、PCI DSS に加えていくつかの PCI 標準で評価されています。これらには、PCI PIN セキュリティ (PCI PIN) と PCI ポイントツーポイント (P2PE) 暗号化が含まれます。入手可能な認証とコンプライアンスガイドについては、AWS Artifactを参照してください。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」を参照してください。一般的な情報については、AWS コンプライアンスプログラムを参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、「AWS Artifact 内のレポートのダウンロード」「」を参照してください。

AWS Payment Cryptography を使用する際のコンプライアンス責任は、データの機密性、企業のコンプライアンス目的、適用法規によって決定されます。AWS はコンプライアンスに役立つ以下のリソースをご用意しています。

- 「セキュリティとコンプライアンスのクイックスタートガイド」「」– これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、AWS でセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするためのステップが記載されています。
- 「AWS コンプライアンスのリソース」「」– このワークブックおよびガイドのコレクションは、顧客の業界と拠点に適用されるものである場合があります。
- AWS Config デベロッパーガイドの「ルールでのリソースの評価」– AWS Config は、リソース設定が、社内のプラクティス、業界のガイドラインそして規制にどの程度適合しているのかを評価します。
- 「AWS Security Hub」- この AWS サービスでは、AWS 内のセキュリティ状態を包括的に表示しており、セキュリティ業界の標準およびベストプラクティスへの準拠を確認するのに役立ちます。

AWS Payment Cryptography の Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS Payment Cryptography リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

対象者

AWS Identity and Access Management (IAM) の使用方法は、 AWS Payment Cryptography で行う作業によって異なります。

サービスユーザー – AWS Payment Cryptography サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの AWS Payment Cryptography 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。 AWS Payment Cryptography の機能にアクセスできない場合は、「AWS Payment Cryptography のアイデンティティとアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の AWS Payment Cryptography リソースを担当している場合は、通常、AWS Payment Cryptography へのフルアクセスがあります。サービスユーザーがどの AWS Payment Cryptography 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。Payment Cryptography で IAM AWS を使用する方法の詳細については、「」を参照してくださいAWS Payment Cryptography と IAM の連携方法。

IAM 管理者 – IAM 管理者は、 AWS Payment Cryptography へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。IAM で使用できる AWS Payment Cryptography アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS Payment Cryptography のアイデンティティベースのポリシーの例。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーション ID の例です。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、『AWS IAM Identity Center ユーザーガイド』の「Multi-factor authentication」(多要素認証) および『IAM ユーザーガイド』の「AWSにおける多要素認証 (MFA) の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての AWS のサービス およびリソースへの完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウ

ント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAM ユーザーガイドの「ルートユーザー認証情報が必要なタスク」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイドの「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、『IAM ユーザーガイド』の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッドアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、『IAM ユーザーガイド』の「サードパーティーアイデンティティプロバイダー向けロールの作成」を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、『AWS IAM Identity Center ユーザーガイド』の「権限セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスにリンクされたロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) — IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細に

ついては、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。

- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、『IAM ユーザーガイド』の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティの許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

AWS Payment Cryptography と IAM の連携方法

IAM を使用して AWS Payment Cryptography へのアクセスを管理する前に、 AWS Payment Cryptography で使用できる IAM 機能を理解しておく必要があります。 AWS Payment Cryptography およびその他の AWS のサービスが IAM と連携する方法の概要を把握するには、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

トピック

- AWS Payment Cryptography のアイデンティティベースのポリシー
- AWS Payment Cryptography タグに基づく承認

AWS Payment Cryptography のアイデンティティベースのポリシー

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、アクションを許可または拒否する条件を指定できます。 AWS Payment Cryptography は、特定のアクション、リソース、および条件キーをサポートします。JSON ポリシーで使用するすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

アクション

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、 依存アクション と呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

AWS Payment Cryptography のポリシーアクションは、アクションの前にプレフィックス を使用しますpayment-cryptography:。例えば、 AWS Payment Cryptography `VerifyCardData` API オペレーションを実行するアクセス許可を付与するには、ポリシーに `payment-cryptography:VerifyCardData`アクションを含めます。ポリシーステートメントには、`Action`

または NotAction 要素を含める必要があります。 AWS Payment Cryptography は、このサービスで実行できるタスクを記述する独自のアクションのセットを定義します。

単一ステートメントに複数アクションを指定するには、次のようにカンマで区切ります:

```
"Action": [
      "payment-cryptography:action1",
      "payment-cryptography:action2"
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、List という単語で始まるすべてのアクション(ListKeys や ListAliases など) を指定するには、次のアクションを含めます。

```
"Action": "payment-cryptography:List*"
```

AWS Payment Cryptography アクションのリストを確認するには、IAM ユーザーガイドのAWS「 Payment Cryptography で定義されるアクション」を参照してください。

リソース

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

決済暗号 (payment-cryptography) キーリソースには次のような ARN があります。

```
arn:${Partition}:payment-cryptography:${Region}:${Account}:key/${keyARN}
```

ARN の形式の詳細については、「Amazon リソースネーム (ARNs AWS 「サービス名前空間」を参照してください。

例えば、ステートメントで arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h インスタンスを指定するには、次の ARN を使用します:

```
"Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h"
```

特定のアカウントに属するすべてのキーを指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/*"
```

キーを作成するためのアクションなど、一部の AWS Payment Cryptography アクションは、特定のリソースで実行できません。このような場合は、ワイルドカード *を使用する必要があります。

```
"Resource": "*"
```

1 つのステートメントで複数のリソースを指定するには、以下のようにカンマを使用します。

```
"Resource": [
      "resource1",
      "resource2"
```

例

AWS Payment Cryptography のアイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS Payment Cryptography のアイデンティティベースのポリシーの例。

AWS Payment Cryptography タグに基づく承認

AWS Payment Cryptography のアイデンティティベースのポリシーの例

デフォルトでは、IAM ユーザーおよびロールには、 AWS Payment Cryptography リソースを作成または変更するアクセス許可はありません。また、、 AWS Management Console AWS CLI、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行する権限をユーザーとロールに付与す

る IAM ポリシーを作成する必要があります。続いて、管理者はそれらの権限が必要な IAM ユーザーまたはグループにそのポリシーをアタッチする必要があります。

JSON ポリシードキュメントのこれらの例を使用して、IAM アイデンティティベースポリシーを作成する方法については、「IAM ユーザーガイド」の「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーのベストプラクティス
- AWS Payment Cryptography コンソールの使用
- ユーザーが自分の許可を表示できるようにする
- AWS Payment Cryptography のすべての側面にアクセスできる
- 指定したキーを使用して API を呼び出すことができます。
- リソースを具体的に拒否できる機能

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰かが AWS Payment Cryptography リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義して、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイドの「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して権限を適用する方法の詳細については、『IAM ユーザーガイド』の「IAM でのポリシーと権限」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、 などの特定の を介してサービスアクションが使用される場合に AWS のサービス、サービスアクションへのアクセスを許可

することもできます AWS CloudFormation。詳細については、IAM ユーザーガイドの [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。

- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

AWS Payment Cryptography コンソールの使用

AWS Payment Cryptography コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 AWS アカウントの AWS Payment Cryptography リソースの詳細を一覧表示および表示できます。最小限必要な許可よりも厳しく制限されたアイデンティティベースポリシーを作成すると、そのポリシーを添付したエンティティ (IAM ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

これらのエンティティが AWS Payment Cryptography コンソールを引き続き使用できるようにするには、エンティティに次の AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーが自分の許可を表示できるようにする

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Payment Cryptography のすべての側面にアクセスできる

> **Warning**
>
> この例では幅広い許可が付与されるため、お勧めしません。代わりに最小の許可が付与されるアクセスモデルを検討してください。

この例では、 AWS アカウントの IAM ユーザーにすべての AWS Payment Cryptography キーへのアクセス権を付与し、 ControlPlane および DataPlane オペレーションを含むすべての AWS Payment Cryptography API を呼び出す機能を付与します。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
            "Effect": "Allow",
            "Action": [
               "payment-cryptography:*"
            ],
            "Resource": [
               "*"
            ]
         }
   ]
}
```

指定したキーを使用して API を呼び出すことができます。

この例では、 AWS アカウントの IAM ユーザーに AWS Payment Cryptography キーの 1 つへのアクセス権を付与`arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`し、このリソースを 2 つの APIs、、`GenerateCardData`および で使用しますVerifyCardData。逆に、IAM ユーザーには、`DeleteKey` や `ExportKey` などの他のオペレーションでこのキーを使用するアクセス権がありません。

リソースは、`key` というプレフィックスを付けたキー、`alias` というプレフィックスが付いたエイリアスのどちらでもかまいません。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
            "Effect": "Allow",
            "Action": [
               "payment-cryptography:VerifyCardData",
               "payment-cryptography:GenerateCardData"
            ],
```

```
                "Resource": [
                    "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
                ]
            }
        ]
    }
```

リソースを具体的に拒否できる機能

> **Warning**
>
> ワイルドカードアクセスを許可することで出る影響を慎重に検討してください。代わりに最小特権モデルを検討してください。

この例では、 AWS アカウント内の IAM ユーザーに AWS Payment Cryptography キーへのアクセスを許可するが、1 つの特定のキーへのアクセス許可を拒否したいとします。ユーザーは、拒否ステートメントで指定されているものを除くすべてのキーを使用して、`VerifyCardData` および `GenerateCardData` にアクセスできます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "payment-cryptography:VerifyCardData",
                "payment-cryptography:GenerateCardData"
            ],
            "Resource": [
                "arn:aws:payment-cryptography:us-east-2:111122223333:key/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "payment-cryptography:GenerateCardData"
            ],
            "Resource": [
```

```
                    "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
                ]
            }
        ]
    }
```

AWS Payment Cryptography のアイデンティティとアクセスのトラブルシューティング

AWS Payment Cryptography に固有の IAM 関連の問題が特定されると、このセクションにトピックが追加されます。IAM トピックに関する一般的なトラブルシューティングコンテンツについては、「IAM ユーザーガイド」の「トラブルシューティングセクション」を参照してください。

AWS Payment Cryptography のモニタリング

モニタリングは、AWS Payment Cryptography および他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要です。AWS には、AWS をモニタリングして異常を検出した場合に報告し、必要に応じて自動的に対処するために、次のモニタリングツールが用意されています。

- Amazon CloudWatch は、AWS のリソースおよび AWS で実行しているアプリケーションをリアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon CloudWatch Logs では、Amazon EC2 インスタンス、CloudTrail、その他ソースから得たログファイルのモニタリング、保存、およびアクセスが可能です。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。
- Amazon EventBridge を使用して、AWS のサービスを自動化し、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに自動的に対応できます。AWS のサービスからのイベントは、ほぼリアルタイムに EventBridge に提供されます。簡単なルールを記述して、注目するイベントと、イベントがルールに一致した場合に自動的に実行するアクションを指定できます。詳細については、Amazon EventBridge ユーザーガイドを参照してください。
- AWS CloudTrail は、AWSアカウントにより、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWS を呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出しの発生日時を特定できます。詳細については、AWS CloudTrailユーザーガイドを参照してください。

> **Note**
>
> AWS CloudTrail ログは CreateKey などのコントロールプレーンオペレーションではサポートされますが、Generate Card Data などのデータプレーンオペレーションではサポートされません。

AWS を使用した Payment Cryptography API 呼び出しのロギング AWS CloudTrail

AWS Payment Cryptographyは、ユーザー、ロール、または AWS Payment Cryptography 内にある AWS サービスによって実行されたアクションを記録するためのサービスである AWS CloudTrail と統合されています。CloudTrail は、AWS Payment Cryptography のすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、AWS Payment Cryptography コンソールからの呼び出しと、AWS Payment Cryptography API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、AWS Payment Cryptography のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的配信を有効にすることができます。追跡を設定しない場合でも、CloudTrail コンソールの [Event history] (イベント履歴) で最新のイベントを表示できます。CloudTrail が収集した情報を使用して、AWS Payment Cryptography に対して行われたリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時、および追加の詳細情報を確認できます。

CloudTrail の詳細については、AWS CloudTrail ユーザーガイドを参照してください。

> **Note**
>
> Cloudrail 統合は現在、コントロールプレーンのオペレーションでのみサポートされています。

CloudTrail 内の AWS Payment Cryptography 情報

AWS アカウントを作成すると、そのアカウントに対して CloudTrail が有効になります。AWS Payment Cryptography でアクティビティが発生すると、そのアクティビティは [Event history] でその他の AWS サービスのイベントとともに CloudTrail イベントにレコードされます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「Viewing Events with CloudTrail Event History」(CloudTrail イベント履歴でのイベントの表示) を参照してください。

AWS Payment Cryptographyのイベントを含む、AWS アカウントのイベントの継続的な記録については、証跡を作成します。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで追跡を作成するときに、追跡がすべての AWS リージョンに適用されます。追跡は、AWSパーティションのすべてのリージョンからのイベントを口グに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- 「追跡を作成するための概要」
- CloudTrail がサポートされているサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- 複数のリージョンからの CloudTrail ログファイルの受信
- 複数のアカウントからの CloudTrail ログファイルの受信

CloudTrail は、CreateKey、ImportKey、DeleteKey 、ListKeys 、TagResource、その他すべてのコントロールプレーンオペレーションなどの AWS Payment Cryptography オペレーションを記録します。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが、ルート認証情報と AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

AWS Payment Cryptography のログファイルエントリについて

「トレイル」は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルには、単一か複数のログエントリがあります。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

AWS Payment Cryptography `CreateKey` アクションを示す CloudTrail ログエントリの例を次に示します。

```
    {
  CloudTrailEvent: {
    tlsDetails= {
```

```
        TlsDetails: {
          cipherSuite=TLS_AES_128_GCM_SHA256,
          tlsVersion=TLSv1.3,
          clientProvidedHostHeader=pdx80.controlplane.paymentcryptography.us-
west-2.amazonaws.com
        }
      },
    requestParameters=CreateKeyInput (
      keyAttributes=KeyAttributes(
        KeyUsage=TR31_B0_BASE_DERIVATION_KEY,
        keyClass=SYMMETRIC_KEY,
        keyAlgorithm=AES_128,
        keyModesOfUse=KeyModesOfUse(
          encrypt=false,
          decrypt=false,
          wrap=false
          unwrap=false,
          generate=false,
          sign=false,
          verify=false,
          deriveKey=true,
          noRestrictions=false)
        ),
      keyCheckValueAlgorithm=null,
      exportable=true,
      enabled=true,
      tags=null),
    eventName=CreateKey,
    userAgent=Coral/Apache-HttpClient5,
    responseElements=CreateKeyOutput(
      key=Key(
        keyArn=arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp,
          keyAttributes=KeyAttributes(
            KeyUsage=TR31_B0_BASE_DERIVATION_KEY,
            keyClass=SYMMETRIC_KEY,
            keyAlgorithm=AES_128,
            keyModesOfUse=KeyModesOfUse(
              encrypt=false,
              decrypt=false,
              wrap=false,
              unwrap=false,
              generate=false,
              sign=false,
```

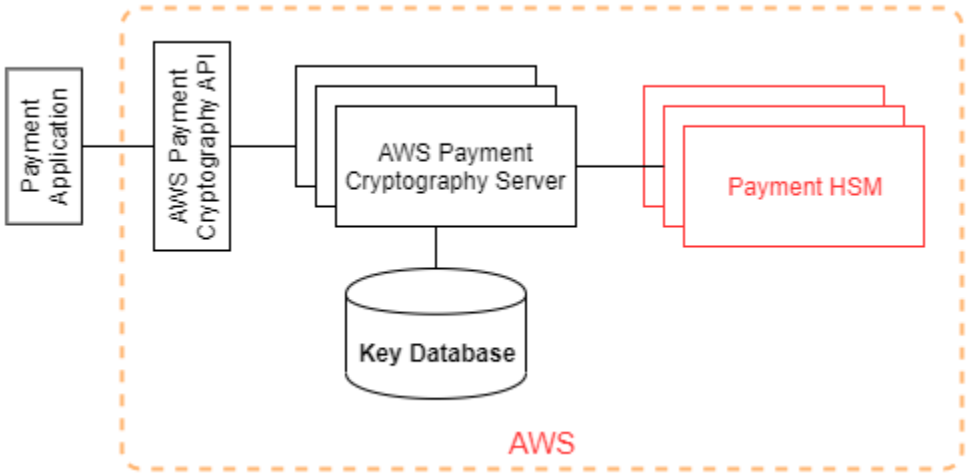
```
              verify=false,
              deriveKey=true,
              noRestrictions=false)
            ),
          keyCheckValue=FE23D3,
          keyCheckValueAlgorithm=ANSI_X9_24,
          enabled=true,
          exportable=true,
          keyState=CREATE_COMPLETE,
          keyOrigin=AWS_PAYMENT_CRYPTOGRAPHY,
          createTimestamp=Sun May 21 18:58:32 UTC 2023,
          usageStartTimestamp=Sun May 21 18:58:32 UTC 2023,
          usageStopTimestamp=null,
          deletePendingTimestamp=null,
          deleteTimestamp=null)
        ),
      sourceIPAddress=192.158.1.38,
        userIdentity={
          UserIdentity: {
            arn=arn:aws:sts::111122223333:assumed-role/TestAssumeRole-us-west-2-PDX80/
ControlPlane-IntegTest-68211a2a-3e9d-42b7-86ac-c682520e0410,
            invokedBy=null,
            accessKeyId=,
            type=AssumedRole,
            sessionContext={
              SessionContext: {
                sessionIssuer={
                  SessionIssuer: {arn=arn:aws:iam::111122223333:role/TestAssumeRole-us-
west-2-PDX80,
                  type=Role,
                  accountId=111122223333,
                  userName=TestAssumeRole-us-west-2-PDX80,
                  principalId=}
                },
                attributes={
                  SessionContextAttributes: {
                    creationDate=Sun May 21 18:58:31 UTC 2023,
                    mfaAuthenticated=false
                  }
                },
                webIdFederationData=null
              }
            },
          username=null,
```

```
          principalId=:ControlPlane-User,
          accountId=111122223333,
          identityProvider=null
        }
      },
      eventTime=Sun May 21 18:58:32 UTC 2023,
      managementEvent=true,
      recipientAccountId=111122223333,
      awsRegion=us-west-2,
      requestID=151cdd67-4321-1234-9999-dce10d45c92e,
      eventVersion=1.08, eventType=AwsApiCall,
      readOnly=false,
      eventID=c69e3101-eac2-1b4d-b942-019919ad2faf,
      eventSource=payment-cryptography.amazonaws.com,
      eventCategory=Management,
      additionalEventData={
    }
  }
}
```

暗号化の詳細

AWS Payment Cryptography は、支払い取引の暗号キーを生成および管理するためのウェブインターフェースを提供します。AWSPayment Cryptography は、標準的なキー管理サービス、支払い取引の暗号化、および集中管理と監査に使用できるツールを提供します。このホワイトペーパーは、AWS Payment Cryptography の暗号化オペレーションを詳細に説明ししており、このサービスが提供する機能の評価に役立ちます。

AWS Payment Cryptography には、分散された PCI PTS HSM 検証済みハードウェアセキュリティモジュールの暗号化オペレーションをリクエストするための複数のインターフェイス (AWS CLI、AWS SDK、AWS Management Console などによる RESTful API を含む) が含まれています。



AWS Payment Cryptography は、ウェブとの接点である AWS Payment Cryptography 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 Payment Cryptography スタックを形成します。AWS Payment Cryptographyへのすべてのリクエストは、Transport Layer Security (TLS) プロトコルを経由して AWS Payment Cryptography ホストで行われる必要があります。サービスホストは、完全な前方秘匿を実現する暗号スイートの TLS のみを許可します。このサービスは、他のすべての API オペレーションで使用できる AWS IAM と同じ認証情報とポリシーの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。

AWS Payment Cryptographyサーバーは、非仮想のプライベートネットワークを経由して基盤となる HSM に接続します。サービスコンポーネントと HSM 間の接続は、認証と暗号化のための連携 TLS (mTLS) で保護されます。

設計目標

AWS Payment Cryptography は、次の要件を満たすように設計されています。

- 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 AWS Payment Cryptography 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。Amazon の従業員は HSM のメイン (またはマスター) キーやバックアップにアクセスできません。メインキーは、AWS Payment Cryptography リージョンに含まれていない HSM と同期することはできません。その他のキーはすべて HSM メインキーによって保護されています。そのため、お客様の AWS Payment Cryptography キーは、お客様のアカウント内で動作する AWS Payment Cryptography サービス以外では使用できません。
- 低遅延かつ高スループット — AWS Payment Cryptography は、Payment Cryptography キーの管理や支払い取引の処理に適した、遅延とスループットレベルの暗号化オペレーションを提供します。
- 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 つの暗号キーを、決済ターミナル、EMV チップカード、または長年使用されているその他の安全な暗号デバイス (SCD) と共有できます。
- リージョンの独立性 — 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や、データレジデンシー要件に準拠する必要があるお客さま向けに、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。
- 乱数の安全なソース — 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 Payment Cryptography 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。AWS Payment Cryptography のキー生成はすべて PCI PTS HSM にリストされている HSM を使用しており、PCI モードで動作します。
- 監査 — AWS Payment Cryptography は、では Amazon CloudWatch で入手できる CloudTrail ログとサービスログで、暗号化キーの使用と管理が CloudTrail ログに記録されます。では、暗号化キーの使用状況を調べることができます。これには、キーを共有しているアカウントによる使用も含まれます。AWSPayment Cryptography は、該当する PCI、カードブランド、および地域の支払いセキュリティ基準に照らして、サードパーティーの評価者によって監査されます。証明書と責任分担ガイドは AWS Artifact でご覧いただけます。
- 伸縮性 — AWS Payment Cryptography はお客様の要求に応じてスケールアウトやスケールインを行います。AWS Payment Cryptographyは、HSM の容量を予測して確保する代わりに、オンデマンドで決済の暗号化を行います。AWSPayment Cryptography は、HSM のセキュリティとコンプライアンスを維持し、顧客のピーク需要を満たすのに十分な容量を提供する責任があります。

基礎

この章のトピックでは、 AWS Payment Cryptography の暗号化プリミティブとその使用場所について説明します。また、 そのサービス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 Payment Cryptography は、パラメータ可能な標準の暗号化アルゴリズムを使用するため、アプリケーションはユースケースに必要なアルゴリズムを実装できます。一連の暗号アルゴリズムは PCI、ANSI X9、EMVCo、および ISO 規格によって定義されています。すべての暗号化は、PCI モードで動作する PCI PTS HSM 標準規格に登録されている HSM によって実行されます。

エントロピーと乱数生成

AWS Payment Cryptography キーの生成は AWS Payment Cryptography HSMsで実行されます。HSM には、サポートされているすべてのキータイプとパラメーターの PCI PTS HSM 要件を満たす乱数ジェネレーターが実装されています。

対称キーのオペレーション

ANSI X9 TR 31、ANSI X9.24、および PCI PIN Annex C で定義されている対称キーアルゴリズムとキーストレングスは、以下でサポートされています。

- ハッシュ関数 — 出力サイズが 2551 を超える SHA2 および SHA3 ファミリーのアルゴリズム。ただし、PCI PTS POI v3 以前のターミナルとの下位互換性は除きます。
- 暗号化と復号化 — キーサイズが 128 ビット以上の AES、または 112 ビット以上のキーサイズ (2 キーまたは 3 キー) の TDEA。
- メッセージ認証コード (MAC): AES を使用する CMAC または GMAC、および承認済みのハッシュ関数を使用し、キーサイズが 128 以上の HMAC。

AWS Payment Cryptography は、HSM メインキー、データ保護キー、TLS セッションキーに AES 256 を使用します。

非対称キーのオペレーション

ANSI X9 TR 31、ANSI X9.24、および PCI PIN Annex C で定義されている非対称キーアルゴリズムとキーストレングスは、以下でサポートされています。

- 承認されたキー確立スキーム — NIST SP800-56A (ECC/FCC2 ベースのキーアグリーメント)、NIST SP800-56B (IFC ベースのキーアグリーメント)、および NIST SP800-38F (AES ベースのキー暗号化/ラッピング) で説明されているとおり。

AWS Payment Cryptography ホストは、Perfect Forward Secrecy を提供する暗号スイートで TLS を使用するサービスへの接続のみを許可します。

キーの保管

AWS Payment Cryptography キーは HSM AES 256 メインキーによって保護され、暗号化されたデータベースの ANSI X9 TR 31 キーブロックに保存されます。データベースは Payment Cryptography AWS サーバーのインメモリデータベースにレプリケートされます。

PCI PIN セキュリティ規範附属書 C によると、AES 256 キーは以下と同等かそれ以上の強度があります。

- 3 キー TDEA
- RSA 15360 ビット
- ECC 512 ビット
- DSA、DH、および MQV 15360/512

対称キーを使用したキーインポート

AWS Payment Cryptography は、インポート用に保護されたキーと同等または強力な対称キー暗号化キー (KEK) を持つ対称キーまたはパブリックキーを持つ暗号文とキーブロックのインポートをサポートします。

非対称キーを使用したキーのインポート

AWS Payment Cryptography は、インポート用に保護されたキーと同等または強力なプライベートキー暗号化キー (KEK) で保護された対称キーまたはパブリックキーを持つ暗号文とキーブロックのインポートをサポートします。復号用に提供される公開キーは、顧客が信頼する機関からの証明書によって信頼性と完全性が保証されている必要があります。

AWS Payment Cryptography が提供するパブリック KEK は、PCI PIN Security および PCI P2PE Annex A への準拠が証明された認証機関 (CA) の認証と整合性保護を備えています。

キーエクスポート

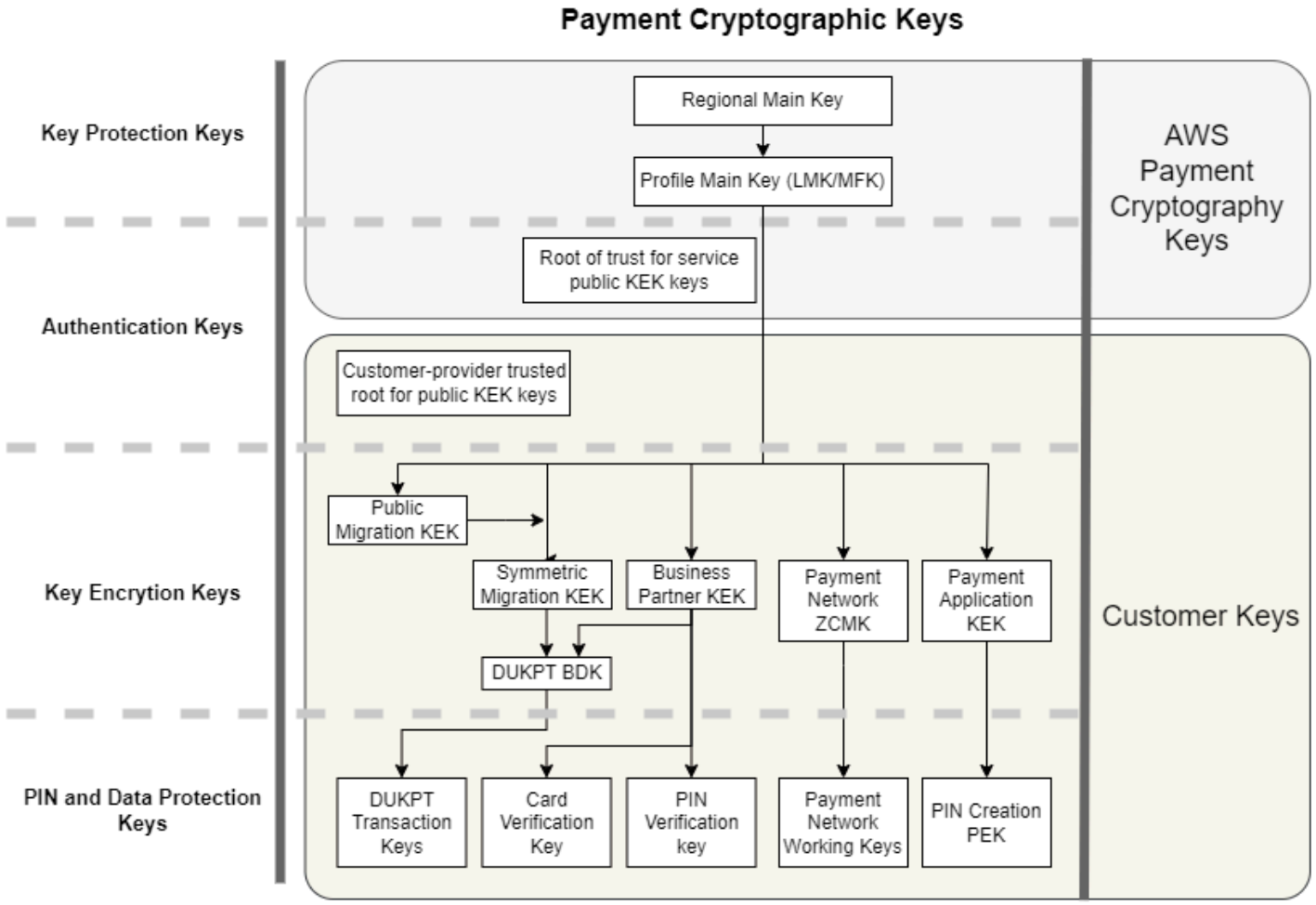
キーは、エクスポートするキーと同じかそれ以上の強度を持つ適切な を持つキーでエクスポート KeyUsage および保護できます。

トランザクション単位の派生ユニークキー (DUKPT) プロトコル

AWS Payment Cryptography は、ANSI X9.24-3 で説明されているように、TDEA および AES ベース導出キー (BDK) で をサポートします。

キー階層

AWS Payment Cryptography のキー階層により、キーは常に保護するキーと同じかそれ以上の強度のキーで保護されます。



AWS Payment Cryptography キーは、サービス内のキー保護に使用されます。

キー	説明
リージョナルメインキー	暗号処理に使用される仮想 HSM イメージまたはプロファイルを保護します。このキーは HSM と安全なバックアップにのみ存在します。
プロファイルメインキー	最上位のカスタマーキー保護キーのことで、これまではカスタマーキーのローカルマスターキー (LMK) またはマスターファイルキー (MFK) と呼ばれていました。このキーは HSM と安全なバックアップにのみ存在します。プロファイルは、決済ユースケースのセキュリティ

キー	説明
	標準で要求される個別の HSM 設定を定義します。
AWS Payment Cryptography パブリックキー暗号化キー (KEK) の信頼ルート	Payment AWS Cryptography が提供する非対称キーを使用したキーのインポートとエクスポート用のパブリックキーを認証および検証するための信頼できるルートパブリックキーと証明書。

カスタマーキーは、他のキーを保護するためのキーと支払い関連データを保護するキーによってグループ分けされています。両タイプのカスタマーキーの例を以下に示します。

キー	説明
お客様提供の公開 KEK キー用の信頼できるルート	非対称キーを使用してキーをインポートおよびエクスポートする際に提供する公開キーを認証および検証するための信頼のルートとして、お客様から提供された公開キーと証明書。
キー暗号化キー (KEK)	KEK は、外部キーストアと AWS Payment Cryptography、ビジネスパートナー、支払いネットワーク、または組織内のさまざまなアプリケーション間の交換のために、他のキーを暗号化するためにのみ使用されます。
トランザクション単位の派生ユニークキー (DUKPT) 基本派生キー (BDK)	BDK は、決済ターミナルごとに固有のキーを作成し、複数のターミナルからのトランザクションを 1 つの承継銀行または買収者のワーキングキーに変換するために使用されます。PCI Point-to-Point Encryption (P2PE) で求められるベストプラクティスは、ターミナルモデル、キーインジェクション、初期化サービス、その他のセグメンテーションごとに異なる BDK を使用し、BDK の侵害による影響を最小限に抑えることです。

キー	説明
決済ネットワークゾーンコントロールマスターキー (ZCMK)	ZCMK はゾーンキーまたはゾーンマスターキーとも呼ばれ、決済ネットワークから初期動作キーを確立するために提供されます。
DUKPT トランザクションキー	DUKPT 用に設定された決済ターミナルは、ターミナルとトランザクションに固有のキーを導出します。トランザクションを受信する HSM は、ターミナル ID とトランザクションシーケンス番号からキーを決定できます。
カードデータ準備キー	EMV 発行者マスターキー、EMV カードキーと検証値、カードパーソナライゼーションデータファイル保護キーを使用して、カードパーソナライゼーションプロバイダーが使用する個々のカードのデータを作成します。これらのキーと暗号検証データは、発行銀行や発行会社が、取引の承認の一環としてカードデータを認証するためにも使用されます。
カードデータ準備キー	EMV 発行者マスターキー、EMV カードキーと検証値、カードパーソナライゼーションデータファイル保護キーを使用して、カードパーソナライゼーションプロバイダーが使用する個々のカードのデータを作成します。これらのキーと暗号検証データは、発行銀行や発行会社が、取引の承認の一環としてカードデータを認証するためにも使用されます。

キー	説明
決済ネットワークのワーキングキー	発行者ワーキングキーまたはアクワイアラーワーキングキーとも呼ばれ、決済ネットワークとの間で送受信されるトランザクションを暗号化するキーです。これらのキーは、ネットワークによって頻繁に (多くの場合、毎日、または 1 時間おきに) ローテーションされます。PIN / デビット取引用の PIN 暗号化キー (PEK) となります。
個人識別番号 (PIN) 暗号化キー (PEK)	PIN ブロックを作成または復号化するアプリケーションでは、PEK を使用してクリアテキスト PIN の保存や送信を防止します。

内部オペレーション

このトピックでは、グローバルに分散されたスケーラブルな Payment Cryptography およびキー管理サービスのカスタマーキーと暗号化オペレーションを保護するためにサービスが実装する内部要件について説明します。

HSM の仕様とライフサイクル

AWS Payment Cryptography は、市販の HSM フリートを使用します。HSM は FIPS 140-2 レベル 3 認証を受けており、PCI セキュリティ標準審議会が承認した PCI PTS デバイスリストに PCI HSM v3 準拠として記載されているファームウェアバージョンとセキュリティポリシーも使用しています。PCI PTS HSM 標準には HSM ハードウェアの製造、出荷、デプロイ、管理、および廃棄に関する追加要件が含まれています。これらの要件は、支払いのセキュリティとコンプライアンスにとって重要ですが、FIPS 140 では対応されていません。

すべての HSM は PCI モードで運用され、PCI PTS HSM セキュリティポリシーに基づいて設定されます。 AWS Payment Cryptography のユースケースをサポートするために必要な関数のみが有効になっています。 AWS Payment Cryptography では、クリアテキスト PINsの印刷、表示、または返却を行うことはできません。

HSM デバイスの物理的セキュリティ

サービスが使用できるのは、配送前に製造元が AWS Payment Cryptography 認証機関 (CA) によってデバイスキーに署名した HSMs のみです。 AWS Payment Cryptography は、HSM 製造元とデバイス証明書の信頼のルートである製造元の CA のサブ CA です。製造元の CA は ANSI TR 34 を実装しており、PCI PIN Security Annex A および PCI P2PE Annex A への準拠が証明されています。製造元は、 AWS Payment Cryptography CA によって署名されたデバイスキーを持つすべての HSM が AWS の指定された受信者に出荷されていることを確認します。

PCI PIN Security の要求に従い、製造元は HSM の出荷とは異なる通信チャネルを介してシリアル番号のリストを提供します。これらのシリアル番号は、HSM を AWS データセンターにインストールするプロセスの各ステップで確認されます。最後に、 AWS Payment Cryptography オペレーターは、インストールされている HSM のリストを製造元のリストと照合して検証してから、シリアル番号を AWS Payment Cryptography キーの受信が許可されている HSM のリストに追加します。

HSM は常に安全な保管場所に保管されるか、以下を含む二重管理下に置かれます。

- メーカーから AWS ラックアセンブリ施設への出荷。
- ラックの組み立て中。
- ラックアセンブリ施設からデータセンターへの出荷。
- 受領およびデータセンターの安全な処理室への設置。HSM ラックは、カードアクセス制御ロック、アラーム付きドアセンサー、カメラによる二重制御を実現します。
- オペレーション中。
- 廃止作業中および廃棄中。

個々の説明責任 chain-of-custodyを持つ完全な は、HSM ごとに維持および監視されます。

HSM の初期化

HSM は、その ID と整合性がシリアル番号、製造元がインストールしたデバイスキー、ファームウェアチェックサムによって検証された後にのみ、 AWS Payment Cryptography フリートの一部として初期化されます。HSM の信頼性と整合性が検証されたら、PCI モードの有効化を含む設定が行われます。その後、 AWS Payment Cryptography リージョンのメインキーとプロファイルのメインキーが確立され、HSM がサービスで使用できるようになります。

HSM のサービスと修理

HSM には、デバイスの暗号境界に違反する必要のない保守可能なコンポーネントがあります。これらのコンポーネントには、冷却ファン、電源、バッテリーが含まれます。HSM または HSM ラック内の別のデバイスの修理が必要な場合、ラックが開いている間ずっとデュアルコントロールが維持されます。

HSM の廃止作業

廃止は、HSM の end-of-life または の障害が原因で発生します。HSM は、機能していればラックから取り出す前に論理的にゼロ化され、AWS データセンターの安全な処理室内で廃棄されます。廃棄される前に、修理のためにメーカーに返却されたり、別の目的で使用されたり、安全な処理室から持ち出されたりすることはありません。

HSM ファームウェアの更新

HSM ファームウェアの更新は、PCI PTS HSM および FIPS 140-2 (または FIPS 140-3) にリストされているバージョンとの整合性を維持するために必要な場合、更新がセキュリティ関連である場合、またはお客様が新しいバージョンの機能から利点を享受できると判断した場合に適用されます。AWS Payment Cryptography HSMs、PCI PTS HSM リストのバージョンと一致する off-the-shelf ファームウェアを実行します。新しいファームウェアバージョンは PCI または FIPS 認定ファームウェアバージョンとの整合性が検証され、機能性テストを経てすべての HSM に展開されます。

オペレーターアクセス

オペレーターは、ごくまれに、通常の運用中に HSM から収集された情報では問題を特定したり変更を計画したりするには不十分であるというトラブルシューティングに、コンソール以外でも HSM にアクセスできます。以下のステップが実行されます。

- トラブルシューティングアクティビティが作成され承認され、コンソール以外のセッションがスケジュールされます。
- HSM は顧客処理サービスから削除されます。
- 主キーは二重制御下で削除されます。
- オペレーターは、コンソール以外でも HSM にアクセスして、承認されたトラブルシューティング作業を、デュアルコントロールのもとで実行できます。
  - コンソール以外のセッションが終了すると、HSM で初期プロビジョニング処理が実行され、標準ファームウェアと設定が戻され、メインキーが同期されてから HSM がサービスを受けている顧客に返却されます。

- セッションの記録は変更追跡に記録されます。
- セッションから得られた情報は、future 変更を計画するために使用されます。

コンソール以外のアクセスレコードはすべて、プロセスのコンプライアンスと、HSM モニタリング、 non-console-access 管理プロセス、またはオペレータートレーニングへの潜在的な変更についてレビューされます。

キー管理

リージョン内のすべての HSM はリージョンメインキーと同期されます。リージョンメインキーは少なくとも 1 つのプロファイルメインキーを保護します。プロファイルメインキーはカスタマーキーを保護します。

すべてのメインキーは HSM によって生成され、ANSI X9 TR 34 および PCI PIN Annex A に沿った非対称技術を使用した対称キー分散によって配布されます。

トピック

- [Generation] (生成)
- リージョンメインキー同期
- リージョンメインキーのローテーション
- プロファイルメインキーの同期
- プロファイルメインキーのローテーション
- 保護
- 耐久性
- 通信セキュリティ
- カスタマーキーの管理
- ロギングとモニタリング

[Generation] (生成)

AES 256 ビットのメインキーは、PCI PTS HSM 乱数ジェネレーターを使用して、サービス HSM フリートにプロビジョニングされた HSM の 1 つで生成されます。

リージョンメインキー同期

HSM リージョンメインキーは、ANSI X9 TR-34 で定義されている次のようなメカニズムを使用して、リージョナルフリート全体のサービスによって同期されます。

- キー分散ホスト (KDH) とキー受信デバイス (KRD) のキーと証明書を使用する連携認証により、公開キーの認証と整合性を確保します。
- 証明書は PCI PIN Annex A2 の要件を満たす認証局 (CA) によって署名されますが、AES 256 ビットキーの保護に適した非対称アルゴリズムとキー強度は除きます。
- 分散型対称キーの識別とキー保護は ANSI X9 TR-34 および PCI PIN Annex A1 と同じです。ただし、AES 256 ビットキーの保護に適した非対称アルゴリズムとキー強度は除きます。

リージョンメインキーは、以下の方法でリージョンの認証とプロビジョニングが行われた HSM に対して設定されます。

- メインキーはリージョンの HSM で生成されます。その HSM はキー分散ホストとして指定されます。
- リージョン内のプロビジョニングされたすべての HSM は、HSM のパブリックキーと再生不可能な認証情報を含む KRD 認証トークンを生成します。
- KRD トークンは、KDH が HSM の ID とキーを受け取る許可を検証した後に KDH 許可リストに追加されます。
- KDH は HSM ごとに認証可能なメインキートークンを生成します。トークンには KDH 認証情報と、作成対象の HSM にのみロード可能な暗号化されたメインキーが含まれています。
- 各 HSM には、その HSM 用に構築されたメインキートークンが送信されます。HSM 自身の認証情報と KDH 認証情報を検証した後、メインキーは KRD プライベートキーによって復号化され、メインキーにロードされます。

1 つの HSM をリージョンと再同期する必要がある場合は以下を参照してください。

- ファームウェアと設定を使用して再検証され、プロビジョニングされます。
- そのリージョンで初めて導入された場合は以下を参照してください。
  - HSM は KRD 認証トークンを生成します。
  - KDH はトークンを許可リストに追加します。
  - KDH は HSM のメインキートークンを生成します。
  - HSM はメインキーをロードします。

- HSM がサービスで利用可能になります。

これにより、次のことが保証されます。

- AWS Payment Cryptography 処理用にリージョン内で検証された HSM のみが、そのリージョンのマスターキーを受け取ることができます。
- AWS Payment Cryptography HSM のマスターキーのみをフリートの HSM に配布できます。

リージョンメインキーのローテーション

リージョンメインキーは、暗号期間の満了時、万が一、キーの侵害が疑われる場合、またはキーのセキュリティに影響があると判断されたサービスの変更後にローテーションされます。

最初のプロビジョニングと同様に、新しいリージョンメインキーが生成され、配布されます。保存したプロファイルメインキーは、新しいリージョンメインキーに変換する必要があります。

リージョンメインキーローテーションは顧客処理には影響しません。

プロファイルメインキーの同期

プロファイルメインキーはリージョンメインキーによって保護されます。これにより、プロファイルは特定の地域に制限されます。

プロファイルメインキーは、それに応じてプロビジョニングされます。

- プロファイルメインキーは、リージョンメインキーが同期された HSM で生成されます。
- プロファイルメインキーは、プロファイル設定やその他のコンテキストで保存され、暗号化されます。
- このプロファイルは、リージョンメインキーを持つリージョン内のすべての HSM が顧客の暗号化機能に使用されます。

プロファイルメインキーのローテーション

プロファイルメインキーは、暗号期間の満了時、キーの侵害が疑われるとき、またはキーのセキュリティに影響すると判断されたサービスの変更後にローテーションされます。

ローテーション手順は以下の通りです。

- 初期プロビジョニングと同様に、新しいプロファイルメインキーが生成され、保留中のメインキーとして配布されます。
- バックグラウンドプロセスにより、カスタマーキーマテリアルが確立されたプロファイルメインキーから保留中のメインキーに変換されます。
- すべてのカスタマーキーが保留中のキーで暗号化されると、保留中のキーはプロファイルメインキーに昇格します。
- バックグラウンド処理により、期限切れのキーで保護されているカスタマーキー情報が削除されます。

プロファイルメインキーローテーションは顧客処理には影響しません。

保護

キーはキー階層にのみ依存して保護されます。主なキーを保護することは、すべてのカスタマーキーの紛失や漏洩を防ぐために重要です。

リージョンメインキーは、HSM で認証され、サービス用にプロビジョニングされた場合にのみバックアップから復元できます。これらのキーは、特定の HSM の特定の KDH からの連携認証可能な暗号化されたメインキートークンとしてのみ保存できます。

プロファイルマスターキーは、地域ごとに暗号化されたプロファイル設定とコンテキスト情報とともに保存されます。

カスタマーキーはキーブロックに保存され、プロファイルマスターキーで保護されます。

すべてのキーは HSM 内にのみ存在するか、同等かそれ以上の暗号強度を持つ別のキーで保護されて保管されます。

耐久性

トランザクション暗号化とビジネス機能のカスタマーキーは、通常は停止を引き起こすような極端な状況でも利用できる必要があります。 AWS Payment Cryptography は、アベイラビリティーゾーンと AWS リージョンにまたがる複数レベルの冗長モデルを利用します。Payment Cryptography オペレーションについて、サービスが提供するものよりも高い可用性と耐久性を求めるお客様は、マルチリージョンアーキテクチャを実装する必要があります。

HSM 認証とメインキートークンは保存され、HSM をリセットする必要がある場合にメインキーを復元したり、新しいメインキーと同期したりするために使用できます。トークンはアーカイブされ、必要に応じて二重制御下でのみ使用されます。

通信セキュリティ

外部

AWS Payment Cryptography API エンドポイントは、リクエストの認証と整合性に関する 1.2 以上の TLS や署名バージョン 4 などの AWS セキュリティ標準を満たしています。

受信 TLS 接続はNetwork Load Balancerで終了し、内部 TLS 接続を介して API ハンドラーに転送されます。

内部

サービスコンポーネント間、およびサービスコンポーネントと他の AWS サービス間の内部通信は、強力な暗号化を使用する TLS によって保護されます。

HSM は、サービスコンポーネントからのみアクセスできる非仮想プライベートネットワーク上にあります。HSM とサービスコンポーネント間のすべての接続は、TLS 1.2 以上の連携 TLS (mTLS) で保護されています。TLS と mTLS の内部証明書は、AWS プライベート認証局を使用して Amazon Certificate Manager によって管理されます。内部 VPC と HSM ネットワークは、予期しないアクティビティや設定変更がないか監視されます。

カスタマーキーの管理

では AWS、お客様の信頼が最優先事項です。お客様の AWS アカウントでサービスにアップロードまたは作成したキーを完全に管理し、キーへのアクセスを設定する責任はお客様にあります。

AWS Payment Cryptography は、サービスによって管理されるキーの HSM の物理的なコンプライアンスとキー管理について全責任を負います。これには、HSM メインキーの所有権と管理、および AWS Payment Cryptography キーデータベース内の保護された顧客キーの保存が必要です。

カスタマーキースペースの分離

AWS Payment Cryptography は、キーが別のアカウントと明示的に共有されていない限り、プリンシパルをキーを所有するアカウントに制限するなど、すべてのキー使用に対してキーポリシーを適用します。

バックアップとリカバリ

リージョンのキーとキー情報は、 AWSによって暗号化されたアーカイブにバックアップされます。アーカイブを復元 AWS するには、 によるデュアルコントロールが必要です。

キーブロック

すべてのキーは ANSI X9 TR-31 形式のキーブロックに保存されます。

キーは、 でサポートされている暗号文やその他のキーブロック形式からサービスにインポートできます ImportKey。同様に、キーがエクスポート可能な場合は、キーエクスポートプロファイルでサポートされている他のキーブロック形式または暗号グラムにエクスポートできます。

キーの使用

キーの使用は、サービス KeyUsage によって設定された に制限されます。このサービスは、要求された暗号オペレーションに対して不適切なキーの使用、使用方法、またはアルゴリズムを使用する要求をすべて拒否します。

キー交換関係

PCI PIN セキュリティと PCI P2PE では、PIN を暗号化するキー (それらのキーの共有に使用される KEK を含む) を共有する組織は、それらのキーを他の組織と共有しないことが義務付けられています。対称キーは、同じ組織内を含め、2 者間でのみの共有がベストプラクティスです。これにより、キーの侵害が疑われ、影響を受けたキーの交換を余儀なくされることによる影響を最小限に抑えることができます。

2 者以上の当事者間でキーを共有する必要があるビジネスケースでも、当事者の数は最小限に抑える必要があります。

AWS Payment Cryptography には、これらの要件内でキーの使用を追跡して強制するために使用できるキータグが用意されています。

例えば、サービスプロバイダーと共有されるすべてのキーに「KIF」=「PosStation」を設定することで、異なるキー注入施設の KEK と BDK を識別できます。もう 1 つの例は、支払いネットワークと共有されているキーに「Network」=PayCard「」というタグを付けることです。タグ付けを行うと、アクセス制御を作成したり、キー管理の実施や実証に役立つ監査レポートを作成したりできます。

キー削除

DeleteKey は、お客様が設定可能な期間を過ぎると、データベース内のキーを削除対象としてマークします。この期間が過ぎると、キーは回復不能に削除されます。これは、キーが誤ってまたは悪意を持って削除されるのを防ぐための安全メカニズムです。削除対象としてマークされたキーは、 以外のアクションでは使用できません RestoreKey。

削除したキーは、削除後 7 日間はサービスバックアップに残ります。この期間中、復元することはできません。

閉鎖された AWS アカウントのキーは削除対象としてマークされます。削除期間に達する前にアカウントを再アクティブ化すると、削除対象としてマークされたキーはすべて復元されますが、無効になります。これらのキーを暗号化オペレーションに使用するには、ユーザーが再び有効にする必要があります。

キーの共有

キーは、AWS Resource Access Manager (https://docs.aws.amazon.com/ARG/index.html) を使用して、組織内外の他のアカウントと共有できます。キーはリソース共有にまとめて、アカウントまたはアカウント内の特定の IAM ユーザーやロールと共有できます。リソース共有ごとに使用許可を指定します。共有許可はキーリソースポリシーによって制限されます。共有キーでは、独自のポリシーによって制限されているアクションは許可されません。共有許可はいつでも取り消すことができます。

ロギングとモニタリング

内部サービスログには以下が含まれます。

- CloudTrail サービスによって行われた AWS サービス呼び出しの ログ
- CloudWatch ログまたは HSM からのイベントに直接記録された両方のイベントの CloudWatch ログ
- HSM とサービスシステムからのログファイル
- ログアーカイブ

すべてのログソースは、キーを含む機密情報を監視し、フィルタリングします。ログは体系的に見直され、機密性の高い顧客情報が含まれていないことが確認されます。

ログへのアクセスは、職務遂行に必要な個人に限定されています。

すべてのログは、AWS のログ保持ポリシーに従って保持されます。

顧客オペレーション

AWS Payment Cryptography は、PCI 標準に基づく HSM の物理的コンプライアンスについて全責任を負います。また、このサービスでは安全なキーストアも提供され、PCI 標準で許可され、作成時またはインポート時にユーザーが指定した目的にのみキーを使用できるようにします。サービスのセ

キュリティとコンプライアンス機能を活用するには、キー属性とアクセスを設定する責任があります。

トピック

キーの生成

キーを作成する際には、ポリシーに準拠したキーの使用を強制するためにサービスが使用する属性を設定します。

- アルゴリズムとキーの長さ
- 使用方法
- 有効性と有効期限

属性ベースのアクセス制御 (ABAC) に使用されるタグは、特定のパートナーが使用するキーを制限するために使用されます。また、アプリケーションも作成時に設定する必要があります。タグを削除または変更できるロールを制限するポリシーを必ず含めてください。

キーを作成する前に、キーを使用および管理できるロールを決定するポリシーが設定されていることを確認する必要があります。

> **Note**
>
> CreateKey コマンドの IAM ポリシーを使用して、キー生成の二重制御を強制および実証できます。

キーのインポート

キーをインポートする場合、キーを準拠して使用するための属性は、キーブロック内の暗号的にバインドされた情報を使用してサービスによって設定されます。基本的なキーコンテキストを設定する

メカニズムは、ソース HSM で作成され、共有または非対称 KEK で保護されたキーブロックを使用することです。これは PCI PIN の要件と一致し、ソースアプリケーションの使用方法、アルゴリズム、およびキー強度を維持します。

キーブロック内の情報に加えて、重要なキー属性、タグ、アクセス制御ポリシーをインポート時に設定する必要があります。

暗号文を使用してキーをインポートしても、ソースアプリケーションからキー属性が転送されることはありません。このメカニズムを使用して属性を適切に設定する必要があります。

多くの場合、キーはクリアテキストを使用して交換され、キー管理者が送信した後、安全なルームで二重制御で取り込まれます。これは AWS Payment Cryptography では直接サポートされていません。API は、お客様の HSM でインポートできる証明書付きのパブリックキーを エクスポートして、サービスでインポート可能なキーブロックを エクスポートします。これにより、独自の HSM を使用してクリアテキストコンポーネントを読み込むことができます。

キーチェック値 (KCV) を使用して、インポートされたキーがソースキーと一致することを確認する必要があります。

ImportKey API の IAM ポリシーを使用して、キーのインポートを二重に制御し、示すことができます。

キーを エクスポートする

パートナーやオンプレミスアプリケーションとキーを共有するには、キーの エクスポートが必要な場合があります。 エクスポートにキーブロックを使用すると、暗号化されたキーマテリアルとの基本的なキーコンテキストが維持されます。

キータグを使用すると、同じタグと値を共有するキーの KEK への エクスポートを制限できます。

AWS Payment Cryptography では、クリアテキストのキーコンポーネントは提供または表示されません。そのためには、キーカストディアンが PCI PTS HSM または ISO 13491 でテストされたセキュア暗号デバイス (SCD) に直接アクセスして表示または印刷する必要があります。SCD に非対称 KEK または対称 KEK を設定して、二重制御のもとでクリアテキストキーコンポーネントの作成を行うことができます。

キーチェック値 (KCV) を使用して、宛先 HSM によってインポートされたものがソースキーと一致することを確認する必要があります。

キーの削除

キー削除 API を使用して、設定した一定期間後にキーを削除するようにスケジュールできます。削除前であれば、キーは回復可能です。キーが削除されると、サービスから完全に削除されます。

DeleteKey API の IAM ポリシーを使用して、キー削除の二重制御を適用およびデモンストレーションできます。

キーローテーション

キーローテーションの効果は、キーエイリアスを使用して新しいキーを作成またはインポートし、その新しいキーを参照するようにキーエイリアスを変更することで実装できます。管理方法によっては、古いキーは削除されるか、無効になります。

のクォータ AWS Payment Cryptography

AWS アカウントには、AWS のサービスごとにデフォルトのクォータ (以前は制限と呼ばれていました) があります。特に明記されていない限り、クォータはリージョンごとに存在します。一部のクォータについては引き上げをリクエストできますが、その他のクォータについては引き上げることはできません。

名前	デフォルト	引き上げ可能	説明
エイリアス	サポートされている各リージョン: 2,000	はい	現在のリージョンにおいて、このアカウントで設定できるエイリアスの最大数。
コントロールプレーンリクエストの合計レート	サポートされている各リージョン: 5/秒	はい	現在のリージョンにおいて、このアカウントで実行できる 1 秒あたりのコントロールプレーンリクエストの最大数。このクォータは、すべてのコントロールプレーンオペレーションの合計に適用されます。
データプレーンリクエストの合計レート (非対称)	サポートされている各リージョン: 20/秒	はい	現在のリージョンにおいて、このアカウントで非対称キーで実行できるデータプレーンオペレーションの 1 秒あたりのリクエストの最大数。このクォータは、すべてのデータプレーンオペレー

名前	デフォルト	引き上げ可能	説明
			ションの合計に適用されます。
データプレーンリクエストの合計レート (対称)	サポートされている各リージョン: 500/秒	はい	現在のリージョンにおいて、このアカウントで対称キーで実行できるデータプレーンオペレーションの 1 秒あたりリクエストの最大数。このクォータは、すべてのデータプレーンオペレーションの合計に適用されます。
キー	サポートされている各リージョン: 2,000	はい	現在のリージョンにおいて、このアカウントで設定できるキーの最大数 (削除されたキーを除く)。

AWS Payment Cryptography ユーザーガイドのドキュメント履歴

次の表に、 AWS Payment Cryptography のドキュメントリリースを示します。

変更	説明	日付
機能リリース	VPC endpoints(PrivateLink) と iCVV の例に関する情報を追加します。	2024 年 5 月 8 日
機能リリース	RSA を使用したキーのインポート/エクスポートと DUKPT IPEK/IK キーのエクスポートに関する新機能に関する情報が追加されました。	2024 年 1 月 15 日
初回リリース	AWS Payment Cryptography ユーザーガイドの初回リリース	2023 年 6 月 8 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。