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、専用の支払い HSM インスタンスを調達することなく、支払いカード業界 (PCI) の基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へのアクセスを提供するマネージド AWS サービスです。 AWS Payment Cryptography は、アクワイアラーなどの支払い機能を実行する顧客を提供します。 支払いファシリテーター、 ネットワーク、 スイッチ、 プロセッサ、 および 銀行は、支払い暗号化オペレーションをクラウド内のアプリケーションの近くに移動し、専用支払い HSMs を含む補助データセンターまたはコロケーション施設への依存関係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、高レベルの稼働時間と耐障害性をサポートするように設計されています。AWS Payment Cryptography は完全に伸縮自在で、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 HSMs の運用要件の多くを排除します。 AWS Payment Cryptography では、パートナーとキーを電子的に共有することもできます。これにより、クリアテキストコンポーネントを紙で共有する必要がなくなります。

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 キーを含む対称および非対称 AWS Payment Cryptography キーを作成および管理し、CVV 生成や DUKPT キー取得などの目的を指定します。
- Payment AWS Cryptography キーを自動的に安全に保存し、ハードウェアセキュリティモジュール (HSMs) で保護しながら、ユースケース間でキーを分離します。
- AWS Payment Cryptography キーへのアクセスまたはアクセスの制御に使用できる「フレンドリ名」である エイリアスを作成、削除、一覧表示、および更新します。
- AWS Payment Cryptography キーに識別、グループ化、自動化、アクセスコントロール、コスト追跡のタグを付けます。
- TR-31 (相互運用可能な Secure Key Exchange Key Block Specification) に従って、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ー) の間で対称キーをインポートおよびエクスポートします。

- TR-34 (非対称手法を使用した対称キーの分散方法) などの電子的手段を使用して、非対称キーペアを使用して AWS Payment Cryptography と他のシステム間で対称キー暗号化キー (KEK) をインポートおよびエクスポートします。

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

- 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>
```

以下に例を示します。

```
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、および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、キーエイリアスであるキー 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、KeyOrigin 属性を確認することで判断できます。 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 で指定されているとおり、その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、ARNS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

AWS Payment Cryptography キー

AWS Payment Cryptography キーは 1 つの に存在します AWS リージョン。Payment Cryptography Service AWS に保存されているキーメタデータとマテリアルで構成されます。キーは、TR-31 キーブロックとして外部ソースからインポートすることも、 AWS Payment Cryptography Service によって生成することもできます。

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 (複数可) であり、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。

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) から派生します。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、このキーを使用してデータ暗号化と mac キーを導き出すことができるため、間違っています。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。

KBPK

キーブロック暗号化キー (KBPK) は、キーブロックを保護し、他のキーをラップ/暗号化するために使用される対称キーの一種です。KBPK は KEK と似ていますが、KEK はキーマテリアルを直接保護します。一方、TR-31 および同様のスキームでは、KBPK は作業キーを間接的にのみ保護します。を使用する場合TR-31、TR31_K1_KEY_BLOCK_PROTECTION_KEY は正しいキータイプですが、TR31_K0_KEY_ENCRYPTION_KEY は履歴目的で互換的にサポートされています。

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ためのキーは、通常、TR-31 標準による TR31_K0_KEY_ENCRYPTION_KEY のKeyUsage を有しています。

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。

PGK

PGK (ピン生成キー) は、ピン検証キーの別の名前です。実際にはピンの生成には使用されず (デフォルトでは暗号的に乱数) 、代わりに PVV などの検証値の生成に使用されます。

PRK

プライマリリージョンキーは、レプリケーションが有効になっている特定の Payment Cryptography キーの信頼できるレプリケーションソースです。PRK は、マルチリージョンキーレプリケーション設定のソース Payment Cryptography キーロールへの参照です。Payment Cryptography キーでレプリケーションを有効にすると、その特定のキーレプリケーション設定の PRK と呼ばれます。

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。これはピン生成キーとも呼ばれます。

RRK

レプリカリージョンキーは、PRK から設定されたレプリカに安全にコピーされたレプリケートされたキーマテリアルとメタデータです AWS リージョン。RRK は Payment Cryptography キーの読み取り専用レプリカです。RRK は、マルチリージョンキーレプリケーション設定で特定のキーが果たすロールのリファレンスです。レプリケーション設定を含む主要なメタデータの変更は、PRK に適用する必要があります。

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。

CVV

カード検証値は、従来マグネティックストライプに埋め込まれ、トランザクションの信頼性を検証するために使用される静的シークレット値です。このアルゴリズムは、iCVV、CAVV、CVV2 などの他の目的にも使用されます。他のユースケースでは、この方法で埋め込まれない場合があります。

CVV2

カード検証値 2 は、従来は支払いカードの前面 (または背面) に印刷され、カードを提示しない支払い (電話やオンラインなど) の信頼性を検証するために使用される静的シークレット値です。CVV と同じアルゴリズムを使用しますが、サービスコードは 000 に設定されます。

iCVV

iCVV は CVV2-likeた値ですが、EMV(チップ) カードの 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ないようにします。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

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。

ECC

ECC (楕円曲線暗号) は、楕円曲線の数学を使用して暗号化キーを作成するパブリックキー暗号化システムです。ECC は、RSA などの従来の方法と同じセキュリティレベルを提供しますが、キーの長さははるかに短く、同等のセキュリティをより効率的に提供します。これは、RSA が実用的なソリューションではないユースケース (RSA キーの長さ > 4096 ビット) に特に関連します。 AWS Payment Cryptography は、ECDH オペレーションで使用できるように NIST で定義された曲線をサポートしています。

ECDH

ECDH (Elliptic Curve Diffie-Hellman) は、2 つの当事者が共有シークレット (KEK や PEK など) を確立できるようにする主要な契約プロトコルです。ECDH では、A と B はそれぞれ独自のパブリック/プライベートキーペアを持ち、パブリックキーを相互に (AWS Payment Cryptography の証明書の形式で) およびキー取得メタデータ (取得方法、ハッシュタイプ、共有情報) と交換します。どちらの当事者もプライベートキーに他のパブリックキーを乗算し、楕円曲線プロパティにより、どちらの当事者も結果のキーを取得 (生成) できます。

EMV

EMV (当初は Europay、Mastercard、Visa) は、支払い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、使用する暗号化を含め、チップ/非

接触カードとそれらがやり取り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。

KCAAS

Key Custodian as A Service (KCAAS) は、キー管理に関するさまざまなサービスを提供します。支払いキーの場合、通常、紙ベースのキーコンポーネントを AWS Payment Cryptography でサポートされている電子フォームに変換したり、電子的に保護されたキーを特定のベンダーが必要とする紙ベースのコンポーネントに変換したりできます。また、損失が継続的な運用に悪影響を及ぼすキーに対して、キーエスクローサービスを提供する場合もあります。KCAAS ベンダーは、PCI DSS、PCI PIN、PCI P2PE 標準に準拠する方法で、 AWS Payment Cryptography などの安全なサービス外にキーマテリアルを管理するという運用上の負担をお客様が軽減するのに役立ちます。

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、そのキーは 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。X9.24 によると、TDES の場合、10 バイトの KSN は通常、キーセット ID の 24 ビット、ターミナル ID の 19 ビット、トランザクションカウンターの 21 ビットで構成されますが、キーセット ID とターミナル ID の境界は AWS Payment Cryptography の機能には影響しません。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。

MPoC

MPoC (商用ハードウェアでのモバイルPOS) は、マーチャントがスマートフォンやその他の商用 off-the-shelf (COTS) モバイルデバイスを使用してカード所有者 PINsまたは非接触支払いを受け入れることができるソリューションのセキュリティ要件に対処する PCI 標準です。

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、ポイントオブセール (POS) で匿名で頻繁に使用されるハードウェアデバイスで、カード所有者はこれを使用して支払い認証情報を提示します。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、ECC) 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、ECC) 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、実際のピンを保存せずにピンを検証するために使用できる暗号化出力の一種です。一般的な用語ですが、 AWS Payment Cryptography では、PVV は Visa または ABA PVV メソッドを指します。この PVV は 4 桁の数字で、入力はカード番号、パンシーケンス番号、パン自体、PIN 検証キーです。検証段階では、 AWS Payment Cryptography はトランザクションデータを使用して内部で PVV を再作成し、 AWS Payment Cryptography のお客様が保存した値を再度比較します。このように、概念的には暗号化ハッシュまたは MAC に似ています。

RSA ラップ/ラップ解除

RSA ラップでは、非対称キーを使用して対称キー (TDES キーなど) をラップし、別のシステムに送信します。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 アンラップでは、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、キーを交換する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を利用しません。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一連のキー属性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 に置き換えられました。

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。

X9.143

X9.143 は、米国規格協会 (ANSI) によって定義されたキーブロック形式であり、同じデータ構造内のキー属性とキー属性の保護をサポートします。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 X9.143 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。X9.143 は以前の TR-31 提案を置き換えますが、ほとんどの場合、それらは後方互換性と前方互換性があり、用語はしばしば同じ意味で使用されます。

関連サービス

AWS Key Management Service

AWS Key Management Service (AWS KMS) は、データの保護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。 AWS KMS は、ハードウェアセキュリティモジュール (HSMs) を使用して KMS AWS キーを保護し検証します。

AWS CloudHSM

AWS CloudHSM は、 AWS クラウド内の専用の汎用 HSM インスタンスをお客様に提供します。 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 AWS CloudHSM できます。

詳細情報

- AWS Payment Cryptography で使用される用語と概念については、AWS 「Payment Cryptography の概念」を参照してください。
- AWS Payment Cryptography Control Plane API の詳細については、AWS 「Payment Cryptography Control Plane API リファレンス」を参照してください。
- AWS Payment Cryptography Data Plane API の詳細については、AWS 「Payment Cryptography Data Plane API リファレンス」を参照してください。
- AWS Payment Cryptography が暗号化を使用し、 AWS Payment Cryptography キーを保護する方法に関する詳細な技術情報については、「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、つまりサービスのエントリポイントの URL を使用します。 AWS SDKsとコマンドラインツールは、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 内のサービスのデフォルトエンドポイントを自動的に使用するため、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 (オハイオ)	us-east-2	controlplane.payment-cryptography.us-east-2.amazonaws.com controlplane.payment-cryptography.us-east-2.api.aws	HTTPS HTTPS	
米国東部 (バージニア北部)	us-east-1	controlplane.payment-cryptography.us-east-1.amazonaws.com controlplane.payment-cryptography.us-east-1.api.aws	HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
米国西部 (オレゴン)	us-west-2	controlplane.payment-cryptography.us-west-2.amazonaws.com controlplane.payment-cryptography.us-west-2.api.aws	HTTPS HTTPS	
アフリカ (ケープタウン)	af-south-1	controlplane.payment-cryptography.af-south-1.amazonaws.com controlplane.payment-cryptography.af-south-1.api.aws	HTTPS HTTPS	
アジアパシフィック (ハイデラバード)	ap-south-2	controlplane.payment-cryptography.ap-south-2.amazonaws.com controlplane.payment-cryptography.ap-south-2.api.aws	HTTPS HTTPS	
アジアパシフィック (ムンバイ)	ap-south-1	controlplane.payment-cryptography.ap-south-1.amazonaws.com controlplane.payment-cryptography.ap-south-1.api.aws	HTTPS HTTPS	
アジアパシフィック (大阪)	ap-northeast-3	controlplane.payment-cryptography.ap-northeast-3.amazonaws.com controlplane.payment-cryptography.ap-northeast-3.api.aws	HTTPS HTTPS	
アジアパシフィック (シンガポール)	ap-southeast-1	controlplane.payment-cryptography.ap-southeast-1.amazonaws.com controlplane.payment-cryptography.ap-southeast-1.api.aws	HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
アジアパシフィック (シドニー)	ap-southeast-2	controlplane.payment-cryptography.ap-southeast-2.amazonaws.com controlplane.payment-cryptography.ap-southeast-2.api.aws	HTTPS HTTPS	
アジアパシフィック (東京)	ap-northeast-1	controlplane.payment-cryptography.ap-northeast-1.amazonaws.com controlplane.payment-cryptography.ap-northeast-1.api.aws	HTTPS HTTPS	
カナダ (中部)	ca-central-1	controlplane.payment-cryptography.ca-central-1.amazonaws.com controlplane.payment-cryptography.ca-central-1.api.aws	HTTPS HTTPS	
欧州 (フランクフルト)	eu-central-1	controlplane.payment-cryptography.eu-central-1.amazonaws.com controlplane.payment-cryptography.eu-central-1.api.aws	HTTPS HTTPS	
欧州 (アイルランド)	eu-west-1	controlplane.payment-cryptography.eu-west-1.amazonaws.com controlplane.payment-cryptography.eu-west-1.api.aws	HTTPS HTTPS	
欧州 (ロンドン)	eu-west-2	controlplane.payment-cryptography.eu-west-2.amazonaws.com controlplane.payment-cryptography.eu-west-2.api.aws	HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
欧州 (パリ)	eu-west-3	controlplane.payment-cryptography.eu-west-3.amazonaws.com controlplane.payment-cryptography.eu-west-3.api.aws	HTTPS 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 (オハイオ)	us-east-2	dataplane.payment-cryptography.us-east-2.amazonaws.com dataplane.payment-cryptography.us-east-2.api.aws	HTTPS HTTPS	
米国東部 (バージニア北部)	us-east-1	dataplane.payment-cryptography.us-east-1.amazonaws.com dataplane.payment-cryptography.us-east-1.api.aws	HTTPS HTTPS	
米国西部 (オレゴン)	us-west-2	dataplane.payment-cryptography.us-west-2.amazonaws.com dataplane.payment-cryptography.us-west-2.api.aws	HTTPS HTTPS	
アフリカ (ケープタウン)	af-south-1	dataplane.payment-cryptography.af-south-1.amazonaws.com dataplane.payment-cryptography.af-south-1.api.aws	HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
アジアパシフィック (ハイデラバード)	ap-south-2	dataplane.payment-cryptography.ap-south-2.amazonaws.com dataplane.payment-cryptography.ap-south-2.api.aws	HTTPS HTTPS	
アジアパシフィック (ムンバイ)	ap-south-1	dataplane.payment-cryptography.ap-south-1.amazonaws.com dataplane.payment-cryptography.ap-south-1.api.aws	HTTPS HTTPS	
アジアパシフィック (大阪)	ap-northeast-3	dataplane.payment-cryptography.ap-northeast-3.amazonaws.com dataplane.payment-cryptography.ap-northeast-3.api.aws	HTTPS HTTPS	
アジアパシフィック (シンガポール)	ap-southeast-1	dataplane.payment-cryptography.ap-southeast-1.amazonaws.com dataplane.payment-cryptography.ap-southeast-1.api.aws	HTTPS HTTPS	
アジアパシフィック (シドニー)	ap-southeast-2	dataplane.payment-cryptography.ap-southeast-2.amazonaws.com dataplane.payment-cryptography.ap-southeast-2.api.aws	HTTPS HTTPS	
アジアパシフィック (東京)	ap-northeast-1	dataplane.payment-cryptography.ap-northeast-1.amazonaws.com dataplane.payment-cryptography.ap-northeast-1.api.aws	HTTPS HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
カナダ (中部)	ca-central-1	dataplane.payment-cryptography.ca-central-1.amazonaws.com dataplane.payment-cryptography.ca-central-1.api.aws	HTTPS HTTPS	
欧州 (フランクフルト)	eu-central-1	dataplane.payment-cryptography.eu-central-1.amazonaws.com dataplane.payment-cryptography.eu-central-1.api.aws	HTTPS HTTPS	
欧州 (アイルランド)	eu-west-1	dataplane.payment-cryptography.eu-west-1.amazonaws.com dataplane.payment-cryptography.eu-west-1.api.aws	HTTPS HTTPS	
欧州 (ロンドン)	eu-west-2	dataplane.payment-cryptography.eu-west-2.amazonaws.com dataplane.payment-cryptography.eu-west-2.api.aws	HTTPS HTTPS	
欧州 (パリ)	eu-west-3	dataplane.payment-cryptography.eu-west-3.amazonaws.com dataplane.payment-cryptography.eu-west-3.api.aws	HTTPS HTTPS	

AWS Payment Cryptography の開始方法

AWS Payment Cryptography の使用を開始するには、まずキーを作成してから、さまざまな暗号化オペレーションでキーを使用します。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るには、次の AWS Payment Cryptography Workshop を試すか、GitHub で利用可能なサンプルプロジェクトを試してください。

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。

> **Warning**
>
> このユーザーガイドの例では、サンプル値を使用する場合があります。キーシリアル番号などの本番環境では、サンプル値を使用しないことを強くお勧めします。

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 SDKs または AWS APIs 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手順では を使用します AWS CLI。

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,KeyModesO
```

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_C0_CARD_VERIFICATION_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。

このセクションでは、ライフサイクル全体でさまざまな AWS Payment Cryptography キータイプを作成および管理する方法を説明します。キーを作成、表示、編集する方法と、キーにタグを付ける方法、キーエイリアスを作成する方法、キーを有効または無効にする方法について説明します。

AWS Payment Cryptography キーはリージョンリソースです。複数の で特定のキーを使用する場合は AWS リージョン、マルチリージョンキーレプリケーションを有効にして、キーマテリアルとメタデータを同じ AWS パーティションとアカウント内で AWS リージョン 指定した に安全にコピーできます。マルチリージョンキーレプリケーションのソースキーはプライマリリージョンキー (PRK) と呼ばれ、これは引き続きすべてのキー管理アクティビティの信頼できるソースです。レプリケートされたキーはレプリカリージョンキー (RRK) と呼ばれ、これは PRK の読み取り専用レプリカです。可用性、ディザスタリカバリ、低レイテンシーに関する設計目標を達成するために、キーでマルチリージョンキーを使用することを検討する必要があります。

トピック

キーの作成

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。キーを作成するときは、キーアルゴリズム、キーの使用方法、許可されたオペレーション、エクス

ポート可能かどうかなどの属性を指定します。 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。

> **Note**
>
> でマルチリージョンキーレプリケーションが有効になっていて Payment Cryptography キーを作成する AWS アカウント と、このキーは自動的にプライマリリージョンキー (PRK) になります。CreateKey コマンドで `--replication-regions`パラメータを指定しない場合でも、PRK はレプリケートされます。詳細については、「マルチリージョンキーレプリケーションの仕組み」を参照してください。

例

- 3KEY TDES ベース取得キーの作成
- CVV/CVV2 用の 2KEY TDES キーの作成
- HMAC キーの作成
- AES-256 キーの作成
- PIN 暗号化キー (PEK) の作成
- 非対称 (RSA) キーの作成
- PIN 検証値 (PVV) キーの作成
- 非対称 ECC キーの作成

3KEY TDES ベース取得キーの作成

Example

このコマンドは、米国東部 (オハイオ) および米国西部 (オレゴン) リージョンにレプリケートされる 3KEY TDES 取得キーを作成します。レスポンスには、reques パラメータ、後続の呼び出しの Amazon リソースネーム (ARN)、およびキーチェック値 (KCV) が含まれます。

```
$ aws payment-cryptography create-key --exportable --key-attributes \
     "KeyUsage=TR31_B0_BASE_DERIVATION_KEY, \
     KeyClass=SYMMETRIC_KEY,KeyAlgorithm=TDES_3KEY, \
     KeyModesOfUse={NoRestrictions=true}" \
     --replication-regions us-east-2 --region us-west-2
```

出力例:

```
{
    "Key": {
        "CreateTimestamp": "2022-10-26T16:04:11.642000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "FE23D3",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": true,
                "Encrypt": false,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_B0_BASE_DERIVATION_KEY"
        },
        "KeyCheckValue": "FE23D3",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-26T16:04:11.559000-07:00"
}
```

CVV/CVV2 用の 2KEY TDES キーの作成

Example

このコマンドは、CVV/CVV2 値を生成および検証するための 2KEY TDES キーを作成します。CVV2 レスポンスには、リクエストパラメータ、後続の呼び出しの Amazon リソースネーム (ARN)、およびキーチェック値 (KCV) が含まれます。

```
$ 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}'
```

出力例:

```
{
    "Key": {
        "CreateTimestamp": "2022-10-26T16:04:11.642000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu",
        "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": "AEA5CD",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-26T16:04:11.559000-07:00"
    }
}
```

HMAC キーの作成

Example

HMAC キーは、ハッシュメッセージ認証コード (HMAC) の生成または検証に使用されます。HMAC キーでは、ハッシュタイプはキーの作成時に割り当てられ (HMAC_SHA224 や HMAC_SHA512 など)、変更することはできません。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=HMAC_SHA512,KeyUsage=TR31_M7_HMAC_KEY,KeyClass=SYMMETRIC_KEY,KeyModesOfUse='{Gene
 = true,Verify = true}'
```

出力例:

```
{
 "Key": {
 "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/qnobl5lghrzunce6",
 "KeyAttributes": {
 "KeyUsage": "TR31_M7_HMAC_KEY",
 "KeyClass": "SYMMETRIC_KEY",
 "KeyAlgorithm": "HMAC_SHA512",
 "KeyModesOfUse": {
 "Encrypt": false,
 "Decrypt": false,
 "Wrap": false,
 "Unwrap": false,
 "Generate": true,
 "Sign": false,
 "Verify": true,
 "DeriveKey": false,
 "NoRestrictions": false
 }
 },
 "KeyCheckValue": "2976E7",
 "KeyCheckValueAlgorithm": "HMAC",
 "Enabled": true,
 "Exportable": true,
 "KeyState": "CREATE_COMPLETE",
 "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
 "CreateTimestamp": "2025-07-30T10:06:12.142000-07:00",
 "UsageStartTimestamp": "2025-07-30T10:06:12.128000-07:00"
 }
}
```

AES-256 キーの作成

Example

このコマンドは、データの暗号化と復号のための AES-256 対称キーを作成します。AES キーは機密データの強力な暗号化を提供し、カード所有者データやその他の機密情報を暗号化するための支払い処理に一般的に使用されますが、TDES は EMV などの発行者のユースケースでより一般的に使用されます。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=AES_256,KeyUsage=TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY,KeyClass=SYMMETRIC_KEY,Key
```

出力例:

```
{
    "Key": {
        "CreateTimestamp": "2025-02-02T10:15:30.142000-08:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyAlgorithm": "AES_256",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY"
        },
        "KeyCheckValue": "2976F5",
        "KeyCheckValueAlgorithm": "CMAC",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2025-02-02T10:15:30.128000-08:00"
    }
}
```

PIN 暗号化キー (PEK) の作成

Example

このコマンドは PIN 値を暗号化するための 3KEY TDES キーを作成しますが、相互運用性の必要性に応じてピンキーを AES にすることもできます。このキーを使用して、トランザクションなど、検証中に PINs を安全に保存したり、PINs を復号したりできます。レスポンスには、リクエストパラメータ、後続の呼び出しの ARN、KCV が含まれます。

```
$ 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}'
```

出力例:

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T08:27:51.795000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
        "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": "7CC9E2",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
    }
}
```

非対称 (RSA) キーの作成

Example

このコマンドは、新しい非対称 RSA 2048 ビットキーペアを生成します。新しいプライベートキーとそれに一致するパブリックキーが作成されます。getPublicCertificate API を使用してパブリックキーを取得できます。

```
$ 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}'
```

出力例:

```
{
    "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": "SHA-1",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-11-15T11:15:42.182000-08:00"
    }
}
```

PIN 検証値 (PVV) キーの作成

Example

このコマンドは、PVV 値を生成するための 3KEY TDES キーを作成します。このキーを使用して、後で計算された PVV と比較できる PVV を生成できます。レスポンスには、リクエストパラメータ、後続の呼び出しの ARN、KCV が含まれます。

```
$ 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}'
```

出力例:

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T10:22:59.668000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
        "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": "7F2363",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-27T10:22:59.614000-07:00"
    }
}
```

非対称 ECC キーの作成

Example

このコマンドは、2 つの当事者間で ECDH (Elliptic Curve Diffie-Hellman) キーアグリーメントを確立するための ECC キーペアを生成します。ECDH では、各当事者がキー目的 K3 とモード X を持つ独自の ECC キーペアを生成し、パブリックキーを交換します。次に、両者はプライベートキーと受信したパブリックキーを使用して、共有派生キーを確立します。

支払いで暗号化キーのシングルユースの原則を維持するために、ECDH キーの取得や署名など、複数の目的で ECC キーペアを再利用しないことをお勧めします。

```
$ aws payment-cryptography create-key --exportable \
    --key-attributes
 KeyAlgorithm=ECC_NIST_P256,KeyUsage=TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT, \
    KeyClass=ASYMMETRIC_KEY_PAIR,KeyModesOfUse='{DeriveKey=true}'
```

出力例:

```
{
    "Key": {
        "CreateTimestamp": "2024-10-17T01:31:55.908000+00:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
wc3rjsssguhxtilv",
        "KeyAttributes": {
            "KeyAlgorithm": "ECC_NIST_P256",
            "KeyClass": "ASYMMETRIC_KEY_PAIR",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": true,
                "Encrypt": false,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": false,
                "Wrap": false
            },
            "KeyUsage": "TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT"
        },
        "KeyCheckValue": "7E34F19F",
        "KeyCheckValueAlgorithm": "SHA-1",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2024-10-17T01:31:55.866000+00:00"
    }
}
```

キーの一覧表示

ListKeys オペレーションを使用して、アカウントとリージョンでアクセスできるキーのリストを取得します。

Example

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

出力例:

```
{
    "Keys": [
        {
            "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
            "Enabled": false,
            "Exportable": true,
            "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
            "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": "7F2363",
            "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、今後データを暗号化または復号化するためにキーを使用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

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"
  }
}
```

AWS Payment Cryptography キーのレプリケート

AWS Payment Cryptography はマルチリージョンキーレプリケーションをサポートしているため、特定の AWS Payment Cryptography キーから同じ AWS パーティションとアカウント AWS リージョン 内の 1 つ以上の にキーマテリアルとメタデータを安全に配布できます。

ソースキーはプライマリリージョンキー (PRK) と呼ばれ、すべてのキー管理アクティビティの信頼できるソースのままですが、PRK とレプリカリージョンキー (RRK) の両方をそれぞれの暗号化オペレーションに使用できます AWS リージョン。

マルチリージョンキーレプリケーションの利点

マルチリージョンキーレプリケーションの利点を以下に示します。

- 高可用性アプリケーションの簡単なセットアップ - AWS Payment Cryptography はキー分散を処理するため、特定のキーの分離されたコピーを作成することなく、複数の AWS リージョン でキーを使用できます。
- 高可用性キーと低レイテンシーキー - マルチリージョンキーレプリケーションを使用すると、キーに複数の でアクセス AWS リージョン でき、可用性が高くなり、レイテンシーが低くなります。
- キーマテリアルの耐久性 - レプリカリージョンキーは完全なキーレプリカであり、暗号化オペレーションでプライマリリージョンキーとは独立して使用できます。RRK は、PRK の壊滅的なデータ損失が発生した場合に耐久性のあるレプリカを提供します。

マルチリージョンキーレプリケーションの仕組み

マルチリージョンキーレプリケーションが有効になっている場合、 AWS Payment Cryptography サービスは安全なキー配布メカニズムを使用して、キーマテリアルとメタデータを指定したレプリカにコピー AWS リージョン します。キー属性、状態、有効化などのプライマリリージョンキーメタデータへの変更は、レプリカリージョンキーに自動的にレプリケートされます。

制約事項と考慮事項

以下は、マルチリージョンキーレプリケーションの制限と考慮事項です。

- この機能は、 AWS リージョン または特定の Payment Cryptography キーで有効にする必要があります。
  - でこの機能が有効になっている場合 AWS リージョン、有効化後に作成されたすべての AWS Payment Cryptography キーは、指定された にレプリケートされます AWS リージョン。この

リージョンで作成されたキーは、プライマリリージョンキーになります。このリージョンの既存のキーは自動的にレプリケートされません。キーレベルで、 内の既存のキーに対してマルチリージョン AWS リージョン キーレプリケーションを有効にできます。

  - 各 は、一意のマルチリージョンキーレプリケーション設定を持つ AWS リージョン ことができます。
  - キーのマルチリージョンレプリケーション設定は、 AWS リージョン マルチリージョンキーレプリケーション設定よりも優先されます。
- レプリカリージョンキーを他の にレプリケートするように設定することはできません AWS リージョン。
- マルチリージョンキーレプリケーションは、トリプルデータ暗号化標準 (3DES)、アドバンスト暗号化標準 (AES)、ハッシュベースのメッセージ認証コード (HMAC) などの対称 Payment Cryptography キーで使用できます。
- 非対称 Payment Cryptography キーは、マルチリージョンキーレプリケーションをサポートしていません。
- レプリカリージョンキーは読み取り専用キーです。プライマリリージョンキーへのすべての変更は、レプリカリージョンキーに適用されます。
- プライマリリージョンキーの変更は、最終的にレプリカリージョンキーと一致します。
- Payment Cryptography キーは、同じ AWS パーティションとアカウントでのみレプリケートできます。
- レプリカリージョンキーは AWS アカウント 、レベル AWS Payment Cryptography 制限にカウントされます。
- プライマリリージョンキーとレプリカリージョンキーは同じキー識別子を使用します。これにより、IAM ポリシーで同じ ARN で両方のキーを参照できます。
- レプリケーションを成功させるには、レプリカ AWS リージョン に アクセス`CreateKey`許可が必要です。

マルチリージョンキーレプリケーションの有効化

AWS Payment Cryptography キーのマルチリージョンキーレプリケーションを有効にするには、2 つの方法があります。

1. AWS リージョン: マルチリージョンキーレプリケーションは、有効に AWS リージョン すると、そのキーで作成されたすべての新しいキーに適用されます。このメソッドは、すべてのキーに対して一貫したレプリケーションを提供します。

2. 特定の AWS Payment Cryptography キー: 個々のキーのマルチリージョンキーレプリケーションを管理して、より詳細な制御レベルを実現できます。

マルチリージョンキーレプリケーションを有効にすると、 Payment Cryptography キーは指定した AWS リージョン にレプリケートされます。

> **Important**
>
> マルチリージョンキーレプリケーションを一時停止することはできません。キーは、レプリケーションが有効になると AWS リージョン 、指定した に自動的にレプリケートされます。マルチリージョンキーレプリケーションは、特定の AWS リージョン または Payment Cryptography キーに対して無効にできます。レプリカリージョンキーを削除するには、プライマリリージョンキーからレプリケーションリージョン AWS リージョン として を削除する必要があります。
> または、PRK stop-key-usage で `StopKeyUsage` API または CLI コマンドを呼び出して、PRK および関連するすべての RRKs の両方の使用を停止することもできます。これらのキーを暗号化オペレーションで使用することはできません。`StopKeyUsage` API または stop-key-usage CLI コマンドを使用しても、PRK で有効な進行中のマルチリージョンキーレプリケーションは停止されません。

特定の AWS リージョン の AWS Payment Cryptography キーのマルチリージョンキーレプリケーション設定を確認するには、 `GetDefaultKeyReplicationRegions` API または get-default-key-replication-regions CLI コマンドを呼び出します。この API アクションまたはコマンド AWS リージョン を呼び出す のキーが PRK になります。

マルチリージョンキーレプリケーションを有効にするには、次の手順に従います。

For AWS リージョン

- 次のコマンドを使用して、 AWS リージョン 指定した のマルチリージョンキーレプリケーションを有効にします。この例では、米国東部 (オハイオ) および米国西部 (オレゴン) でマルチリージョンキーレプリケーションが有効になっています。このコマンドを使用するには、コマンド例の###############を独自の情報に置き換えます。

```
aws payment-cryptography enable-default-key-replication-regions \
    --replication-regions us-east-2 us-west-2
```

> **Note**
>
> のマルチリージョンキーレプリケーションを有効に AWS リージョン しても、既存の AWS Payment Cryptography キーのレプリケーション設定は変更されません。この機能は、既存のキーに対してキーレベルで有効にできます。マルチリージョンキーレプリケーションが に対して有効になった後に作成されたキーのみが AWS リージョン 、リージョンレプリケーション設定を使用します。

For specific AWS Payment Cryptography keys

- 次のコマンドを使用して、特定の Payment Cryptography キーのマルチリージョンキーレプリケーションを有効にします。この例では、米国東部 (オハイオ) でマルチリージョンキーレプリケーションが有効になっています。このコマンドを使用するには、コマンド例の*####################*を独自の情報に置き換えます。

```
aws payment-cryptography add-key-replication-regions \
    --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h \
    --replication-regions us-east-2
```

または、キー作成リクエストにレプリケーションを含めることで、この機能を有効にして新しい Payment Cryptography キーを作成することもできます。 AWS リージョン

> **Note**
>
> キーレプリケーション設定は、レプリ AWS リージョン ケーション設定よりも優先されます。

マルチリージョンキーレプリケーションの無効化

マルチリージョンキーレプリケーションを無効にする場合は、マルチリージョンキーレプリケーションを有効にする方法に応じて、 disable-default-key-replicationまたは remove-key-replication-regions CLI コマンドを呼び出すことができます。マルチリージョンキーレプリケーション AWS リージョン を無効にするには、キーの ARN と を指定する必要があります。

考慮事項

レプリケーションリージョンキーの削除は結果整合性があります。

特定の AWS リージョン の AWS Payment Cryptography キーのマルチリージョンキーレプリケーション設定を確認するには、 `GetDefaultKeyReplicationRegions` API または get-default-key-replication-regions CLI コマンドを呼び出します。

マルチリージョンキーのレプリケーションを無効にするには、次の手順に従います。

For AWS リージョン

- 次のコマンドを使用して、指定した AWS リージョン のマルチリージョンキーレプリケーションを無効にします。この例では、米国東部 (オハイオ) でマルチリージョンキーレプリケーションが無効になっています。このコマンドを使用するには、コマンド例の*##########*を独自の情報に置き換えます。

```
aws payment-cryptography disable-default-key-replication-regions \
    --replication-regions us-east-2
```

For specific AWS Payment Cryptography keys

- 次のコマンドを使用して、特定の Payment Cryptography キーのマルチリージョンキーレプリケーションを無効にします。この例では、米国東部 (オハイオ) でマルチリージョンキーレプリケーションは無効になっています。このコマンドを使用するには、コマンド例の*############*を独自の情報に置き換えます。

```
aws payment-cryptography remove-key-replication-regions \
    --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h \
    --replication-regions us-east-2
```

セキュリティに関する考慮事項

Payment Cryptography キーにマルチリージョンキーレプリケーションを使用する場合のセキュリティ上の考慮事項を次に示します。詳細については、「AWS Payment Cryptography 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。

- キーマテリアルの共有を制限します。
- IAM ポリシーを作成するときは、最小権限のアクセス許可のプリンシパルに従います。

- 読み取り専用キーであるため、レプリカリージョンキーを変更することはできません。

ベストプラクティス

Payment Cryptography キーでマルチリージョンキーレプリケーションを使用する場合 AWS のベストプラクティスを以下に示します。

- 指定した へのマルチリージョンキーレプリケーションがすぐには行われない場合でも、アプリケーション AWS リージョン が動作し続けます。マルチリージョンキーのレプリケーションが完了したタイミングを知る必要がある場合は、GetKey API アクションを使用してモニタリングできます。を使用して、キーレプリケーションイベントをモニタリングできますAWS CloudTrail。
- あるリージョンから別のリージョン AWS リージョン にフェイルオーバーした場合の自動デプロイプロセスをテストして実装します。

料金

AWS Payment Cryptography で作成したレプリカリージョンキーには料金が発生します。これらのキーは ごとに課金されます AWS リージョン。Payment Cryptography の最新の料金情報については、 AWS Payment Cryptography の料金ページを参照してください。

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 の外部で利用できる場合を除き、元に戻すことはできません。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、キーを削除するのではなく、キーの使用を停止することを検討してください。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できる場合を除き、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**
>
> を閉じまたは削除 AWS アカウントすると、 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たとき、および 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。

マルチリージョンキーレプリケーションを使用する場合、プライマリリージョンキー (PRK) である Payment Cryptography キーを削除すると、レプリカリージョンキー (RRK) も自動的に削除されます。RRK を PRK のように削除することはできません。RRK を削除する場合は、PRK のレプリケーションリージョンを変更する必要があります。

待機期間について

キーの削除は元に戻すことができないため、 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 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。基本的なキー情報の他に、キーの状態が DELETE_PENDING に変更されたというフィールドと、DeletePendingTimestamp がキーの現在の削除予定日時を表すフィールドが 2 つあります。

```
$ 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": "0A3674",
        "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": "0A3674",
        "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

Payment Cryptography AWS キーを他のソリューションからインポートし、HSMs などの他のソリューションにエクスポートできます。多くのお客様は、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します。Payment Cryptography AWS は、コンプライアンスとコントロールの維持に役立つ、最新の電子的なキー管理アプローチを使用するように設計されています。紙ベースのキーコンポーネントの代わりに、標準ベースの電子キー交換を使用することをお勧めします。

最小キー強度とインポートおよびエクスポート関数への影響

PCI では、暗号化オペレーション、キーストレージ、キー転送に特定の最小キー強度が必要です。これらの要件は、PCI 標準が改訂されると変更される可能性があります。ルールでは、ストレージまたはトランスポートに使用されるラッピングキーは、少なくとも保護されるキーと同じ強度にする必要があります。次の表に示すように、エクスポート中にこの要件を自動的に適用し、キーがより弱いキーによって保護されないようにします。

次の表は、ラッピングキー、保護するキー、および保護方法のサポートされている組み合わせを示しています。

	ラップキー											
保護するキー	TDES	TDES	AES_	AES_	AES_	RSA_	RSA_	RSA_	ECC_	ECC_	ECC_	注意事項
TDES_2KEY	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	ECDH	ECDH	ECDH	
TDES_3KEY	✗ サポートされていません	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	ECDH	ECDH	ECDH	

	ラップキー											
保護するキー	TDES	TDES	AES_	AES_	AES_	RSA_	RSA_	RSA_	ECC_	ECC_	ECC_	注意事項
AES_128	✗ サポートされていません	✗ サポートされていません	TR-3	TR-3	TR-3	✗ サポートされていません	TR-3	TR-3	ECDH	ECDH	ECDH	
AES_192	✗ サポートされていません	✗ サポートされていません	✗ サポートされていません	TR-3	TR-3	✗ サポートされていません	✗ サポートされていません	✗ サポートされていません	✗ サポートされていません	ECDH	ECDH	

	ラップキー											
保護するキー	TDES	TDES	AES_	AES_	AES_	RSA_	RSA_	RSA_	ECC_	ECC_	ECC_	注意事項
AES_256	✗ サポートされていません	✗ サポートされていません	✗ サポートされていません	✗ サポートされていません	TR-3	✗ サポートされていません	✗ サポートされていません	✗ サポートされていません	✗ サポートされていません	✗ サポートされていません	ECDH	

詳細については、PCI HSM 標準の「付録 D - 承認済みアルゴリズムの最小および同等のキーサイズと強度」を参照してください。

キー暗号化キー (KEK) 交換

ANSI X9.24 TR-34 標準を使用することをお勧めします。この初期キータイプは、キー暗号化キー (KEK)、ゾーンマスターキー (ZMK)、またはゾーンコントロールマスターキー (ZCMK) と呼ばれます。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 Wrap/Unwrap を使用できます。AES-256 キーの交換が必要な場合は、ECDH を使用できます。

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、紙のキーコンポーネントを引き続き処理する必要がある場合は、オフライン HSM を使用するか、サードパーティーのキーカストディアンをサービスとして使用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り、既存の HSMs とキーを同期したりするには、GitHub の AWS Payment Cryptography サンプルコードを参照してください。

ワーキングキー (WK) 交換

作業キーの交換には、業界標準 (ANSI X9.24 TR 31-2018 および X9.143) を使用します。これには、TR-34、RSA Wrap、ECDH、または同様のスキームを使用して KEK を既に交換していることが必要です。このアプローチは、キーマテリアルをそのタイプと使用状況に暗号化的にバインドするための PCI PIN 要件を満たしています。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 が含まれます。

トピック

キーのインポート

> **Important**
>
> 例には、AWS CLI V2 の最新バージョンが必要です。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。

目次

キーのインポートの概要

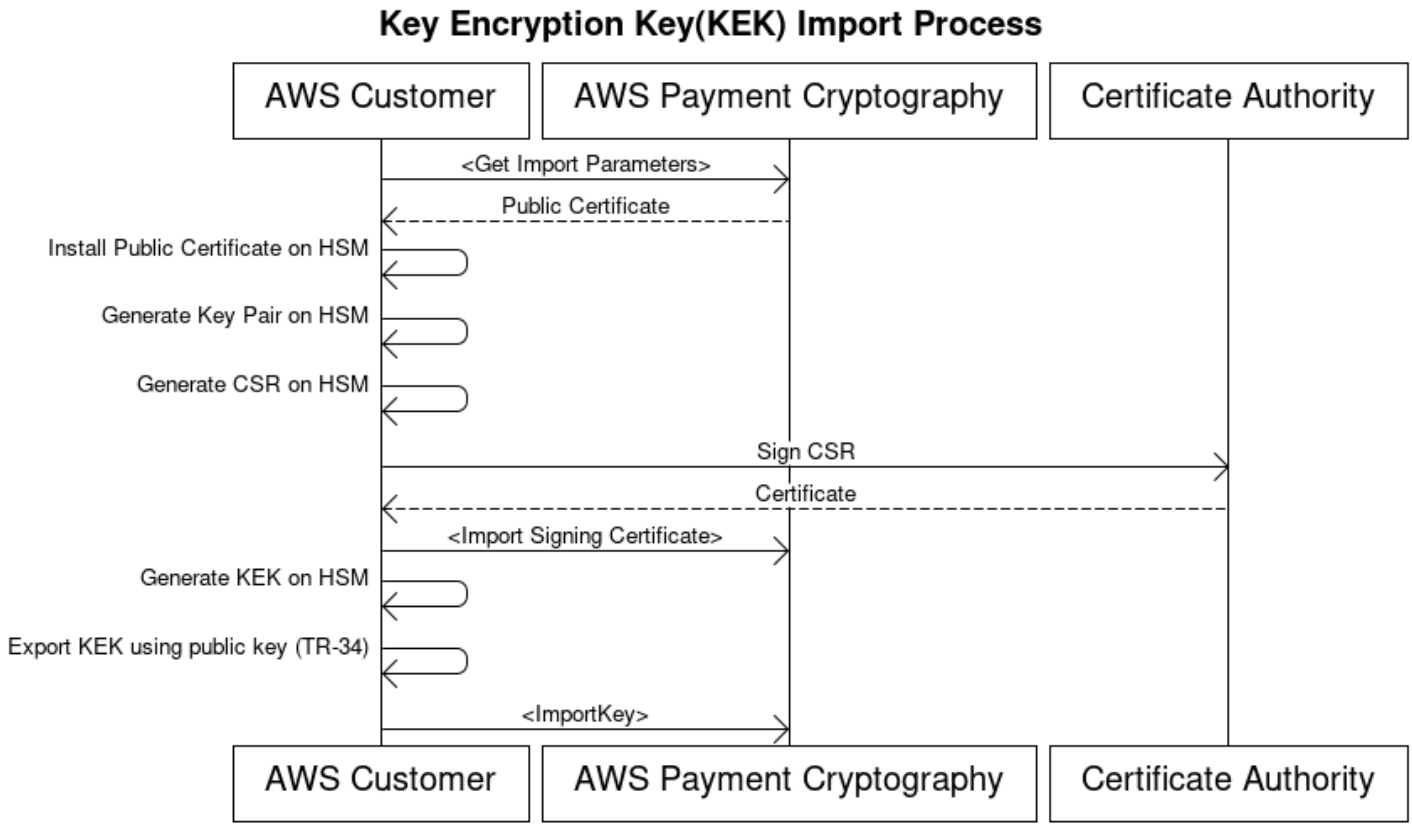
> **Note**
>
> X9.143、TR-31、または TR-34 キーブロックを使用してキーをインポートする場合、 AWS Payment Cryptography は通常、オプションのヘッダーを保持します (ただし、使用しません) 。HM(HMAC ハッシュタイプ) ヘッダーは、暗号化オペレーション中に使用されます。KP ヘッダー (ラッピングキーの KCV) はインポートプロセスに固有であり、保持されません。

カウンターパーティとキーを交換する場合、通常は最初にキー交換キー (KEK) を交換します。その後、このキーは後続のキーを保護するために使用されます。電子形式を使用すると、TR-34、ECDH、RSA ラップなどの非対称手法を使用して KEK を交換することができます。以降のキーは、TR-31 などの対称キー交換を使用して交換されます。この KEK は存続期間が長く、ポリシーとその定義された暗号化期間に基づいて数年ごとにのみ更新できます。

1 つまたは 2 つのキーのみを交換する場合は、非対称手法を使用して BDK などのキーを直接交換することもできます。 AWS Payment Cryptography は、両方のキー交換方法をサポートしています。

対称キーのインポート

非対称手法 (TR-34) によるキーのインポート



TR-34 は RSA 非対称暗号化を使用して、交換のために対称キーを暗号化して署名します。これにより、ラップされたキーの機密性 (暗号化) と整合性 (署名) の両方が確保されます。

独自のキーをインポートするには、GitHub の AWS Payment Cryptography サンプルプロジェクトを確認してください。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、GitHub でサンプルコードを参照するか、これら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. Initialize Import 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、キーインポート用のキーペアを生成し、キーに署名して、証明書と証明書ルートを返します。このキーを使用してエクスポートするキーを暗号化します。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 base64 でエンコードされ、有効期間が短く、この目的にのみ使用されます。`ImportToken` 値を保存します。

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

```
{
    "ImportToken": "import-token-bwxli6ocftypneu5",
    "ParametersValidUntilTimestamp": 1698245002.065,
    "WrappingKeyCertificateChain": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0....",
    "WrappingKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0....",
    "WrappingKeyAlgorithm": "RSA_2048"
}
```

2. キーソースシステムにパブリック証明書をインストールする

   ほとんどの HSMs では、ステップ 1 で生成されたパブリック証明書をインストール、ロード、または信頼して、それを使用してキーを エクスポートする必要があります。これには、HSM に応じて、証明書チェーン全体またはステップ 1 のルート証明書のみが含まれます。

3. ソースシステムでキーペアを生成し、証明書チェーンを 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 (キーディストリビューションホストまたは KDH) がペイロードに署名します。この目的のためにパブリックキーを生成し、 AWS パブリックキー証明書 (X509) を作成して Payment Cryptography に返します。

   HSM からキーを転送する場合は、その HSM にキーペアを作成します。HSM、サードパーティー、または などのサービスが証明書を生成 AWS Private CA できます。

   KeyMaterialType が `RootCertificatePublicKey`、KeyUsageType が の `importKey` コマンドを使用して、ルート証明書を AWS Payment Cryptography にロードしますTR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。

   中間証明書の場合は、KeyMaterialType が `TrustedCertificatePublicKey`、KeyUsageType が の `importKey` コマンドを使用しますTR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。複数の中間証明書に対してこのプロセスを繰り返します。チェーンで最後にインポートされた証明書KeyArnの を、後続のインポートコマンドの入力として使用します。

> **Note**
>
> リーフ証明書をインポートしないでください。インポートコマンド中に直接指定します。

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 標準を使用したキーのエクスポートをサポートしています。ステップ 1 のパブリックキーを KRD (暗号化) 証明書として指定し、ステップ 3 のキーを KDH (署名) 証明書として指定します。 AWS Payment Cryptography にインポートするには、TR-34.2012 non-CMS two pass 形式を指定します。これは TR-34 Diebold 形式とも呼ばれます。

5. コールインポートキー

   KeyMaterialType が の importKey API を呼び出しますTR34_KEY_BLOCK。ステップ 3 でインポートされた最後の CA の keyARN`certificate-authority-public-key-identifier`、ステップ 4 でラップされたキーマテリアルを に`key-material`、ステップ 3 でリーフ証明書をに使用します`signing-key-certificate`。ステップ 1 の import-token を含めます。

```
$ aws payment-cryptography import-key \
    --key-material='{"Tr34KeyBlock": { \
    "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/zabouwe3574jysdl", \
    "ImportToken": "import-token-bwxli6ocftypneu5", \
    "KeyBlockFormat": "X9_TR34_2012", \
    "SigningKeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2RENDQXFTZ0F3SUJ...", \
    "WrappedKeyBlock":
 "308205A106092A864886F70D010702A08205923082058E020101310D300B0609608648016503040201308203.
 \
    }'
```

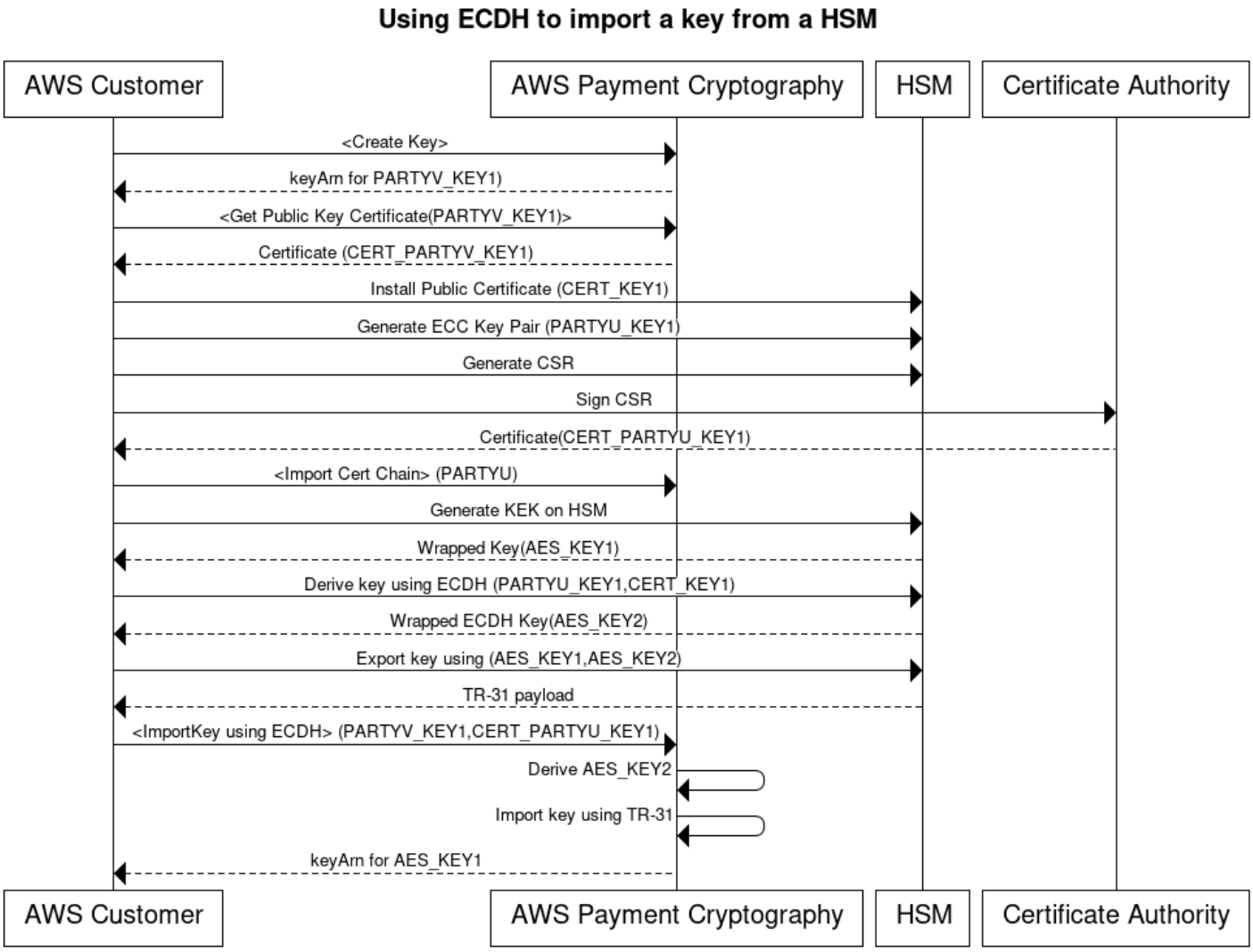
```
{
  "Key": {
    "CreateTimestamp": "2023-06-13T16:52:52.859000-04:00",
    "Enabled": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
    "KeyAttributes": {
```

```
      "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_K1_KEY_ENCRYPTION_KEY"
    },
    "KeyCheckValue": "CB94A2",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "KeyOrigin": "EXTERNAL",
    "KeyState": "CREATE_COMPLETE",
    "UsageStartTimestamp": "2023-06-13T16:52:52.859000-04:00"
  }
}
```


6. 暗号化オペレーションまたはそれ以降のインポートにインポートされたキーを使用する

   インポートされた KeyUsage が TR31_K0_KEY_ENCRYPTION_KEY の場合、TR-31 を使用した後続のキーインポートにこのキーを使用できます。他のキータイプ (TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY など) では、暗号化オペレーションにキーを直接使用できます。

非対称手法 (ECDH) を使用してキーをインポートする



Elliptic Curve Diffie-Hellman (ECDH) は、ECC 非対称暗号化を使用して、事前交換されたキーを必要とせずに、2 つの当事者間で共有キーを確立します。ECDH キーはエフェメラルであるため、AWS Payment Cryptography では保存されません。このプロセスでは、ECDH を使用して 1 回限りの KBPK/KEK が導出されます。その派生キーは、別の KBPK、IPEK キー、またはその他のキータイプなど、転送する実際のキーをラップするためにすぐに使用されます。

インポートする場合、送信システムは一般的に Party U (Initiator) と呼ばれ、 AWS Payment Cryptography は Party V (Responder) と呼ばれます。

> **Note**
>
> ECDH は任意の対称キータイプの交換に使用できますが、AES-256 キーを安全に転送できる唯一のアプローチです。

1. ECC キーペアの生成

   を呼び出しcreate-keyて、このプロセスの ECC キーペアを作成します。この API は、キーのインポートまたはエクスポート用のキーペアを生成します。作成時に、この ECC キーを使用して派生できるキーの種類を指定します。ECDH を使用して他のキーを交換 (ラップ) する場合は、 の値を使用しますTR31_K1_KEY_BLOCK_PROTECTION_KEY。

   > **Note**
   >
   > 低レベルの ECDH は、任意の目的に使用できる派生キーを生成しますが、 AWS Payment Cryptography は、キーを単一の派生キータイプにのみ使用できるようにすることで、複数の目的でキーを誤って再利用することを制限します。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=ECC_NIST_P256,KeyUsage=TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT,KeyClass=ASYM
 --derive-key-usage "TR31_K1_KEY_BLOCK_PROTECTION_KEY"
```

```
{
                "Key": {
                    "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/wc3rjsssguhxtilv",
                    "KeyAttributes": {
                        "KeyUsage": "TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT",
                        "KeyClass": "ASYMMETRIC_KEY_PAIR",
                        "KeyAlgorithm": "ECC_NIST_P256",
                        "KeyModesOfUse": {
                            "Encrypt": false,
                            "Decrypt": false,
                            "Wrap": false,
                            "Unwrap": false,
                            "Generate": false,
                            "Sign": false,
```

```
                            "Verify": false,
                            "DeriveKey": true,
                            "NoRestrictions": false
                        }
                    },
                    "KeyCheckValue": "2432827F",
                    "KeyCheckValueAlgorithm": "CMAC",
                    "Enabled": true,
                    "Exportable": true,
                    "KeyState": "CREATE_COMPLETE",
                    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                    "CreateTimestamp": "2025-03-28T22:03:41.087000-07:00",
                    "UsageStartTimestamp": "2025-03-28T22:03:41.068000-07:00"
                }
            }
```

2. パブリックキー証明書の取得

   `get-public-key-certificate` を呼び出して、特定のリージョンの AWS Payment Cryptography に固有のアカウントの CA によって署名された X.509 証明書としてパブリックキーを受け取ります。

   Example

```
$ aws payment-cryptography get-public-key-certificate \
                  --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/wc3rjsssguhxtilv
```

```
{
            "KeyCertificate": "LS0tLS1CRUdJTi...",
            "KeyCertificateChain": "LS0tLS1CRUdJT..."
      }
```

3. カウンターパーティシステムにパブリック証明書をインストールする (当事者 U)

   多くの HSMs では、ステップ 1 で生成されたパブリック証明書をインストール、ロード、または信頼して、それを使用してキーを エクスポートする必要があります。これには、HSM に応じて、証明書チェーン全体またはステップ 1 のルート証明書のみが含まれます。詳細については、HSM ドキュメントを参照してください。

4. ソースシステムで ECC キーペアを生成し、証明書チェーンを AWS Payment Cryptography に提供する

   ECDH では、各当事者がキーペアを生成し、共通のキーについて合意します。 AWS Payment Cryptography がキーを取得するには、X.509 パブリックキー形式のカウンターパーティのパブリックキーが必要です。

   HSM からキーを転送する場合は、その HSM にキーペアを作成します。キーブロックをサポートする HSMs、キーヘッダーは のようになりますD0144K3EX00E0000。証明書を作成するときは、通常、HSM で CSR を生成し、HSM、サードパーティー、 などのサービスが証明書を生成 AWS Private CA できます。

   KeyMaterialType が RootCertificatePublicKey、KeyUsageType が の importKey コマンドを使用して、ルート証明書を AWS Payment Cryptography にロードしますTR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。

   中間証明書の場合は、KeyMaterialType が TrustedCertificatePublicKey、KeyUsageType が の importKey コマンドを使用しますTR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。複数の中間証明書に対してこのプロセスを繰り返します。チェーンで最後にインポートされた証明書KeyArnの を、後続のインポートコマンドの入力として使用します。

   > **Note**
   >
   > リーフ証明書をインポートしないでください。インポートコマンド中に直接指定します。

5. Party U HSM で ECDH を使用して 1 回限りのキーを取得する

   多くの HSMsおよび関連システムは、ECDH を使用したキーの確立をサポートしています。ステップ 1 のパブリックキーをパブリックキーとして指定し、ステップ 3 のキーをプライベートキーとして指定します。取得方法などの許容されるオプションについては、 API ガイドを参照してください。

   > **Note**
   >
   > ハッシュタイプなどの取得パラメータは、両側で完全に一致する必要があります。それ以外の場合は、別のキーを生成します。

6. ソースシステムからキーをエクスポートする

   最後に、標準の TR-31 コマンドを使用して AWS Payment Cryptography に転送するキーをエクスポートします。ECDH 派生キーを KBPK として指定します。エクスポートするキーは、ラッピングキーがエクスポートするキーと少なくとも同じ強度である限り、TR-31 の有効な組み合わせの対象となる任意の TDES キーまたは AES キーにすることができます。

7. コールインポートキー

   KeyMaterialType が の `import-key` API を呼び出します`DiffieHellmanTr31KeyBlock`。にはステップ 3 でインポートされた最後の CA の KeyARN`certificate-authority-public-key-identifier`、 にはステップ 4 でラップされたキーマテリアル`key-material`、 にはステップ 3 のリーフ証明書を使用します`public-key-certificate`。ステップ 1 のプライベートキー ARN を含めます。

```
$ aws payment-cryptography import-key \
          --key-material='{
            "DiffieHellmanTr31KeyBlock": {
              "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-
cryptography:us-east-2:111122223333:key/swseahwtq2oj6zi5",
              "DerivationData": {
                "SharedInformation": "1234567890"
              },
              "DeriveKeyAlgorithm": "AES_256",
              "KeyDerivationFunction": "NIST_SP800",
              "KeyDerivationHashAlgorithm": "SHA_256",
              "PrivateKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/wc3rjsssguhxtilv",
              "PublicKeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN....",
              "WrappedKeyBlock":
 "D0112K1TB00E0000D603CCA8ACB71517906600FF8F0F195A38776A7190A0EF0024F088A5342DB98E2735084A7
            }
          }'
```

```
{
        "Key": {
          "CreateTimestamp": "2025-03-13T16:52:52.859000-04:00",
          "Enabled": true,
          "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
          "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_K1_KEY_ENCRYPTION_KEY"
          },
          "KeyCheckValue": "CB94A2",
          "KeyCheckValueAlgorithm": "ANSI_X9_24",
          "KeyOrigin": "EXTERNAL",
          "KeyState": "CREATE_COMPLETE",
          "UsageStartTimestamp": "2025-03-13T16:52:52.859000-04:00"
        }
      }
```


8. 暗号化オペレーションまたはそれ以降のインポートにインポートされたキーを使用する

   インポートされた KeyUsage が TR31_K0_KEY_ENCRYPTION_KEY の場合、TR-31 を使用した後続のキーインポートにこのキーを使用できます。他のキータイプ (TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY など) では、暗号化オペレーションにキーを直接使用できます。

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/ラップ解除をサポートしています。TR-34 と同様に、この手法は RSA 非対称暗号化を使用して、交換のために対称キーを暗号化します。ただし、TR-34 とは異なり、このメソッドでは送信側がペイロードに署名することはありません。また、この RSA ラップ手法では、キーブロックが含まれていないため、転送中にキーメタデータの整合性は維持されません。

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

1. Initialize Import コマンドを呼び出す

   get-parameters-for-import を呼び出して、 KeyMaterialTypeの を使用してインポートプロセスを初期化しますKEY_CRYPTOGRAM。TDES キーを交換するWrappingKeyAlgorithmときは、 RSA_2048に を使用します。TDES RSA_3072または AES-128 キーを交換するRSA_4096ときは、 または を使用します。この API は、キーインポート用のキーペアを生成し、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。このキーを使用してエクスポートするキーを暗号化します。これらの証明書は有効期間が短く、この目的にのみ使用されます。

   ```
   $ 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. キーソースシステムにパブリック証明書をインストールする

   多くの HSMs では、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、または信頼して、それを使用してキーをエクスポートする必要があります。

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 ラップを使用したキーのエクスポート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 () として指定しますWrappingKeyCertificate。信頼チェーンが必要な場合は、ステップ 1 WrappingKeyCertificateChainの を使用します。HSM からキーをエクスポートするとき

は、パディングモード = PKCS#1 v2.2 OAEP (SHA 256 または SHA 512 を使用) で RSA 形式を指定します。

4. 呼び出し import-key

KeyMaterialType の を使用して import-key API を呼び出しますKeyMaterial。ステップ 1 ImportTokenの 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 (キー使用状況など) を指定します。

```
$ 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"
    }
  }
}
```

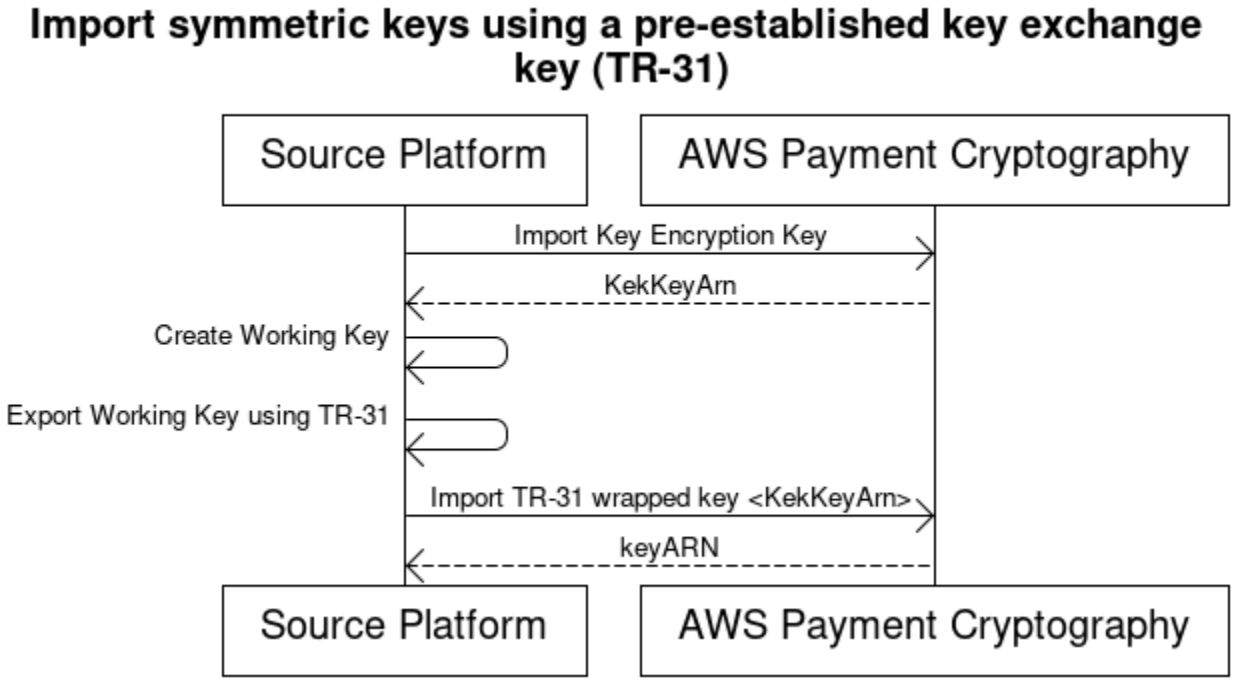
```
$ aws payment-cryptography import-key --cli-input-json file://import-key-
cryptogram.json
```

```
{
  "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またはそれ以降のインポートにインポートされたキーを使用する

   インポートされた KeyUsageが TR31_K0_KEY_ENCRYPTION_KEYまたは の場合TR31_K1_KEY_BLOCK_PROTECTION_KEY、TR-31 を使用した後続のキーインポートにこのキーを使用できます。キータイプが他のタイプ (などTR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY) の場合は、暗号化オペレーションに直接キーを使用できます。

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。



複数のキーを交換する場合やキーローテーションをサポートする場合、パートナーは通常、最初に初期キー暗号化キー (KEK) を交換します。これは、紙のキーコンポーネントなどの手法、または AWS Payment Cryptography の場合は TR-34 を使用して行うことができます。

KEK を確立したら、それ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 を使用したこのキー交換をサポートしています。

1. キー暗号化キーのインポート (KEK)

   KEK がインポート済みであり、keyARN (または keyAlias) が使用可能であることを確認します。
2. ソースプラットフォームでキーを作成する

   キーが存在しない場合は、ソースプラットフォームで作成します。または、 AWS Payment Cryptography で キーを作成し、 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 TR-31 として指定します。ソースプラットフォームは、エクスポートするキーと使用するキー暗号化キーを要求します。
4. AWS Payment Cryptography にインポートする

import-key コマンドを呼び出すときは、 のキー暗号化キーの keyARN (またはエイリアス) を使用しますWrappingKeyIdentifier。のソースプラットフォームからの出力を使用しますWrappedKeyBlock。

Example

```
$ aws payment-cryptography import-key \
    --key-material='{"Tr31KeyBlock": { \
    "WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ov6icy4ryas4zcza", \
    "WrappedKeyBlock":
 "D0112B0AX00E00002E0A3D58252CB67564853373D1EBCC1E23B2ADE7B15E967CC27B85D5999EF58E11662991F
 \
    }'
```

```
{
  "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、ECC) パブリックキーのインポート

インポートされるすべての証明書は、チェーン内の発行 (前身) 証明書と少なくとも同じ強度である必要があります。つまり、RSA_2048 CA は RSA_2048 リーフ証明書の保護にのみ使用でき、ECC 証明書は同等の強度の別の ECC 証明書で保護する必要があります。ECC P384 証明書は、P384 または P521 CA によってのみ発行できます。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。

RSA 公開キーのインポート

AWS Payment Cryptography では、パブリック RSA キーを X.509 証明書としてインポートできます。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します。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 形式であり、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

   ルート証明書をインポートするには、次のコマンドを使用します。

Example

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。TR-34 と ECDH は実行時にリーフ証明書を渡すため、このオプションは別のシステムからパブリックキーを使用してデータを暗号化する場合にのみ使用されます。KeyUsage は TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION に設定されます。

Example

```
$ aws payment-cryptography import-key \
    --key-material='{"Tr31KeyBlock": { \
    "WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ov6icy4ryas4zcza", \
    "WrappedKeyBlock":
 "D0112B0AX00E00002E0A3D58252CB67564853373D1EBCC1E23B2ADE7B15E967CC27B85D5999EF58E11662991F
 \
    }'
```

```
{
  "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"
  }
}
```

ECC パブリックキーのインポート

AWS Payment Cryptography では、パブリック ECC キーを X.509 証明書としてインポートできます。証明書をインポートするには、まずルート CA 証明書と中間証明書をインポートします。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 形式であり、base64 でエンコードされている必要があります。

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

   ルート証明書をインポートするには、次のコマンドを使用します。

Example

2. Payment Cryptography AWS に中間証明書をインポートする

   中間証明書をインポートするには、次のコマンドを使用します。

Example

3. パブリックキー証明書 (リーフ) を AWS Payment Cryptography にインポートする

   リーフ ECC 証明書をインポートすることはできますが、現在 AWS Payment Cryptography にはストレージ以外に定義された関数はありません。これは、ECDH 関数を使用する場合、リーフ証明書が実行時に渡されるためです。

キーのエクスポート

目次

対称キーのエクスポート

> **Important**
>
> 開始 AWS CLI する前に、 の最新バージョンがあることを確認してください。アップグレードするには、「 AWS CLIのインストール」を参照してください。

非対称技術 (TR-34) を使用してキーをエクスポートする

TR-34 は RSA 非対称暗号化を使用して、交換のために対称キーを暗号化して署名します。暗号化は機密性を保護し、署名は整合性を確保します。キーをエクスポートすると、 AWS Payment Cryptography がキー分散ホスト (KDH) として機能し、ターゲットシステムがキー受信デバイス (KRD) になります。

> **Note**
>
> HSM が TR-34 エクスポートをサポートしているがTR-34 インポートをサポートしていない場合は、まず TR-34 を使用して HSM と AWS Payment Cryptography 間で共有 KEK を確立することをお勧めします。その後、TR-31 を使用して残りのキーを転送できます。

1. エクスポートプロセスを初期化する

   get-parameters-for-export を実行して、キーエクスポートのキーペアを生成します。このキーペアを使用して TR-34 ペイロードに署名します。TR-34 の用語では、これは KDH 署名証明書です。証明書は有効期間が短く、 で指定された期間のみ有効ですParametersValidUntilTimestamp。

   > **Note**
   >
   > すべての証明書は base64 エンコードです。

   Example

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

   ```
   {
     "SigningKeyCertificate":
    "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2RENDQXFTZ0F3SUJ...",
     "SigningKeyCertificateChain": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS....",
     "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

   ステップ 1 の証明書チェーンを受信システムにインポートします。

3. 受信システムの証明書を設定する

送信されたペイロードを保護するために、送信側 (KDH) はペイロードを暗号化します。受信システム (通常は HSM またはパートナーの HSM) は、パブリックキーを生成し、X.509 パブリックキー証明書を作成する必要があります。 AWS Private CA を使用して証明書を生成できますが、任意の認証機関を使用できます。

証明書を取得したら、 ImportKey コマンドを使用してルート証明書を AWS Payment Cryptography にインポートします。KeyMaterialType を RootCertificatePublicKey に、KeyUsageType を TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE に設定します。

これはリーフ証明書に署名するルートキーKeyUsageTypeであるため、TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATUREとして を使用します。Payment AWS Cryptography にリーフ証明書をインポートする必要はありません。インラインで渡すことができます。

> **Note**
>
> 以前にルート証明書をインポートした場合は、このステップをスキップします。中間証明書の場合は、 を使用しますTrustedCertificatePublicKey。

4. キーをエクスポートする

を KeyMaterialTypeに設定して ExportKey API を呼び出しますTR34_KEY_BLOCK。以下を指定する必要があります。

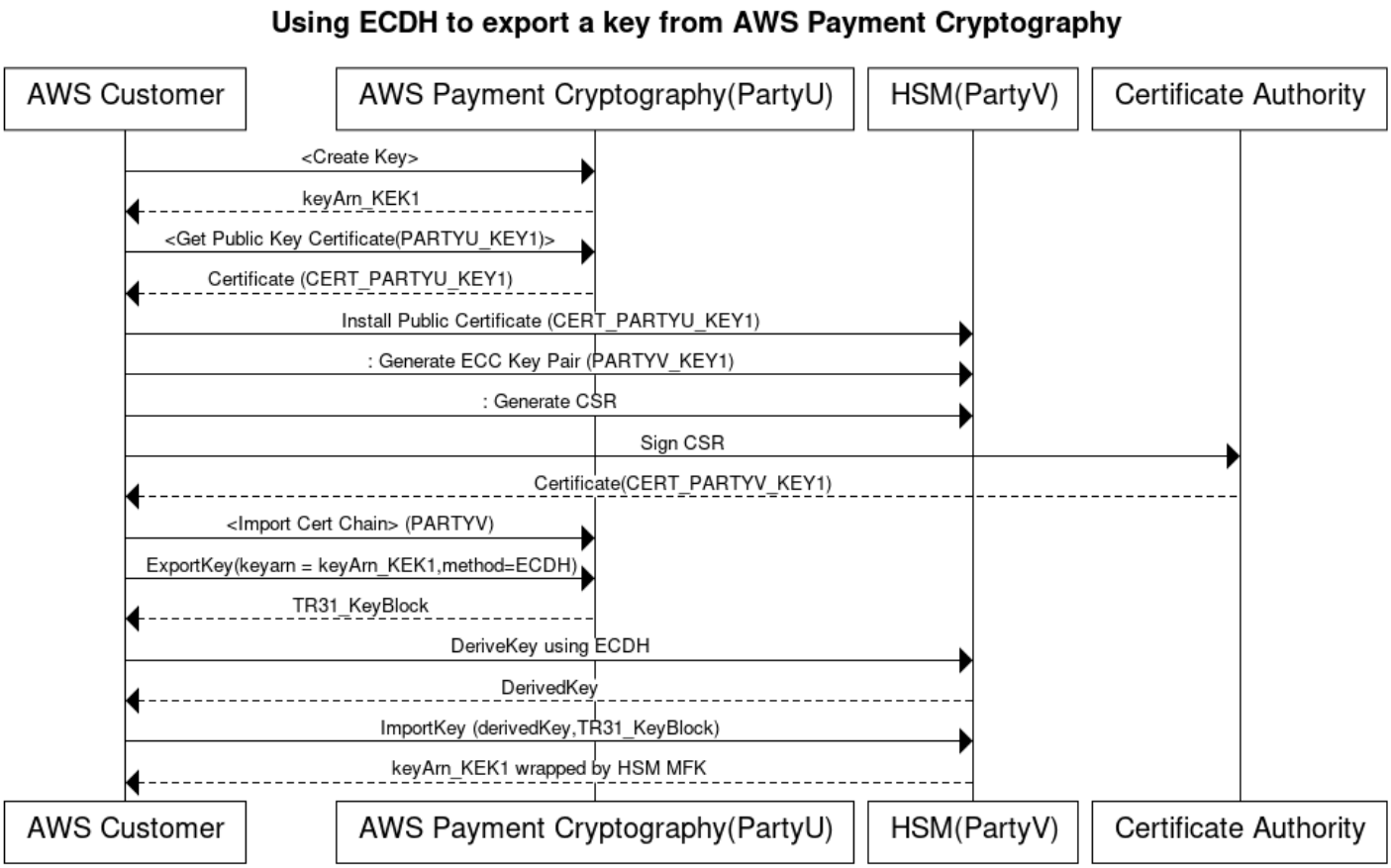
- ステップ 3 のルート CA の keyARN。 CertificateAuthorityPublicKeyIdentifier
- としてのステップ 3 のリーフ証明書 WrappingKeyCertificate
- としてエクスポートするキーの keyARN (またはエイリアス) --export-key-identifier
- ステップ 1 のエクスポートトークン

Example

```
$ aws payment-cryptography export-key \
    --export-key-identifier "example-export-key" \
    --key-material '{"Tr34KeyBlock": { \
    "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/4kd6xud22e64wcbk", \
    "ExportToken": "export-token-au7pvkbsq4mbup6i", \
    "KeyBlockFormat": "X9_TR34_2012", \
    "WrappingKeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2RENDQXFXZ0F3SUJBZ0lSQ..."} \
    }'
```

```
{
  "WrappedKey": {
    "KeyMaterial": "308205A106092A864886F70D010702A08205923082058...",
    "WrappedKeyMaterialFormat": "TR34_KEY_BLOCK"
  }
}
```

非対称手法 (ECDH) を使用したキーのエクスポート



Elliptic Curve Diffie-Hellman (ECDH) は、ECC 非対称暗号化を使用して、事前交換されたキーを必要とせずに、2 つの当事者間で共有キーを確立します。ECDH キーはエフェメラルであるため、AWS Payment Cryptography では保存されません。このプロセスでは、ECDH を使用して 1 回限りの KBPK/KEK が導出されます。この派生キーは、別の KBPK、BDK、IPEK キー、またはその他のキータイプなど、転送するキーをラップするためにすぐに使用されます。

エクスポートする場合、 AWS Payment Cryptography は Party U (Initiator) と呼ばれ、受信システムは Party V (Responder) と呼ばれます。

> **Note**
>
> ECDH は任意の対称キータイプの交換に使用できますが、KEK がまだ確立されていない場合に AES-256 キーの転送に使用できる唯一のアプローチです。

1. ECC キーペアの生成

を呼び出しcreate-keyて、このプロセスの ECC キーペアを作成します。この API は、キーのインポートまたはエクスポート用のキーペアを生成します。作成時に、この ECC キーを使用して派生できるキーの種類を指定します。ECDH を使用して他のキーを交換 (ラップ) する場合は、 の値を使用しますTR31_K1_KEY_BLOCK_PROTECTION_KEY。

> **Note**
>
> 低レベルの ECDH は、任意の目的に使用できる派生キーを生成しますが、 AWS Payment Cryptography は、キーを単一の派生キータイプにのみ使用できるようにすることで、複数の目的でキーを誤って再利用することを制限します。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=ECC_NIST_P256,KeyUsage=TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT,KeyClass=ASYM
 --derive-key-usage "TR31_K1_KEY_BLOCK_PROTECTION_KEY"
```

```
{
        "Key": {
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
wc3rjsssguhxtilv",
            "KeyAttributes": {
                "KeyUsage": "TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT",
                "KeyClass": "ASYMMETRIC_KEY_PAIR",
                "KeyAlgorithm": "ECC_NIST_P256",
                "KeyModesOfUse": {
                    "Encrypt": false,
                    "Decrypt": false,
                    "Wrap": false,
                    "Unwrap": false,
                    "Generate": false,
                    "Sign": false,
                    "Verify": false,
                    "DeriveKey": true,
                    "NoRestrictions": false
                }
            },
            "KeyCheckValue": "2432827F",
            "KeyCheckValueAlgorithm": "CMAC",
            "Enabled": true,
            "Exportable": true,
```

```
            "KeyState": "CREATE_COMPLETE",
            "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
            "CreateTimestamp": "2025-03-28T22:03:41.087000-07:00",
            "UsageStartTimestamp": "2025-03-28T22:03:41.068000-07:00"
        }
    }
```

2. パブリックキー証明書の取得

   get-public-key-certificate を呼び出して、特定のリージョンの AWS Payment Cryptography に固有のアカウントの CA によって署名された X.509 証明書としてパブリックキーを受け取ります。

   Example

```
$ aws payment-cryptography get-public-key-certificate \
           --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/wc3rjsssguhxtilv
```

```
{
      "KeyCertificate": "LS0tLS1CRUdJTi...",
      "KeyCertificateChain": "LS0tLS1CRUdJT..."
    }
```

3. カウンターパーティシステムにパブリック証明書をインストールする (当事者 V)

   多くの HSMs では、キーを確立するために、ステップ 1 で生成されたパブリック証明書をインストール、ロード、または信頼する必要があります。これには、HSM に応じて、証明書チェーン全体またはルート証明書のみが含まれます。具体的な手順については、HSM ドキュメントを参照してください。

4. ソースシステムで ECC キーペアを生成し、証明書チェーンを AWS Payment Cryptography に提供する

   ECDH では、各当事者がキーペアを生成し、共通のキーについて合意します。 AWS Payment Cryptography がキーを取得するには、X.509 パブリックキー形式のカウンターパーティのパブリックキーが必要です。

   HSM からキーを転送する場合は、その HSM にキーペアを作成します。キーブロックをサポートする HSMs、キーヘッダーは のようになりますD0144K3EX00E0000。証明書を作成するとき

は、通常、HSM で CSR を生成し、HSM、サードパーティー、 などのサービスが証明書を生成 AWS Private CA できます。

KeyMaterialType が RootCertificatePublicKey、KeyUsageType が の importKey コマンドを使用して、ルート証明書を AWS Payment Cryptography にロードしますTR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。

中間証明書の場合は、KeyMaterialType が TrustedCertificatePublicKey、KeyUsageType が の importKey コマンドを使用しますTR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。複数の中間証明書に対してこのプロセスを繰り返します。チェーンで最後にインポートされた証明書KeyArnの を、後続のエクスポートコマンドの入力として使用します。

> **Note**
>
> リーフ証明書をインポートしないでください。エクスポートコマンド中に直接指定します。

5. AWS Payment Cryptography からキーとエクスポートキーを取得する

エクスポート時に、サービスは ECDH を使用してキーを取得し、すぐにそれを KBPK として使用して、TR-31 を使用してエクスポートするキーをラップします。エクスポートするキーは、ラッピングキーがエクスポートするキーと少なくとも同じ強度である限り、TR-31 の有効な組み合わせの対象となる任意の TDES キーまたは AES キーにすることができます。

```
$ aws payment-cryptography export-key \
            --export-key-identifier arn:aws:payment-cryptography:us-
west-2:529027455495:key/e3a65davqhbpjm4h \
            --key-material='{
              "DiffieHellmanTr31KeyBlock": {
                "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-
cryptography:us-east-2:111122223333:key/swseahwtq2oj6zi5",
                "DerivationData": {
                  "SharedInformation": "ADEF567890"
                },
                "DeriveKeyAlgorithm": "AES_256",
                "KeyDerivationFunction": "NIST_SP800",
                "KeyDerivationHashAlgorithm": "SHA_256",
                "PrivateKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/wc3rjsssguhxtilv",
                "PublicKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FUR..."
```

```
            }
        }'
```

```
{
        "WrappedKey": {
            "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK",
            "KeyMaterial":
 "D0112K1TB00E00007012724C0FAAF64DA50E2FF4F9A94DF50441143294E0E995DB2171554223EAA56D078C4CF
            "KeyCheckValue": "E421AD",
            "KeyCheckValueAlgorithm": "ANSI_X9_24"
        }
    }
```

6. Party V HSM で ECDH を使用して 1 回限りのキーを取得する

   多くの HSMsおよび関連システムは、ECDH を使用したキーの確立をサポートしています。ステップ 1 のパブリックキーをパブリックキーとして指定し、ステップ 3 のキーをプライベートキーとして指定します。取得方法などの許容されるオプションについては、API ガイドを参照してください。

   > **Note**
   >
   > ハッシュタイプなどの取得パラメータは、両側で完全に一致する必要があります。それ以外の場合は、別のキーを生成します。

7. キーをターゲットシステムにインポートする

   最後に、標準の TR-31 コマンドを使用して AWS Payment Cryptography からキーをインポートします。ECDH 派生キーを KBPK として指定し、以前に AWS Payment Cryptography からエクスポートされた TR-31 キーブロックを使用します。

非対称手法を使用したキーのエクスポート (RSA ラップ)

TR-34 が使用できない場合は、RSA ラップ/ラップ解除をキー交換に使用できます。TR-34 と同様に、このメソッドは RSA 非対称暗号化を使用して対称キーを暗号化します。ただし、RSA ラップには以下は含まれません。

- 送信側によるペイロード署名
- 転送中にキーメタデータの整合性を維持するキーブロック

> **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 キーと証明書を作成する

   ラップされたキーを受信するための RSA キーを作成または識別します。キーは X.509 証明書形式である必要があります。証明書が AWS Payment Cryptography にインポートできるルート証明書によって署名されていることを確認します。

2. AWS Payment Cryptography にルートパブリック証明書をインポートする

   `--key-material` オプションimport-keyで を使用して証明書をインポートする

```
$ 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": "LS0tLS1CRUdJTiBDRV..."} \
    }'
```

```
{
  "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. キーをエクスポートする

   AWS Payment Cryptography 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指示します。以下を指定する必要があります。

   - ステップ 2 でインポートしたルート証明書の ARN
   - エクスポート用のリーフ証明書
   - エクスポートする対称キー

   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) バージョンです。

Example例 – キーのエクスポート

```
$ 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": "LS0tLS1CRUdJTiBDEXAMPLE...",
      "WrappingSpec": "RSA_OAEP_SHA_256"
    }
  }
}
```

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

```
{
  "WrappedKey": {
    "KeyMaterial":
 "18874746731E9E1C4562E4116D1C2477063FCB08454D757D81854AEAEE0A52B1F9D303FA29C02DC82AE778535
    "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 ラップ解除 (AWS Payment Cryptography を含む) を使用したキーのインポートをサポートしています。インポートするときは、以下を指定します。

- 暗号化証明書としてのステップ 1 のパブリックキー
- RSA としての形式
- PKCS#1 v2.2 OAEP としてのパディングモード (SHA 256 を使用)

> **Note**
>
> ラップされたキーは hexBinary 形式で出力されます。システムで base64 などの別のバイナリ表現が必要な場合は、 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

複数のキーを交換する場合やキーローテーションをサポートする場合は、通常、最初に紙のキーコンポーネントを使用して初期キー暗号化キー (KEK) を交換するか、 AWS Payment Cryptography で TR-34 を使用します。KEK を確立したら、それを使用して、他の KEKs を含む後続のキーを転送できます。このキー交換は、HSM ベンダーによって広くサポートされている ANSI TR-31 を使用してサポートされています。

1. キー暗号化キー (KEK) の設定

   すでに KEK を交換しており、keyARN (または keyAlias) が使用可能であることを確認します。

2. AWS Payment Cryptography でキーを作成する

   キーがまだ存在しない場合は作成します。または、他のシステムでキーを作成し、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

   TR-31 形式でエクスポートする場合は、エクスポートするキーと使用するラッピングキーを指定します。

Example例 – TR31 キーブロックを使用したキーのエクスポート

```
$ 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":
 "D0144K0AB00E0000A24D3ACF3005F30A6E31D533E07F2E1B17A2A003B338B1E79E5B3AD4FBF7850FACF9A3784
    "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK"
  }
}
```

4. システムにキーをインポートする

   システム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 つのベース導出キー (BDK) を生成できます。ターミナルは BDK に直接アクセスできません。代わりに、各ターミナルは 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を受け取ります。各 IPEK は、一意のキーシリアル番号 (KSN) を使用して BDK から取得されます。

KSN 構造は暗号化タイプによって異なります。

- TDES の場合: 10 バイトの KSN には以下が含まれます。
  - キーセット ID の 24 ビット
  - ターミナル ID の 19 ビット
  - トランザクションカウンターの 21 ビット
- AES の場合: 12 バイトの KSN には以下が含まれます。
  - BDK ID の 32 ビット

- 取得識別子 (ID) の 32 ビット
- トランザクションカウンターの 32 ビット

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が用意されています。生成されたキーは、TR-31, TR-34、または RSA 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 で後続のオペレーションには使用できません。

KSN の最初の 2 つの部分間の分割は強制しません。取得識別子を BDK に保存する場合は、 AWS タグを使用できます。

> **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。たとえば、12345678901234560001 と 12345678901234569999 の入力は同じ IPEK を生成します。

```
$ 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"
}
}
```

エクスポート用のキーブロックヘッダーを指定する

ASC TR-31 または TR-34 形式でエクスポートするときに、キーブロック情報を変更または追加できます。次の表は、TR-31 キーブロック形式と、エクスポート中に変更できる要素を示しています。

キーブロック属性	目的	エクスポート中に を変更できますか?	注意事項
バージョン ID	キーマテリアルを保護するために使用される方法を定義します。標準には以下が含まれます。 • バージョン A と C (キーバリアント - 非推奨) • バージョン B (TDES を使用した取得) • バージョン D (AES を使用したキー取得)	いいえ	TDES ラッピングキーにはバージョン B、AES ラッピングキーにはバージョン D を使用します。インポートオペレーションでのみバージョン A と C がサポートされています。
キーブロックの長さ	残りのメッセージの長さを指定します。	いいえ	この値は自動的に計算されます。指定に応じてキーパディングを追加する可能性があるため、ペイロードを復号する前に長さが正しく表示されないことがあります。
キーの使用方法	キーで許可される目的を定義します。次に例を示します。 • C0 (カード検証) • B0 (ベース導出キー)	いいえ	

キーブロック属性	目的	エクスポート中に を変更できますか?	注意事項
アルゴリズム	基になるキーのアルゴリズムを指定します。以下がサポートされています。 • T (TDES) • H (HMAC) • A (AES)	いいえ	この値はそのままエクスポートします。
キーの使用方法	次のような許可されたオペレーションを定義します。 • 生成と検証 (C) • Encrypt/Decrypt/Wrap/ラップ解除 (B)	はい*	
キーバージョン	キーの置換/ローテーションのバージョン番号を示します。指定しない場合、デフォルトは 00 です。	はい - 追加できます	

キーブロック属性	目的	エクスポート中に を変更できますか?	注意事項
キーのエクスポート可能性	キーをエクスポートできるかどうかを制御します。 • N - エクスポート不可 • E - X9.24 に従ってエクスポートする (キーブロック) • S - キーブロック形式または非キーブロック形式でエクスポートする	はい*	
オプションのキーブロック	はい - 追加できます	オプションのキーブロックは、キーに暗号的にバインドされた名前と値のペアです。たとえば、DUKPT キーの KeySetID などです。名前と値のペアの入力に基づいて、ブロック数、各ブロックの長さ、パディングブロック (PB) が自動的に計算されます。	

* 値を変更する場合、新しい値は AWS Payment Cryptography の現在の値よりも制限されている必要があります。例えば、次のようになります。

- 現在のキーモードが Generate=True、Verify=True の場合、Generate=True、Verify=False に変更できます。

- キーがすでにエクスポート不可に設定されている場合は、エクスポート可能に変更することはできません。

キーをエクスポートすると、エクスポートするキーから現在の値が自動的に適用されます。ただし、受信システムに送信する前に、これらの値を変更または追加する場合があります。一般的なシナリオをいくつか紹介します。

- キーを支払いターミナルにエクスポートする場合、ターミナルは通常キーをインポートするだけであり、エクスポートすべきではないNot Exportableため、そのエクスポート可能性を に設定します。
- 関連付けられたキーメタデータを受信システムに渡す必要がある場合は、TR-31 オプションのヘッダーを使用して、カスタムペイロードを作成する代わりにメタデータをキーに暗号的にバインドします。
- キーローテーションを追跡するには、 KeyVersionフィールドを使用してキーバージョンを設定します。

TR-31/X9.143 は一般的なヘッダーを定義しますが、 AWS Payment Cryptography パラメータを満たし、受信システムが受け入れることができる限り、他のヘッダーを使用できます。エクスポート中のキーブロックヘッダーの詳細については、 API ガイドの「キーブロックヘッダー」を参照してください。

次の仕様で BDK キー (KIF など) をエクスポートする例を示します。

- キーバージョン: 02
- KeyExportability: NON_EXPORTABLE
- KeySetID: 00ABCDEFAB (00 は TDES キー、ABCDEFABCD は初期キーを示します)

キーモードは指定されていないため、このキーは arn:aws:payment-cryptography:us-east-2:111122223333:key/5rplquuwozodpwsp (DeriveKey = true) から使用モードを継承します。

> **Note**
>
> この例でエクスポート可能性をエクスポート不可に設定しても、KIF は引き続き以下を実行できます。
>
> - DUKPT で使用される IPEK/IK などのキーを取得する

- これらの派生キーをエクスポートしてデバイスに をインストールする

これは、 標準で特に許可されています。

```
$ aws payment-cryptography export-key \
    --key-material='{"Tr31KeyBlock": { \
    "WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza", \
    "KeyBlockHeaders": { \
    "KeyModesOfUse": { \
    "Derive": true}, \
    "KeyExportability": "NON_EXPORTABLE", \
    "KeyVersion": "02", \
    "OptionalBlocks": { \
    "BI": "00ABCDEFABCD"}}} \
    }' \
    --export-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp
```

```
{
"WrappedKey": {
    "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK",
    "KeyMaterial": "EXAMPLE_KEY_MATERIAL_TR31",
    "KeyCheckValue": "A4C9B3",
    "KeyCheckValueAlgorithm": "ANSI_X9_24"
    }
}
```

一般的なヘッダー

X9.143 は、一般的なユースケース用に特定のヘッダーを定義します。HM(HMAC Hash) ヘッダーを除き、 AWS Payment Cryptography はこれらのヘッダーを解析または使用しません。

ヘッダー名	目的	一般的な検証	注意事項
BI	DUKPT の基本派生キー識別子	2 つの 16 進数文字 (TDES の場合は 00、AES の場合は 11	(AES DUKPT の場合は BDK ID) またはキーセット識別子

ヘッダー名	目的	一般的な検証	注意事項
		文字）、TDES KSI の場合は 10 進数文字、または BDK ID (AES DUKPT) の場合は 8 進数文字。	(KSI、TDES DUKPT の場合は KSI) が含まれます。BDK ID または KSI を交換するときに使用できますが、IK ブロックと KS ブロックに含まれる他のデータを交換する必要はありません。通常、BI は KIF に送信するときに使用され、IK または KS はターミナル自体に挿入するときに使用されます。
HM	HMAC オペレーションのハッシュタイプを指定します。	• 10 – SHA-1 • 20 – SHA-224 • 21 – SHA-256 • 22 – SHA-384 • 23 – SHA-512 • 24 – SHA-512/224 • 25 – SHA-512/256 • 30 – SHA3-224 • 31 – SHA3-256 • 32 – SHA3-384 • 33 – SHA3-512 • 40 – SHAKE128 • 41 – SHAKE256	エクスポート時にこのフィールドが自動的に入力され、インポート時に解析されます。SHAKE128 などのサービスでサポートされていないハッシュタイプはインポートできますが、暗号化関数に使用できない場合があります。

ヘッダー名	目的	一般的な検証	注意事項
IK	AES DUKPT の初期キーシリアル番号	16 進数文字	この値は、受信デバイス上の初期 DUKPT キーの使用をインスタンス化するために使用され、BDK から派生した初期キーを識別します。このフィールドには通常、取得データが含まれますが、カウンターは含まれません。TDES DUKPT には KS を使用します。
KS	TDES DUKPT の初期キーシリアル番号	20 文字の 16 進数	この値は、受信デバイス上の初期 DUKPT キーの使用をインスタンス化するために使用され、BDK から派生した初期キーを識別します。このフィールドには通常、取得データ + ゼロ化されたカウンター値が含まれます。AES DUKPT には IK を使用します。

ヘッダー名	目的	一般的な検証	注意事項
KP	ラッピングキーの KCV	2 つの 16 進文字は KCV メソッドを表します (X9.24 メソッドの場合は 00、CMAC メソッドの場合は 01)。続いて KCV 値が続きます。通常は 6 進数文字です。たとえば、010FA329 は、01 (CMAC) メソッドを使用して計算された 0FA329 の KCV を表します。	この値は、受信デバイス上の初期 DUKPT キーの使用をインスタンス化するために使用され、BDK から派生した初期キーを識別します。このフィールドには通常、取得データ + ゼロ化されたカウンター値が含まれます。AES DUKPT には IK を使用します。
PB	パディングブロック	ランダムな印刷可能な ASCII 文字	サービスはエクスポート時にこのフィールドを自動的に入力し、オプションのヘッダーが暗号化ブロックの長さの倍数になるようにします。

非対称 (RSA) キーのエクスポート

証明書形式でパブリックキーをエクスポートするには、 get-public-key-certificate コマンドを使用します。このコマンドは以下を返します。

- 証明書
- ルート証明書

両方の証明書は base64 エンコードです。

> **Note**
>
> このオペレーションはべき等ではありません。後続の呼び出しでは、基盤となる同じキーを使用している場合でも、異なる証明書を生成す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..."
}
```

高度なトピック

このセクションでは、高度なキー交換のシナリオと設定について説明します。

トピック

- Bring Your Own Certificate Authority (BY")

Bring Your Own Certificate Authority (BY")

デフォルトでは、サービス内で作成された非対称 (RSA,ECC) キーにパブリックキー証明書が必要な場合、これらの証明書は AWS Payment Cryptography とアカウント固有の認証機関 (CA) によって発行されます。これは、CA を特定または設定したり、 Certificate Signing Requests (CSR) を管理したりすることなく、X.509 を簡単に使用できるようにすることを目的としています。

AWS Payment Cryptography は、ポリシーまたはコンプライアンスの理由で必要な場合に、独自の CA を使用することもできます。

概要:

BY" 機能を使用すると、TR-34 インポート/エクスポート、RSA Unwrap、ECDH ベースのキー転送など、証明書が使用されている任意の場所で独自の認証機関を使用できます。これは、組織全体で一

貫した証明書チェーンを維持する必要がある場合や、特定の CA 証明書を必要とするパートナーと連携する場合に便利です。次の例は、TR-34 キーエクスポートを使用した BY" ワークフローを示しています。

標準の TR-34 エクスポートフローと比較した 3 つの主な違いは次のとおりです。

1. 署名 RSA キーは CreateKey を使用して明示的に作成されます。以前は、GetParametersForExport を介して暗黙的に作成されていました。
2. 新しい API GetCertificateSigningRequest は、外部 CA によって署名できる Certificate Signing Request (CSR) を作成します。
3. ExportKey API は、実行時に証明書を提供できるように拡張されています。以前は、これは暗黙的に によって提供されimport-tokenていました。これはオプションフィールドになります。

> ⚠ 重要な考慮事項
>
> - これらの例では、RSA-2048 キーを使用して TDES-2KEY キーをラップします。AES-128 をエクスポートするときは、すべてのキーが RSA-3072 または RSA-4096 であることを確認します。
> - 最も一般的なエラーは、 SigningKeyIdentifierと で表されるキーSigningKeyCertificateが一致しないことです。

BY" ワークフロー

次の手順は、TR-34 エクスポートの完全な BY" ワークフローを示しています。

Steps

- ステップ 1: RSA キーを作成する
- ステップ 2: 証明書署名リクエストを生成する
- ステップ 3: CSR を確認する (オプション)
- ステップ 4: 認証機関で CSR に署名する
- ステップ 5: CA 証明書をインポートする
- ステップ 6: KRD 暗号化証明書を取得する
- ステップ 7: BY" を使用してキーをエクスポートする

ステップ 1: RSA キーを作成する

まず、最終的に KDH 署名証明書となる RSA キーペアを作成します。タグを追加して、キーの目的を特定できます。

Example署名用の RSA キーを作成する

```
$ aws payment-cryptography create-key --exportable \
    --key-attributes
 KeyAlgorithm=RSA_2048,KeyUsage=TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE,KeyClass=ASYMMETRI
```

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
xgmq6fs6uow736uc",
        "KeyAttributes": {
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE",
            "KeyClass": "ASYMMETRIC_KEY_PAIR",
            "KeyAlgorithm": "RSA_2048",
            "KeyModesOfUse": {
                "Sign": true
            }
        },
        "KeyCheckValue": "41E3723C",
        "KeyCheckValueAlgorithm": "SHA_1",
        "Enabled": true,
        "Exportable": true,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY"
    }
}
```

次のステップで必要になるKeyArnため、 に注意してください。

ステップ 2: 証明書署名リクエストを生成する

GetCertificateSigningRequest API を使用して、外部 CA によって署名される証明書署名リクエスト (CSR) を生成します。出力は base64 でエンコードされた PEM ファイルです。base64 でコンテンツをデコードして保存すると、PEM 形式で有効な CSR が作成されます。

Example CSR の生成

```
$ aws payment-cryptography-data get-certificate-signing-request \
```

```
    --key-identifier arn:aws:payment-cryptography:us-east-1:111122223333:key/
xgmq6fs6uow736uc \
    --signing-algorithm SHA512 \
    --certificate-subject '{
        "CommonName": "MyCertificateAWSUSEAST",
        "Organization": "Amazon",
        "OrganizationUnit": "PaymentCryptography",
        "Country": "US",
        "StateOrProvince": "Virginia",
        "City": "Arlington"
    }'
```

```
{
    "CertificateSigningRequest": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0..."
}
```

`CertificateSigningRequest` フィールドには、署名のために CA に送信する base64 エンコードされた CSR が含まれています。

ステップ 3: CSR を確認する (オプション)

オプションで OpenSSL を使用して CSR の内容を確認し、有効で期待どおりに動作することを確認できます。

Example OpenSSL で CSR を確認する

```
$ echo "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0..." | base64 -d | openssl req -
text
```

ステップ 4: 認証機関で CSR に署名する

CSR を生成したら、認証機関 (CA) による署名が必要です。本番環境では、通常、 AWS Private CA または組織が確立した CA インフラストラクチャを使用します。テスト目的で、OpenSSL を使用して自己署名証明書を作成できます。

の使用 AWS Private CA

を使用して CSR に署名するには AWS Private CA、まず base64 でエンコードされた CSR をデコードし、ファイルに保存します。次に IssueCertificate API を使用します。

Exampleで CSR に署名する AWS Private CA

```
$ echo "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0..." | base64 -d > csr.pem

$ aws acm-pca issue-certificate \
    --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/12345678-1234-1234-1234-123456789012 \
    --csr fileb://csr.pem \
    --signing-algorithm SHA256WITHRSA \
    --validity Value=365,Type=DAYS
```

```
{
    "CertificateArn": "arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/12345678-1234-1234-1234-123456789012/certificate/abcdef1234567890"
}
```

次に、署名付き証明書を取得します。

Example署名付き証明書の取得

```
$ aws acm-pca get-certificate \
    --certificate-authority-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/12345678-1234-1234-1234-123456789012 \
    --certificate-arn arn:aws:acm-pca:us-east-1:111122223333:certificate-
authority/12345678-1234-1234-1234-123456789012/certificate/abcdef1234567890
```

```
{
    "Certificate": "-----BEGIN CERTIFICATE-----\nMIID...\n-----END CERTIFICATE-----",
    "CertificateChain": "-----BEGIN CERTIFICATE-----\nMIID...\n-----END
 CERTIFICATE-----"
}
```

エクスポートステップで使用する証明書コンテンツを保存します。ExportKey API に提供するときは、base64 エンコードする必要があります。

OpenSSL を使用したテスト

テスト目的で、OpenSSL を使用して自己署名 CA を作成し、CSR に署名できます。まず、CA プライベートキーと自己署名証明書を作成します。

Example OpenSSL を使用してテスト CA を作成する

```
$ # Generate CA private key
openssl genrsa -out ca-key.pem 4096

$ # Create self-signed CA certificate
openssl req -new -x509 -days 3650 -key ca-key.pem -out ca-cert.pem \
    -subj "/C=US/ST=Virginia/L=Arlington/O=TestOrg/CN=Test CA"
```

次に、前のステップから CSR をデコードし、テスト CA で署名します。

Example OpenSSL で CSR に署名する

```
$ # Decode the base64-encoded CSR
echo "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0..." | base64 -d > csr.pem

$ # Sign the CSR with the CA
openssl x509 -req -in csr.pem -CA ca-cert.pem -CAkey ca-key.pem \
    -CAcreateserial -out signed-cert.pem -days 365 -sha512
```

```
Certificate request self-signature ok
subject=C=US, ST=Virginia, L=Arlington, O=Amazon, OU=PaymentCryptography,
 CN=MyCertificateAWSUSEAST
```

署名付き証明書が になりましたsigned-cert.pem。ExportKey API に提供するときは、この証明書を base64 でエンコードする必要があります。

Example Base64 署名付き証明書をエンコードする

```
$ cat signed-cert.pem | base64 -w 0
```

ステップ 5: CA 証明書をインポートする

使用する CA は、任意の証明書が使用されないように最初に信頼する必要があります。ImportKey API を使用して外部 CA のルート証明書をインポートします。中間 CA を使用する場合は、 import-key を再度呼び出しますが、 TrustedPublicKeyの代わりに を指定RootCertificatePublicKeyし、ルート CA ARN を指定します。

Exampleルート CA 証明書をインポートする

```
$ 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": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t..."
    }
}'
```

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
xivpaqy7qbbm7cdw",
        "KeyAttributes": {
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE",
            "KeyClass": "PUBLIC_KEY",
            "KeyAlgorithm": "RSA_4096",
            "KeyModesOfUse": {
                "Verify": true
            }
        },
        "Enabled": true,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "EXTERNAL"
    }
}
```

エクスポートステップKeyArnで使用する CA を書き留めます。

ステップ 6: KRD 暗号化証明書を取得する

この例では、 AWS Payment Cryptography にインポートし直すため、 サービスを呼び出して、GetParametersForImport API を使用して KRD パブリックキー証明書を受け取ります。実際のシナリオでは、これは HSM、ATM、支払い端末、支払い端末管理システムなどの他のシステムによって提供されます。

Exampleインポートのパラメーターを取得する

```
$ aws payment-cryptography-data get-parameters-for-import \
```

```
    --key-material-type "TR34_KEY_BLOCK" \
    --wrapping-key-algorithm RSA_2048
```

```
{
    "WrappingKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...",
    "WrappingKeyCertificateChain": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...",
    "WrappingKeyAlgorithm": "RSA_2048",
    "ImportToken": "import-token-v2rxpl6drxeptn7w",
    "ParametersValidUntilTimestamp": "2025-11-01T18:45:31.271000-07:00"
}
```

ステップ 7: BY" を使用してキーをエクスポートする

最後に、ExportKey API を使用して独自の CA 署名証明書で TR-34 を使用してキーをエクスポートします。外部 CA によって署名された署名証明書を指定します。

Example BY" を使用した TR-34 エクスポート

```
$ aws payment-cryptography-data export-key \
    --export-key-identifier arn:aws:payment-cryptography:us-east-1:111122223333:key/
iox73p5f4c4yjiod \
    --key-material '{
        "Tr34KeyBlock": {
            "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-
cryptography:us-east-1:111122223333:key/j625deyfqlwctu57",
            "SigningKeyIdentifier": "arn:aws:payment-cryptography:us-
east-1:111122223333:key/xgmq6fs6uow736uc",
            "SigningKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...",
            "KeyBlockFormat": "X9_TR34_2012",
            "WrappingKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t..."
        }
    }'
```

```
{
    "WrappedKey": {
        "WrappedKeyMaterialFormat": "TR34_KEY_BLOCK",
        "KeyMaterial": "3082055A06092A864886F70D010702A082054B30820547...",
        "KeyCheckValue": "3DCA31",
        "KeyCheckValueAlgorithm": "ANSI_X9_24"
    }
}
```

エクスポートされたキーブロックは、標準の TR-34 インポートプロセスを使用して受信システムによってインポートできるようになりました。

追加のメモ

- これらの例は、AWS CLI を使用して示されています。Java、Python、Go、Rust を含むすべての AWS SDKs で同じ機能を使用できます。
- 自己署名 CA でテストする場合は、OpenSSL を使用してテスト CA を作成し、CSR に署名できます。本番環境では、組織で確立された CA インフラストラクチャを使用します。

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。

エイリアスを使用して、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。たとえば、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。複数のリージョンで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に関連付けられた AWS Payment Cryptography キーを参照します。

AWS Payment Cryptography キーのエイリアスは、 `CreateAlias` API を使用して作成できます。

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 (list-aliases)

の表示、エイリアスに関連付けられた AWS Payment Cryptography キーの変更 (update-alias)、エイリアスの削除 (delete-alias) 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

AWS Payment Cryptography でのエイリアスの仕組み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。たとえば、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てい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に、キーの使用状況など同じ属性がすべて含まれてい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、一部のキーにはエイリアスが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で 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ります。

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か、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 1 つの暗号化マテリアルを表し、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 KeyIdentifier を入力として受け取り、属性、状態、タイムスタンプなどのキーメタデータを返しますが、実際の暗号化キーマテリアルは返しません。

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":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThtZ0F3SUJBZ0lSQUtlN2piaHFKZjJPd3FGUWI5c3VuC
}
```

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、削除が保留中でない限り、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 AWS ることができる) 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、請求を含め AWS のサービス、多くのユーザー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、削除が保留中でない限り、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。

たとえば、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 サービスはタグ付けをサポートしているため、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。たとえば、 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 タグに基づく認可」を参照してください。タグを使用して AWS 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「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。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
- ListResourceTags: 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。タグを使用する場合は、CreateKey オペレーションの Tags パラメータを指定します。

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。

たとえば、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、ListResourceTags 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。タグをキーで表示するには、ListResourceTags 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"}]'
```

ListResourceTags: キーのタグを取得する

ListResourceTags オペレーションでは、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 オペレーションでは、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。オペレーションが正常に機能したことを確認するには、 ListResourceTags 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 および UntagResource のようなタグ付けオペレーションへのアクセスを制御できます。

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。

JSON

```
{
  "Version":"2012-10-17",
  "Id": "example-key-policy",
  "Statement": [
```

```
    {
      "Sid": "EnableIAMUserPermissions",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
      "Action": "payment-cryptography:*",
      "Resource": "*"
    },
    {
      "Sid": "AllowAllTaggingPermissions",
      "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:ListTagsForResource",
      "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 オペレーションのタグパラメータを使用して、キー作成時にキーにタグを追加できます。

JSON

```
{
  "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 リクエストは複数のタグを含む可能性があるため、ForAllValues または ForAnyValue 集合演算子をaws:TagKeys 条件で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 演算子はリクエストにタグがない場合も true を返しますが、TagResource と UntagResource は、タグが指定されていない場合、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。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKey",
      "Effect": "Allow",
      "Action": "payment-cryptography:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyViewAllTags",
      "Effect": "Allow",
      "Action": "payment-cryptography:ListTagsForResource",
      "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

Payment Cryptography AWS 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「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。

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。これは、1 つのキーだけに適用されるキーポリシー、または特定のキーを使用しないオペレーション (ListKeys または ListAliases オペレーションなど) ではサポートされません。

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 AWS アカウント のアクセス許可を持っている 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 ログおよび CloudWatch アラームを使用して、KMS キーへのアクセスに影響する可能性のある変更をタグ付けするようにアラートします。
- 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。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
      "Effect": "Allow",
```

```
      "Action": [
        "payment-cryptography:DecryptData"
      ],
      "Resource": "arn:aws:payment-cryptography: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。

JSON

```
{
  "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、ここでは便利なように有効な組み合わせが用意されています。

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
  - 使用可能なキーモードの組み合わせ: { DeriveKey = true },{ NoRestrictions = true }
- TR31_C0_CARD_VERIFICATION_KEY

  - 許可されるキーアルゴリズム:
    TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 使用可能な主な使用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 } ,{ NoRestrictions = true }
- TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS
  - 許可されるキーアルゴリズム:
    TDES_2KEY 、TDES_3KEY*、AES_128*、AES_192*、AES_256*
  - 使用可能なキーモードの組み合わせ:{DeriveKey = true}, { Payment Cryptography = true}
- TR31_E1_EMV_MKEY_CONFIDENTIALITY
  - 許可されるキーアルゴリズム: TDES_2KEY 、TDES_3KEY、AES_128*、AES_192*、AES_256*
  - 使用可能なキーモードの組み合わせ:{DeriveKey = true}, { Payment Cryptography = true}
- TR31_E2_EMV_MKEY_INTEGRITY
  - 許可されるキーアルゴリズム:
    TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 使用可能なキーモードの組み合わせ:{DeriveKey = true}, { Payment Cryptography = true}
- TR31_E4_EMV_MKEY_DYNAMIC_NUMBER
  - 許可されるキーアルゴリズム:
    TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 使用可能なキーモードの組み合わせ:{DeriveKey = true}, { Payment Cryptography = true}
- TR31_E5_EMV_MKEY_CARD_PERSONALIZATION
  - 許可されるキーアルゴリズム:
    TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 使用可能なキーモードの組み合わせ:{DeriveKey = true}, { Payment Cryptography = true}
- TR31_E6_EMV_MKEY_OTHER
  - 許可されるキーアルゴリズム:
    TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 使用可能なキーモードの組み合わせ:{DeriveKey = true}, { Payment Cryptography = true}
- TR31_K0_KEY_ENCRYPTION_KEY

  - TR31_K1_KEY_BLOCK_PROTECTION_KEY を使用することをお勧めします。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 } ,{ 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 } ,{ 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
  - 使用可能な主な使用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
  - 使用可能な主な使用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
  - 使用可能な主な使用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
  - 使用可能な主な使用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 } ,{ 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
  - 使用可能な主な使用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
  - 使用可能な主な使用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ときの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。
- TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT
  - ECDH などのキーアグリーメントアルゴリズムに使用されます
  - 許可されるキーアルゴリズム: ECC_NIST_P256、ECC_NIST_P384、ECC_NIST_P521
  - 使用できるキーモードの組み合わせ: { DeriveKey = true }。
  - 注:DeriveKeyUsage は、このベースキーから派生するキーの種類を指定するために使用されます。これはキーの作成/インポート時に修正されます。
- TR31_K2_TR34_ASYMMETRIC_KEY
  - TR-34 などの X9.24 互換キー交換メカニズムに使用される非対称キー
  - 許可され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: { DeriveKey = true }。
  - 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ときの唯一の有効なオプションです

* このアルゴリズムとキータイプの組み合わせは現在、暗号化オペレーションではサポートされていません

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。ハッシュとドメイン固有のアルゴリズムも同様に、対称キーまたはパブリックキーなしでは検証できません。

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、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 手法を使用して派生し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するユースケースでは、ReEncrypt コマンドを使用することもできます。

> **Note**
>
> 暗号化/復号関数を使用する場合、すべての入力は hexBinary にあると見なされます。例えば、1 の値は 31 (16 進数) として入力され、小文字の t は 74 (16 進数) として表されます。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、「 API Guide for Encrypt, Decrypt, and Re-Encrypt」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` API は、対称および非対称データ暗号化キーと、DUKPT および EMV 派生キーを使用してデータを暗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。

プライマリ入力は、データの暗号化に使用される暗号化キー、暗号化する hexBinary 形式のプレーンテキストデータ、初期化ベクトルや TDES などのブロック暗号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、 の場合は 8 バイトTDES、 の場合は 16 バイト、 の場合はキーAESの長さの倍数である必要がありますRSA。対称キー入力 (TDES、AES、DUKPT、EMV) は、入力データがこれらの要件を満たしていない場合に備えて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`EncryptionAttributes`用に で定義したパディングタイプを示しています。

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、「 API Guide for Encrypt」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 派生対称キーを使用してデータを暗号化する
- 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**
>
> すべての例では、関連するキーが既に存在することを前提としています。キーは CreateKey オペレーションを使用して作成することも、ImportKey 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。

Example

この例では、CreateKey オペレーションを使用して作成された、または ImportKey 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。このオペレーションでは、キーの KeyModesOfUse が Encrypt に設定され、KeyUsage が TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY に設定されている必要があります。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 Payment Cryptography は、 TDESおよび AES DUKPT キーをサポートしています。このオペレーションでは、キーの KeyModesOfUse が DeriveKey に設定され、KeyUsage が TR31_B0_BASE_DERIVATION_KEY に設定されている必要があります。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 派生対称キーを使用してデータを暗号化する

Example

この例では、作成済みの EMV 派生対称キー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。このオペレーションでは、キーの KeyModesOfUse を に設定Deriveし、KeyUsage を TR31_E1_EMV_MKEY_CONFIDENTIALITYまたは に設定する必要がありますTR31_E6_EMV_MKEY_OTHER。詳細については、「暗号化オペレーションのキー」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、ImportKey オペレーションを使用してインポートされた RSA 公開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。このオペレーションでは、キーの KeyModesOfUse が Encrypt に設定され、KeyUsage が TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION に設定されている必要があります。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-1:111122223333:key/5dza7xqd6soanjtb",
 "KeyCheckValue": "FF9DE9CE"
}
```

データの復号化

`Decrypt Data` API は、対称および非対称データ暗号化キーと、DUKPT および EMV 派生キーを使用してデータを復号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、「Decrypt 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 派生対称キーを使用したデータの復号
- 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、対称キーを使用して暗号文データを復号します。この例では、 AESキーを示していますが、 TDES_2KEYと もサポートTDES_3KEYされています。このオペレーションでは、キーの KeyModesOfUse が Decrypt に設定され、KeyUsage が TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY に設定されている必要があります。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、CreateKey オペレーションを使用して作成された、または ImportKey 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。このオペレーションでは、キーの KeyModesOfUse が DeriveKey に設定され、KeyUsage が TR31_B0_BASE_DERIVATION_KEY に設定されている必要があります。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 派生対称キーを使用したデータの復号

Example

この例では、CreateKey オペレーションを使用して作成されたか、ImportKey オペレーションを使用してインポートされた EMV 派生対称キーを使用して暗号文データを復号します。このオペレーションでは、キーの KeyModesOfUse を に設定Deriveし、KeyUsage を TR31_E1_EMV_MKEY_CONFIDENTIALITYまたは に設定する必要がありますTR31_E6_EMV_MKEY_OTHER。詳細については、「暗号化オペレーションのキー」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": "31323334313233343132333431323334"
}
```

RSA キーを使用してデータを復号化します。

Example

この例では、CreateKey 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。このオペレーションでは、キーの KeyModesOfUse が Decrypt に設定され、KeyUsage が TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION に設定されている必要があります。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-1:111122223333:key/5dza7xqd6soanjtb",
    "KeyCheckValue": "FF9DE9CE",
    "PlainText": "313233343132333431323334313233343132333431323334"
}
```

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 V7 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 (MMYY 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、 の入力PANとカードの有効期限を使用して、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ある特定の 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 ガイドのVerifyCardValidationData」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 を検証する

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 verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999} --validation-data 801
```

```
{
    "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**
>
> PCI に準拠するには、受信と送信の PrimaryAccountNumber の値が一致する必要があります。ある PAN から別の PAN に PIN を翻訳することはできません。

トピック

PEK から DUKPT へのピン

Example

この例では、DUKPT を使用した AES ISO 4 PIN ブロックから ISO 0 PIN ブロックを使用した PEK TDES 暗号化に PIN を変換します。これは、支払いターミナルが ISO 4 でピンを暗号化し、次の接続がまだ AES をサポートしていない場合に、ダウンストリーム処理のために TDES に変換される可能性がある場合に一般的です。

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-block
 "AC17DC148BDA645E" --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": "1F4209C670E49F83E75CC72E81B787D9",
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
            "KeyCheckValue": "7CC9E2"
        }
```

PEK から PEK への PIN

Example

この例では、ある PEK (PIN 暗号化キー) で暗号化された PIN を別の PEK に変換します。これは一般的に、異なる暗号化キーを使用する異なるシステムまたはパートナー間でトランザクションをルーティングするときに使用され、プロセス全体で PIN を暗号化して PCI PIN コンプライアンスを維持します。どちらのキーもこの例では TDES 3KEY 暗号化を使用しますが、AES ISO-4 から TDES ISO-0、DUKPT から PEK、AS2805 から PEK へのさまざまなオプションを使用できます。

```
$ 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-translation-attributes
 IsoFormat0='{PrimaryAccountNumber=171234567890123}' \
    --outgoing-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh
```

```
{
    "PinBlock": "E8F2A6C4D1B93E7F",
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
    "KeyCheckValue": "9A325B"
}
```

出力 PIN ブロックが 2 番目の PEK で暗号化され、対応するキーを保持するダウンストリームシステムに安全に送信できるようになりました。

PIN データ生成

生成 PIN データ関数は、トランザクション時または認可時にユーザーがピンエントリを検証するために使用される PVV やピンブロックオフセットなど、PIN 関連の値を生成するために使用されます。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。

ランダムピンと一致する Visa PVV を生成する

Example

この例では、出力が暗号化された (PinData.PinBlock) と PIN block (pinData.Offset) である新しい PVV (ランダム) ピンを生成します。PinData.PinBlock) 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、キーのタイプが 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"
            }
        }
```

既知のピンの Visa PVV を生成する

Example

この例では、特定の (暗号化された) ピンの PVV を生成します。暗号化されたピンは、支払いターミナルから、またはユーザーが選択できるピンフローを使用してカード所有者からなど、アップストリームで受信できます。キー入力は、PAN、、Pin Verification Key、Pin Encryption Key、Encrypted Pin Blockおよび ですPIN block format。

```
$ 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
 VisaPinVerificationValue={PinVerificationKeyIndex=1,EncryptedPinBlock=AA584CED31790F37}
```

```
{
            "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。このメソッドは、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、決定論的であることは、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、発行者の処理が非常に効率的です。最も明白なのは、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に、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可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。

- 提供されたパンを 16 文字にパディングします。<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、インスタンス「A」が 9 にマッピングされ、1 が 1 にマッピングされる可能性があるため、16 進数桁を 10 進数にマッピングします。
- 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ピンまたはランダムピンを生成した場合、モジュロはカスタマー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

- 例: ピンの IBM3624 ピンオフセットを生成する

例: 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 (PinData.PinBlock) と`IBM3624`オフセット値 `PIN block` (pinData.Offset).PinData.PinBlock) 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、Ibm3624RandomPin 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方法を示しています。 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、検証のためにトランザクション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、ターミナルまたは他のアップストリームプロバイダーからの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、実行時に次の値も提供されます。入力ピンの暗号化に使用されるキー (これは多くの場合 IWK)PAN、および検証する値 (PVVまたは)PIN offset。

AWS Payment Cryptography がピンを検証できる場合、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",

        }
```

PVV メソッドを使用して暗号化された PIN を検証する - エラーの不正な PIN

Example

この例では、特定の PAN の PIN を検証しようとしますが、ピンが正しくないため失敗します。

SDKs を使用する場合、これは {"Message"Pin ブロックの検証に失敗しました。"Reason"INVALID_PIN"} と表示されます。

```
$ 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=9999}" --
encrypted-pin-block AC17DC148BDA645E
```

```
       An error occurred (VerificationFailedException) when calling the VerifyPinData
 operation: Pin block verification failed.
```

PVV メソッドを使用して暗号化された PIN を検証する - エラーの不正な入力

Example

この例では、特定の PAN の PIN を検証しようとしますが、不正な入力が原因で失敗し、受信データが有効な PIN ではありませんでした。一般的な原因は、1/wrong key being used 2/input parameters such as pan or pin block format are incorrect 3/pin block is corrupted です。

```
$ aws payment-cryptography-data verify-pin-data --verification-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2
 --encryption-key-identifier  --primary-account-number 171234567890123
 --pin-block-format ISO_FORMAT_0  --verification-attributes
 VisaPin="{PinVerificationKeyIndex=1,VerificationValue=9999}" --encrypted-pin-block
 AC17DC148BDA645E
```

```
       An error occurred (ValidationException) when calling the VerifyPinData
operation: Pin block provided is invalid. Please check your input to ensure all field
values are correct.
```

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に対して、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ピンの追加???と似ています。ピンが一致した場合、API は http 200 を返します。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、通常はトランザクション認可時に 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対応物) 上で行われます。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 - プライマリーアカウント番号フィールドで指定
- 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) などのキー派生方法 - SessionKeyDerivationAttributes で指定
- マスターキー派生モード (EMV オプション A など) - メジャーキー派生モードで指定
- トランザクションデータ - TransactionData フィールドで指定される、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 (金額、日付など)
- 発行者マスターキー - 個々のトランザクションを保護するために使用される暗号文 (AC) キーの生成に使用されるマスターキーで、KeyIdentifier フィールドで指定されます。

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:
0000000017000000000000000840008000800008401605170000000093800000B03011203 (74 文字または 37 バイト)

パディング付き:
0000000017000000000000000840008000800008401605170000000093800000B03011203 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。ARCQ (認可リクエスト暗号) 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
 0000000017000000000000000840008000800008401605170000000009380000B03011203000000 \
--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。ARCQ が検証されていない場合、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
 \
00000000000000000000000000000000000000000000800000000000000000
--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。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) などの暗号化ハッシュは、暗号化を利用して 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。このサービスは、CBC-MAC のタイプである ISO9797 アルゴリズム 1 および 3 もサポートしています。 MACs

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 MACs をサポートしています。

ISO9797 アルゴリズム 1

ISO9797_ALGORITHM1 KeyUsageの で表されます。フィールドがブロックサイズの倍数でない場合 (TDES では 8 バイト/16 進数文字、AES では 16 バイト/32 文字) 、 AWS Payment Cryptography は ISO9797 パディング方法 1 を自動的に適用します。他のパディング方法が必要な場合は、サービスを呼び出す前に適用できます。

ISO9797 アルゴリズム 3 (リテールマック)

ISO9797_ALGORITHM3 KeyUsageの で表されます。アルゴリズム 1 と同じパディングルールが適用されます

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。

AS2805.4.1 MAC

TR31_M0_ISO_16609_MAC_KEY KeyUsageの で表されます。AS2805 の詳細については、「」を参照してください。 ???

DUKPT MAC

DUKPT MAC は通常、支払いターミナルとの間で送受信されるメッセージのソースとペイロードを確認するために使用されます。DUKPT 取得手法を使用してキーを取得し、MAC を実行します。このオプションで使用されるキーは、TR31_B0_BASE_DERIVATION_KEY KeyUsageの によって示されます。

EMV MAC

EMV MAC は通常、EMV ドキュメントでは整合性キーと呼ばれます。EMV 導出手法を使用してキーを取得し、内部的に ISO9797_ALGORITHM3 を使用します。通常、発行者スクリプトをチップカードに送信して再プログラミングするために使用されます。このオプションで使用されるキーは、TR31_E2_EMV_MKEY_INTEGRITY KeyUsageの で示されます。スクリプトを送信し、オフラインピンを更新する場合は、これらのオペレーションの両方を実行する GenerateMacEmvPinChange を参照してください。

トピック

- MAC の生成
- MAC の検証

MAC の生成

Generate MAC API は、既知の暗号化キー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) を生成することで、カード磁気ストライプからのトラックデータなどのカード関連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で ISO 9797-1 アルゴリズム 1 および ISO 9797-1 アルゴリズム 3 MAC (静的 MAC キーと派生 DUKPT キーを使用) 、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 API のすべてのオプションの詳細については、GenerateMac」とVerifyMac」を参照してください。

オプションのパラメータ mac-length を使用すると、出力値を切り捨てることができます (ただし、これはコード内で実行することもできます) 。8 文字の長さは、8 バイトまたは 16 進数文字を指します。

MAC キーは、CreateKey を呼び出して AWS Payment Cryptography で作成することも、ImportKey を呼び出してインポートすることもできます。

> **Note**
>
> CMAC アルゴリズムと HMAC アルゴリズムにはパディングは必要ありません。その他はすべて、TDES の場合は 8 バイト (16 進数文字) 、AES の場合は 16 バイト (32 進数文字) の倍数であるアルゴリズムのブロックサイズにデータをパディングする必要があります。

例

- HMAC の生成
- ISO 9797-1 アルゴリズム 3 を使用して MAC を生成する
- CMAC を使用して MAC を生成する
- DUKPT CMAC を使用して MAC を生成する

HMAC の生成

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。ハッシュの長さ (256 など) は、キーの作成時に定義され、変更できません。

オプションの mac-length パラメータは出力 MAC をトリミングしますが、これは サービス外で実行することもできます。この値はバイト単位であるため、16 の値は長さ 32 の 16 進文字列を想定します。

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
```

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

ISO 9797-1 アルゴリズム 3 を使用して MAC を生成する

この例では、カードデータ認証に ISO 9797-1 アルゴリズム 3 (小売 MAC) を使用して MAC を生成します。キーは、KeyUsage が TR31_M3_ISO_9797_3_MAC_KEY に設定され、KeyModesOfUse が Generate に設定されている必要があります。

Example

```
$ aws payment-cryptography-data generate-mac \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h \
    --message-data
 "3b3130383834393030313037333934313533d32343038323236303030373030303f33" \
    --generation-attributes="Algorithm=ISO9797_ALGORITHM3"
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
    "KeyCheckValue": "2976EA",
    "Mac": "A8F7A73DAF87B6D0"
}
```

CMAC を使用して MAC を生成する

CMAC はキーが AES の場合に最も一般的に使用されますが、TDES もサポートしています。この例では、AES キーによるカードデータ認証に CMAC (ISO 9797-1 アルゴリズム 5) を使用して MAC を生成します。キーは、KeyUsage が `TR31_M6_ISO_9797_5_CMAC_KEY` に設定され、KeyModesOfUse が `Generate` に設定されている必要があります。

Example

```
$ aws payment-cryptography-data generate-mac \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi \
    --message-data
 "3b3130383834393030313037333934313533d32343038323236303030373030303f33" \
    --generation-attributes Algorithm="CMAC"
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "C1EB8F",
    "Mac": "1F8C36E63F91E4E93DF7842BF5E2E5F7"
}
```

DUKPT CMAC を使用して MAC を生成する

この例では、カードデータ認証に CMAC を使用する DUKPT (トランザクションあたりの派生一意キー) を使用して MAC を生成します。キーは KeyUsage を に設定`TR31_B0_BASE_DERIVATION_KEY`し、KeyModesOfUse を true `DeriveKey`に設定する必要があります。DUKPT キーは、基本派生キー (BDK) とキーシリアル番号 (KSN) を使用して、トランザクションごとに一意のキーを取得します。

Example

```
$ aws payment-cryptography-data generate-mac --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/qnobl5lghrzunce6 --message-data
 "3b3130383834393030313037333934313533d323430383232363030303730303033f33" --generation-
attributes="DukptCmac={KeySerialNumber="932A6E954ABB32DD00000001",Direction=BIDIRECTIONAL}"
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6",
    "KeyCheckValue": "C1EB8F"
}
```

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 で作成することも、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で DUKPT MAC、HMAC、および EMV MAC 暗号化キーをサポートします。

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。

例

- HMAC の検証
- DUKPT CMAC を使用して MAC を検証する

HMAC の検証

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 を に設定`TR31_M7_HMAC_KEY`し、KeyModesOfUse を true `Verify`に設定する必要があります。

Example

```
$ aws payment-cryptography-data verify-mac \
     --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6 \
     --message-data
 "3b343038383439303031303733393431353d323430383232363030303730303f33" \
     --mac ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C \
     --verification-attributes Algorithm=HMAC_SHA256
```

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

DUKPT CMAC を使用して MAC を検証する

この例では、カードデータ認証に DUKPT (トランザクションあたりの派生一意キー) と CMAC を使用して MAC を検証します。キーは KeyUsage を に設定TR31_B0_BASE_DERIVATION_KEYし、KeyModesOfUse を true DeriveKeyに設定する必要があります。DUKPT キーは、基本派生キー (BDK) とキーシリアル番号 (KSN) を使用して、トランザクションごとに一意のキーを取得します。DukptKeyVariant の値は、送信者と受信者の間で一致する必要があります。REQUEST は通常、ターミナルからバックエンド、バックエンドからターミナルへの VERIFY、および 1 つのキーを両方向に使用する場合の「IRECTIONAL」に使用されます。

Example

```
$ aws payment-cryptography-data verify-mac \
     --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi \
     --message-data
 "3b343038383439303031303733393431353d323430383232363030303730303033f33" \
     --mac D8E804EE74BF1D909A2C01C0BDE8EF34 \
     --verification-attributes
 DukptCmac='{"KeySerialNumber":"932A6E954ABB32DD00000001","DukptKeyVariant":"BIDIRECTIONAL"}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "C1EB8F"
}
```

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

- GenerateCardData
- VerifyCardData
- PIN データを生成 (VISA/ABA スキーム用)
- PIN データを生成 (用) IBM3624
- VerifyPinData (ビザ/ABA スキーム用)
- VerifyPinData (IBM3624 用)

- 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
- GenerateMacEmvPinChange
- 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

GenerateCardData

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
GenerateCardData	• 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 }
GenerateCardData	• CARD_VERIFICATION_VALUE_1 • CARD_VERIFICATION_VALUE_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY	{ Generate = true },{ Generate = true, Verify = true }
GenerateCardData	• CARDHOLDER_AUTHENTICATION_V	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{DeriveKey = true}

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
	ERIFICATION_VALUE			
GenerateCardData	• DYNAMIC_CARD_VERIFICATION_CODE	TR31_E4_EMV_MKEY_DYNAMIC_NUMBER	• TDES_2KEY	{DeriveKey = true}
GenerateCardData	• DYNAMIC_CARD_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{DeriveKey = true}

VerifyCardData

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	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY	{ Generate = true }, { Generate = true, Verify = true }

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
• CARD_VERIFICATION_VALUE_2			
• 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}

PIN データを生成 (VISA/ABA スキーム用)

`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 }

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 生成キー	TR31_V2_VISA_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true }

PIN データを生成 (用) **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 • { Encrypt = true, Wrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true } IBM 3624_PIN_OFFSET 用 • { Encrypt = 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
			• { 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 (ビザ/ABA スキーム用)

`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 (**IBM3624** 用)

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, 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}
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}
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	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 }
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 }

Direction	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
				• { 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いため、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 } • { 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 }

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 (HMAC)	TR31_M7_HMAC_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true }
MAC キー (AS2805)	TR31_M0_ISO_16609_MAC_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true }

GenerateMacEmvPinChange

GenerateMacEmvPinChange は、EMV オフライン PIN 変更オペレーションの MAC 生成と PIN 暗号化を組み合わせます。このオペレーションには、MAC 生成用の整合性キーと PIN 暗号化用の機密性キーの 2 つの異なるキータイプが必要です。

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
Secure Messaging Integrity キー	TR31_E2_EMV_MKEY_INTEGRITY	• TDES_2KEY	• { NoRestrictions = true }
Secure Messaging の機密性キー	TR31_E1_EMV_MKEY_CONFIDENTIALITY	• TDES_2KEY	• {DeriveKey = true}
現在の PIN PEK (PIN 暗号化キー)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192	• { 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
		• AES_256	Wrap = true, Unwrap = true } • { NoRestrictions = true }
新しい PIN PEK (PIN 暗号化キー)	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 }
ARQC キー Note Visa および Amex 取得スキームにのみ適用されます。	TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS	• TDES_2KEY	• {DeriveKey = 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 • AES_192 • AES_256	• { 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 }
信頼された 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 }
ECDH などのキーアグリーメントアルゴリズムに使用されるキー	TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT	• ECC_NIST_P256 • ECC_NIST_P384 • ECC_NIST_P521	• {DeriveKey = 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

一般的なユースケース

AWS Payment Cryptography は、多くの一般的な支払い暗号化オペレーションをサポートしています。以下のトピックは、一般的なユースケースでこれらのオペレーションを使用する方法のガイドとして機能します。すべてのコマンドのリストについては、 AWS Payment Cryptography API を確認してください。

トピック

- 発行者と発行者プロセッサ
- 調達と支払いのファシリテーター

発行者と発行者プロセッサ

発行者のユースケースは通常、いくつかの部分で構成されます。このセクションでは、 関数 (ピンの操作など) 別に整理されています。本番システムでは、キーは通常、特定のカードビンに限定され、次に示すようにインラインではなく、ビンのセットアップ中に作成されます。

トピック

- 一般関数
- ネットワーク固有の関数

一般関数

トピック

- ランダムピンと関連付けられた PVV を生成し、値を検証する
- 特定のカードの CVV を生成または検証する
- 特定のカードの CVV2 を生成または検証する
- 特定のカードの iCVV を生成または検証する
- EMV ARQC を検証し、ARPC を生成する
- EMV MAC の生成と検証
- PIN 変更用の EMV MAC の生成

ランダムピンと関連付けられた PVV を生成し、値を検証する

トピック

- キーを作成する (複数可)
- ランダムピンを生成し、PVV を生成して、暗号化された PIN と PVV を返す
- PVV メソッドを使用して暗号化された PIN を検証する

キーを作成する (複数可)

ランダムピンと PVV を生成するには、PVV を生成するためのピン検証キー (PVK) と、ピンを暗号化するためのピン暗号化キーの 2 つのキーが必要です。ピン自体はサービス内でランダムに安全に生成され、どちらのキーにも暗号的に関連しません。

PGK は、PVV アルゴリズム自体に基づくアルゴリズム TDES_2KEY のキーである必要があります。PEK は TDES_2KEY、TDES_3KEY、または AES_128 です。この場合、PEK はシステム内での使用を目的としているため、AES_128 が適しています。PEK を他のシステム (カードネットワーク、アクワイアラー、ATMs など) と交換する場合、または移行の一環として移動する場合、TDES_2KEY は互換性の理由からより適切な選択肢である可能性があります。

PEK を作成する

```
$ aws payment-cryptography create-key \
               --exportable
               --key-attributes
 KeyAlgorithm=AES_128,KeyUsage=TR31_P0_PIN_ENCRYPTION_KEY,\
                   KeyClass=SYMMETRIC_KEY,\
                   KeyModesOfUse='{Encrypt=true,Decrypt=true,Wrap=true,Unwrap=true}' --
tags='[{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
ivi5ksfsuplneuyt",
                   "KeyAttributes": {
                       "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY",
                       "KeyClass": "SYMMETRIC_KEY",
```

```
                    "KeyAlgorithm": "AES_128",
                    "KeyModesOfUse": {
                        "Encrypt": false,
                        "Decrypt": false,
                        "Wrap": false,
                        "Unwrap": false,
                        "Generate": true,
                        "Sign": false,
                        "Verify": true,
                        "DeriveKey": false,
                        "NoRestrictions": false
                    }
                },
                "KeyCheckValue": "7CC9E2",
                "KeyCheckValueAlgorithm": "CMAC",
                "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KeyArnを表す に注意してください。たとえば、arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsupltitudeyt です。これは次のステップで行います。

PVK を作成する

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_V2_VISA_PIN_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyMo
  --tags='[{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/ov6icy4ryas4zcza",
                    "KeyAttributes": {
                        "KeyUsage": "TR31_V2_VISA_PIN_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": "51A200",
                "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"
            }
        }
```


arn:aws:payment-cryptography:us-east-2:111122223333:key/ov6icy4ryas4zcza など、キー`KeyArn`を表す を書き留めます。これは次のステップで行います。

ランダムピンを生成し、PVV を生成して、暗号化された PIN と PVV を返す

Example

この例では、出力が暗号化された (PinData.PinBlock) と `PIN block` (pinData.VerificationValue) である新しい PVV (ランダム) 4 桁のピンを生成します。PinData.PinBlock) pinData.VerificationValue). キー入力は`PAN`、、 `Pin Verification Key`(ピン生成キーとも呼ばれます) 、、`Pin Encryption Key`および PIN ブロック形式です。

このコマンドでは、キーのタイプが 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"
            }
        }
```

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、検証のためにトランザクション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、ターミナルまたは他のアップストリームプロバイダーからの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、実行時に次の値も提供されます。暗号化されたピン、入力ピンの暗号化に使用されるキー (IWK と呼ばれることが多い)PAN、および検証する値 (PVVまたは)PIN offset。

AWS Payment Cryptography がピンを検証できる場合、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",
}
```

特定のカードの CVV を生成または検証する

CVV または CVV1 は、従来カードの磁気ストライプに埋め込まれていた値です。これは CVV2 とは異なります (カード所有者が表示でき、オンライン購入に使用できます)。

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、CVK の 2 倍長 3DES (2KEY TDES) キーを作成します。

> **Note**
>
> CVV、CVV2、および iCVV はすべて、同一のアルゴリズムではなくても同様のアルゴリズムを使用しますが、入力データは異なります。すべて同じキータイプ TR31_C0_CARD_VERIFICATION_KEY を使用しますが、目的ごとに個別のキーを使用することをお勧めします。これらは、以下の例のようにエイリアスやタグを使用して区別できます。

キーを作成する

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVV"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
r52o3wbqxyf6qlqr",
                "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": "DE89F9",
                "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"
            }
        }
```

arn:aws:payment-cryptography:us-east-2:111122223333:key/r52o3wbqxyf6qlqr など、キーKeyArnを表す を書き留めます。これは次のステップで行います。

CVV の生成

Example

この例では、入力が 、サービスコード (ISO/IEC 7813 で定義) が 121PAN、カードの有効期限がある特定の PAN の CVV を生成します。

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/
r52o3wbqxyf6qlqr --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=121}'
```

```
                {
                    "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/r52o3wbqxyf6qlqr",
                    "KeyCheckValue": "DE89F9",
                    "ValidationData": "801"
                }
```

CVV を検証する

Example

この例では、CVK、、、121 のサービスコード、カードの有効期限、検証するトランザクション中に提供された CVV の入力を使用して、特定の PAN の CVV を検証します。 PAN

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。

> **Note**
>
> CVV はユーザーが入力した値 (CVV2 など) ではありませんが、通常はマグストライプに埋め込まれます。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。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/r52o3wbqxyf6qlqr
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=121} --validation-data 801
```

```
{
                    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
r52o3wbqxyf6qlqr",
                    "KeyCheckValue": "DE89F9",
                    "ValidationData": "801"
}
```

特定のカードの CVV2 を生成または検証する

CVV2 は、従来カードの裏側で提供され、オンライン購入に使用される値です。仮想カードの場合、アプリや画面に表示されることもあります。暗号的には、CVV1 と同じですが、サービスコード値が異なります。

キーを作成する

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVV2"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/7f7g4spf3xcklhzu",
                   "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": "AEA5CD",
                   "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"
            }
        }
```

arn:aws:payment-cryptography:us-east-2:111122223333:key/7f7g4spf3xcklhzu など、キーKeyArnを表す を書き留めます。これは次のステップで行います。

CVV2 の生成

Example

この例では、 の入力PANとカードの有効期限を使用して、特定の PAN の CVV2 を生成します。

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 generate-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/7f7g4spf3xcklhzu
 --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue2='{CardExpiryDate=1127}'
```

```
                {
                    "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu",
                    "KeyCheckValue": "AEA5CD",
                    "ValidationData": "321"
                }
```

CVV2 を検証する

Example

この例では、CVK の入力を含む特定の PAN の CVV2 PANと、検証するトランザクション中に提供されたカードの有効期限と CVV を検証します。

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。

> **Note**
>
> CVV2 およびその他の入力は、ユーザーが入力した値です。そのため、これが定期的に検証に失敗する問題の兆候であるとは限りません。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/7f7g4spf3xcklhzu
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue2='{CardExpiryDate=1127} --validation-data 321
```

```
{
                        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu",
                        "KeyCheckValue": "AEA5CD",
                        "ValidationData": "801"
   }
```

特定のカードの iCVV を生成または検証する

iCVV は CVV/CVV2 と同じアルゴリズムを使用しますが、iCVV はチップカード内に埋め込まれます。サービスコードは 999 です。

キーを作成する

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"ICVV"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
c7dsi763r6s7lfp3",
                   "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": "1201FB",
                "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KeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/c7dsi763r6s7lfp3 です。これは次のステップで行います。

iCVV の生成

Example

この例では、入力が 、サービスコード (ISO/IEC 7813 で定義) が 999PAN、カードの有効期限がある特定の PAN の iCVV を生成します。

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/
c7dsi763r6s7lfp3 --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999}'
```

```
                    {
                        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/c7dsi763r6s7lfp3",
                        "KeyCheckValue": "1201FB",
                        "ValidationData": "532"
                    }
```

iCVV を検証する

Example

検証の場合、入力は CVK、PAN、999 のサービスコード、カードの有効期限、検証するトランザクション中に提供された iCVV です。

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。

> **Note**
>
> iCVV はユーザーが入力した値 (CVV2 など) ではありませんが、通常は EMV/チップカードに埋め込まれます。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。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/c7dsi763r6s7lfp3
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999} --validation-data 532
```

```
{
                    "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/c7dsi763r6s7lfp3",
                    "KeyCheckValue": "1201FB",
                    "ValidationData": "532"
   }
```

EMV ARQC を検証し、ARPC を生成する

ARQC (認可リクエスト暗号) は、EMV (チップ) カードによって生成され、トランザクションの詳細と認可されたカードの使用を検証するために使用される暗号文です。カード、ターミナル、トランザクション自体からのデータが組み込まれています。

バックエンドの検証時に、同じ入力が AWS Payment Cryptography に提供され、暗号文が内部で再作成され、トランザクションで指定された値と比較されます。この意味では、MAC に似ています。EMV 4.4 Book 2 は、この関数の 3 つの側面を定義します。1 回限りのトランザクションキーを生成するキー取得メソッド (共通セッションキー - CSK)、最小ペイロード、およびレスポンス (ARPC) を生成するメソッドです。

個々のカードスキームでは、組み込む追加のトランザクションフィールドを指定するか、それらのフィールドが表示される順序を指定できます。その他の (一般的に廃止された) スキーム固有の派生スキームも存在し、このドキュメントの他の箇所で説明されています。

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

キーを作成する

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN18"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
pw3s6nl62t5ushfk",
                    "KeyAttributes": {
                        "KeyUsage": "TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS",
                        "KeyClass": "SYMMETRIC_KEY",
                        "KeyAlgorithm": "TDES_2KEY",
                        "KeyModesOfUse": {
                            "Encrypt": false,
                            "Decrypt": false,
                            "Wrap": false,
                            "Unwrap": false,
                            "Generate": false,
                            "Sign": false,
```

```
                        "Verify": false,
                        "DeriveKey": true,
                        "NoRestrictions": false
                    }
                },
                "KeyCheckValue": "08D7B4",
                "KeyCheckValueAlgorithm": "ANSI_X9_24",
                "Enabled": true,
                "Exportable": true,
                "KeyState": "CREATE_COMPLETE",
                "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
                "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
            }
        }
```

arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk など、キーKeyArnを表す を書き留めます。これは次のステップで行います。

ARQC の生成

ARQC は EMV カードによってのみ生成されます。そのため、 AWS Payment Cryptography にはそのようなペイロードを生成する機能はありません。テスト目的では、適切なペイロードと、さまざまなスキームによって一般的に提供される既知の値を生成できる多数のライブラリがオンラインで利用できます。

ARQC を検証する

Example

AWS Payment Cryptography が ARQC を検証できる場合、http/200 が返されます。ARPC (レスポンス) はオプションで提供でき、ARQC の検証後にレスポンスに含めることができます。

```
$ 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
 00000000170000000000000008400080008000084016051700000000093800000B1F220103000000000000000000
 --session-key-derivation-attributes='{"EmvCommon":
{"ApplicationTransactionCounter":"000B",
 "PanSequenceNumber":"01","PrimaryAccountNumber":"9137631040001422"}}' --auth-response-
attributes='{"ArpcMethod2":{"CardStatusUpdate":"12345678"}}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
    "KeyCheckValue": "08D7B4",
    "AuthResponseValue":"2263AC85"
}
```

EMV MAC の生成と検証

EMV MAC は、EMV 派生キーの入力を使用して MAC になり、結果のデータに対して ISO9797-3 (小売) MAC を実行します。EMV MAC は通常、ブロック解除スクリプトなどの EMV カードにコマンドを送信するために使用されます。

> **Note**
>
> AWS Payment Cryptography はスクリプトの内容を検証しません。含める特定のコマンドの詳細については、スキームまたはカードマニュアルを参照してください。

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

トピック

- キーを作成する
- EMV MAC の生成

キーを作成する

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E2_EMV_MKEY_INTEGRITY,KeyClass=SYMMETRIC_KEY,KeyModesOfUs
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN18"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
pw3s6nl62t5ushfk",
            "KeyAttributes": {
```

```
                "KeyUsage": "TR31_E2_EMV_MKEY_INTEGRITY",
                "KeyClass": "SYMMETRIC_KEY",
                "KeyAlgorithm": "TDES_2KEY",
                "KeyModesOfUse": {
                    "Encrypt": false,
                    "Decrypt": false,
                    "Wrap": false,
                    "Unwrap": false,
                    "Generate": false,
                    "Sign": false,
                    "Verify": false,
                    "DeriveKey": true,
                    "NoRestrictions": false
                }
            },
            "KeyCheckValue": "08D7B4",
            "KeyCheckValueAlgorithm": "ANSI_X9_24",
            "Enabled": true,
            "Exportable": true,
            "KeyState": "CREATE_COMPLETE",
            "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
            "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
            "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
        }
    }
```

キーKeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk です。これは次のステップで行います。

EMV MAC の生成

一般的なフローは、バックエンドプロセスが EMV スクリプトを生成し (カードのブロック解除など)、このコマンドを使用して署名し (1 つの特定のカードに固有の 1 回限りのキーを取得)、MAC を返すことです。次に、コマンド + MAC がカードに送信されて適用されます。コマンドをカードに送信することは AWS Payment Cryptography の範囲外です。

> **Note**
>
> このコマンドは、暗号化されたデータ (PIN など) が送信されない場合のコマンドを対象としています。EMV Encrypt をこのコマンドと組み合わせて、このコマンドを呼び出す前に暗号化されたデータを発行者スクリプトに追加できます。

メッセージデータ

メッセージデータには、APDU ヘッダーとコマンドが含まれます。これは実装によって異なる場合がありますが、この例はブロック解除 (84 24 00 00 08) の APDU ヘッダーで、その後に ATC (0007) が続き、次に前のトランザクションの ARQC (999E57FD0F47CACE) が続きます。サービスはこのフィールドの内容を検証しません。

セッションキー取得モード

このフィールドは、セッションキーの生成方法を定義します。EMV_COMMON_SESSION_KEY は新しい実装に一般的に使用されますが、EMV2000 | AMEX | MASTERCARD_SESSION_KEY | VISA も使用できます。

MajorKeyDerivationMode

EMV はモード A、B、または C を定義します。モード A が最も一般的であり、 AWS Payment Cryptography は現在モード A またはモード B をサポートしています。

PAN

アカウント番号。通常はチップフィールド 5A または ISO8583 フィールド 2 で使用できますが、カードシステムから取得することもできます。

PSN

カードシーケンス番号。使用しない場合は、00 と入力します。

SessionKeyDerivationValue

これはセッションごとの取得データです。取得スキームに応じて、フィールド 9F26 の最後の ARQC (ApplicationCryptogram) または 9F36 の最後の ATC のいずれかになります。

[Padding] (パディング)

パディングは自動的に適用され、ISO/IEC 9797-1 パディング方法 2 を使用します。

Example

```
$ aws payment-cryptography-data generate-mac --message-data
 842400000800079 99E57FD0F47CACE --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk --message-
data 8424000008999E57FD0F47CACE0007 --generation-attributes
 EmvMac="{MajorKeyDerivationMode=EMV_OPTION_A,PanSequenceNumber='00',PrimaryAccountNumber='2235
```

```
{
```

```
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk",
"KeyCheckValue": "08D7B4",
"Mac":"5652EEDF83EA0D84"
}
```

PIN 変更用の EMV MAC の生成

EMV PIN の変更には、発行者スクリプトの MAC の生成と、EMV チップカードでのオフライン PIN 変更の新しい PIN の暗号化という 2 つのオペレーションが組み合わされます。このコマンドは、ピンがチップカードに保存されている特定の国でのみ必要です (これは欧州の国で一般的です)。これは一般的に、カード所有者が PIN を変更する必要があり、新しい PIN を MAC と一緒にカードに安全に送信してコマンドの信頼性を確認する必要がある場合に使用されます。

> **Note**
>
> カードにコマンドを送信するだけで PIN を変更しない場合は、ARPC CSU を使用するか、代わりに EMV MAC コマンドを生成することを検討してください。

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

PIN 変更のために EMV MAC と暗号化された PIN を生成する

このオペレーションには、MAC 生成用の EMV 整合性キー (KeyUsage: TR31_E2_EMV_MKEY_INTEGRITY) と PIN 暗号化用の EMV 機密性キー (KeyUsage: TR31_E4_EMV_MKEY_CONFIDENTIALITY) の 2 つのキーが必要です。一般的なフローは、バックエンドプロセスが発行者スクリプトの MAC と暗号化された新しい PIN の両方を含む EMV PIN 変更スクリプトを生成することです。その後、コマンドと暗号化された PIN がカードに送信され、オフライン PIN が更新されます。コマンドをカードに送信することは AWS Payment Cryptography の範囲外です。

メッセージデータ

メッセージデータには、発行者スクリプトの APDU コマンドが含まれます。サービスはこのフィールドの内容を検証しません。

新しい暗号化された PIN ブロック

カードに送信される新しい暗号化された PIN ブロック。これは、PIN 暗号化キーを使用して暗号化された値として提供する必要があります。

新しい PIN PEK 識別子

この API に渡される前に新しい PIN を暗号化するために使用されるキー。

Secure Messaging Integrity キー

MAC 生成に使用される EMV 整合性キー (KeyUsage: TR31_E2_EMV_MKEY_INTEGRITY)。

Secure Messaging の機密性キー

PIN 暗号化に使用される EMV 機密性キー (KeyUsage: TR31_E4_EMV_MKEY_CONFIDENTIALITY)。

MajorKeyDerivationMode

EMV はモード A、B、または C を定義します。モード A が最も一般的であり、 AWS Payment Cryptography は現在モード A またはモード B をサポートしています。

モード

暗号化モード、通常は PIN 変更オペレーション用の CBC。

PAN

アカウント番号。通常はチップフィールド 5A または ISO8583 フィールド 2 で使用できますが、カードシステムから取得することもできます。

PanSequenceNumber

カードシーケンス番号。使用しない場合は、00 と入力します。

ApplicationCryptogram

これはセッションごとの取得データで、通常はフィールド 9F26 からの最後の ARQC です。

PinBlockLengthPosition

PIN ブロックの長さがエンコードされる場所を指定します。通常、NONE に設定されます。不明な場合は、カードスキームの仕様を確認してください。

PinBlockPaddingType

PIN ブロックのパディングタイプを指定します。通常、NO_PADDING に設定されます。不明な場合は、カードスキームの仕様を確認してください。

Example

```
$ aws payment-cryptography-data generate-mac-emv-pin-change \
```

```
    --message-data 00A4040008A000000004101080D80500000001010A04000000000000 \
    --new-encrypted-pin-block 67FB27C75580EFE7 \
    --new-pin-pek-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt \
    --pin-block-format ISO_FORMAT_0 \
    --secure-messaging-confidentiality-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/tqv5yij6wtxx64pi \
    --secure-messaging-integrity-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/pw3s6nl62t5ushfk \
    --derivation-method-attributes
 'EmvCommon={ApplicationCryptogram=1234567890123457,MajorKeyDerivationMode=EMV_OPTION_A,Mode=CB
```

```
{
    "SecureMessagingIntegrityKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/pw3s6nl62t5ushfk",
    "SecureMessagingIntegrityKeyCheckValue": "08D7B4",
    "SecureMessagingConfidentialityKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/tqv5yij6wtxx64pi",
    "SecureMessagingConfidentialityKeyCheckValue": "C1EB8F",
    "Mac": "5652EEDF83EA0D84",
    "EncryptedPinBlock": "F1A2B3C4D5E6F7A8"
}
```

ネットワーク固有の関数

トピック

- Visa 固有の関数
- Mastercard 固有の関数
- American Express 固有の関数
- JCB 固有の関数

Visa 固有の関数

トピック

- ARQC - CVN18/CVN22
- ARQC - CVN10
- 3DS CAVV V7
- dCVV (動的カード検証値) - CVN17

ARQC - CVN18/CVN22

CVN18 および CVN22 は、キー取得の CSK メソッドを使用します。正確なトランザクションデータは、これら 2 つの方法によって異なります。トランザクションデータフィールドの構築の詳細については、スキームドキュメントを参照してください。

ARQC - CVN10

CVN10 は、EMV トランザクションの古い Visa メソッドであり、セッション (トランザクションごと) 取得ではなく、カードキー取得ごとに を使用し、別のペイロードも使用します。ペイロードの内容の詳細については、スキームにお問い合わせください。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN10"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
pw3s6nl62t5ushfk",
                "KeyAttributes": {
                    "KeyUsage": "TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS",
                    "KeyClass": "SYMMETRIC_KEY",
                    "KeyAlgorithm": "TDES_2KEY",
                    "KeyModesOfUse": {
                        "Encrypt": false,
                        "Decrypt": false,
                        "Wrap": false,
                        "Unwrap": false,
                        "Generate": false,
                        "Sign": false,
                        "Verify": false,
                        "DeriveKey": true,
                        "NoRestrictions": false
                    }
                },
                "KeyCheckValue": "08D7B4",
                "KeyCheckValueAlgorithm": "ANSI_X9_24",
```

```
                "Enabled": true,
                "Exportable": true,
                "KeyState": "CREATE_COMPLETE",
                "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
                "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
            }
        }
```

キーKeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk です。これは次のステップで行います。

ARQC を検証する

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
 0000000017000000000000000840008000800008401605170000000009380000B03011203000000 \
  --session-key-derivation-attributes='{"Visa":{"PanSequenceNumber":"01" \
  ,"PrimaryAccountNumber":"9137631040001422"}}'
```

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

3DS CAVV V7

Visa Secure (3DS) トランザクションの場合、発行者アクセスコントロールサーバー (ACS) によって CAVV (カード所有者認証検証値) が生成されます。CAVV は、カード所有者認証が行われた証拠であり、認証トランザクションごとに一意であり、アクワイアラーによって認可メッセージで提供されま

す。CAVV v7 は、マーチャント名、購入金額、購入日などの要素を含む、トランザクションに関する追加データを承認にバインドします。このようにして、実質的にトランザクションペイロードの暗号化ハッシュになります。

暗号的には、CAVV V7 は CVV アルゴリズムを使用しますが、入力はすべて変更/再利用されています。CAVV V7 ペイロードを生成する入力を生成する方法については、適切なサードパーティー/Visa のドキュメントを参照してください。

キーを作成する

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CAVV-V7"},
{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
dnaeyrjgdjjtw6dk",
                "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": "F3FB13",
                "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KeyArnを表す を書き留めます。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/dnaeyrjgdjjtw6dk です。これは次のステップで行います。

CAVV V7 を生成する

Example

この例では、仕様で指定された入力を持つ特定のトランザクションの CAVV V7 を生成します。このアルゴリズムでは、フィールドが再利用/再利用される可能性があるため、フィールドラベルが入力と一致すると想定しないでください。

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/
dnaeyrjgdjjtw6dk --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=9431,ServiceCode=431}'
```

```
            {
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
dnaeyrjgdjjtw6dk",
                "KeyCheckValue": "F3FB13",
                "ValidationData": "491"
            }
```

CAVV V7 を検証する

Example

検証の場合、入力は CVK、計算された入力値、検証するトランザクション中に提供された CAVV です。

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。

> **Note**
>
> CAVV はユーザーが入力した値 (CVV2 など) ではありませんが、発行者 ACS によって計算されます。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。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/dnaeyrjgdjjtw6dk
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue1='{CardExpiryDate=9431,ServiceCode=431} --validation-data 491
```

```
{
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
dnaeyrjgdjjtw6dk",
                "KeyCheckValue": "F3FB13",
                "ValidationData": "491"
}
```

dCVV (動的カード検証値) - CVN17

dCVV (動的カード検証値) は、非接触 EMV トランザクションに使用される Visa 固有の動的暗号文です。初期 EMV と呼ばれ、トランザクションごとに一意の検証値を生成することでセキュリティを強化します。dCVV は、プライマリアカウント番号 (PAN)、PAN シーケンス番号 (PSN)、アプリケーショントランザクションカウンター (ATC)、予測不可能な番号、トラックデータなどの入力を使用します。まだいくつかの場所で使用されていますが、主に CVN18 などの他のアルゴリズムに置き換えられています。

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DynamicCardVerificationValue」を参照してください。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"DCVV"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
mw7dn3qxvkfh8ztc",
                "KeyAttributes": {
                    "KeyUsage": "TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS",
                    "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": "A8E4D2",
                "KeyCheckValueAlgorithm": "ANSI_X9_24",
                "Enabled": true,
                "Exportable": true,
                "KeyState": "CREATE_COMPLETE",
                "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                "CreateTimestamp": "2025-02-02T11:45:30.648000-08:00",
                "UsageStartTimestamp": "2025-02-02T11:45:30.626000-08:00"
            }
        }
```

キーKeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/mw7dn3qxvkfh8ztc です。これは次のステップで行います。

dCVV の生成

Example

この例では、非接触 EMV トランザクションの dCVV を生成します。入力には、PAN、PAN シーケンス番号、アプリケーショントランザクションカウンター、予測不可能な番号、追跡データが含まれます。

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/mw7dn3qxvkfh8ztc \
    --primary-account-number=5111112627662122 \
    --generation-attributes
 DynamicCardVerificationValue='{ApplicationTransactionCounter=01,PanSequenceNumber=00,TrackData
 \
    --validation-data-length 5
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
mw7dn3qxvkfh8ztc",
    "KeyCheckValue": "A8E4D2",
    "ValidationData": "36667"
}
```

dCVV を検証する

Example

この例では、トランザクション中に提供された dCVV を検証します。検証のためには、生成に使用されるのと同じ入力を指定する必要があります。

AWS Payment Cryptography が検証できる場合、http/200 が返されます。値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-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/mw7dn3qxvkfh8ztc \
    --primary-account-number=5111112627662122 \
    --validation-data=36667 \
    --verification-attributes
 DynamicCardVerificationValue='{ApplicationTransactionCounter=01,PanSequenceNumber=00,TrackData
```

```
{
```

```
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
mw7dn3qxvkfh8ztc",
    "KeyCheckValue": "A8E4D2"
}
```

Mastercard 固有の関数

トピック

- DCVC3
- ARQC - CVN14/CVN15
- ARQC - CVN12/CVN13
- 3DS SPA2 "

DCVC3

DCVC3 は EMV CSK および Mastercard CVN12 スキームより前であり、動的キーを利用する別のアプローチを表します。他のユースケースにも転用されることがあります。このスキームでは、入力は PAN、PSN、Track1/Track2 データ、予測不可能な番号、トランザクションカウンター (ATC) です。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"DCVC3"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
hrh6qgbi3sk4y3wq",
                "KeyAttributes": {
                    "KeyUsage": "TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS",
                    "KeyClass": "SYMMETRIC_KEY",
                    "KeyAlgorithm": "TDES_2KEY",
                    "KeyModesOfUse": {
                        "Encrypt": false,
```

```
                    "Decrypt": false,
                    "Wrap": false,
                    "Unwrap": false,
                    "Generate": false,
                    "Sign": false,
                    "Verify": false,
                    "DeriveKey": true,
                    "NoRestrictions": false
                }
            },
            "KeyCheckValue": "08D7B4",
            "KeyCheckValueAlgorithm": "ANSI_X9_24",
            "Enabled": true,
            "Exportable": true,
            "KeyState": "CREATE_COMPLETE",
            "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
            "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
            "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
        }
    }
```

キーKeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/hrh6qgbi3sk4y3wq です。これは次のステップで行います。

DCVC3 を生成する

Example

DCVC3 は通常チップカードによって生成されますが、この例のように手動で生成することもできます。

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk
 --primary-account-number=5413123456784808 --generation-attributes
 DynamicCardVerificationCode='{ApplicationTransactionCounter=0000,TrackData=524106000000069D13
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
    "KeyCheckValue": "08D7B4",
    "ValidationData": "865"
  }
```

DCVC3 を検証する

Example

この例では、DCVC3 を検証します。例えば、11 のカウンターは 000B として表す必要があります。サービスは 3 桁の DCVC3 を想定しているため、4 (または 5) 桁の値を保存している場合は、3 桁になるまで左側の文字を切り捨てます (たとえば、15321 では検証データ値が 321 になります) 。

AWS Payment Cryptography が検証できる場合、http/200 が返されます。値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-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk
 --primary-account-number=5413123456784808 --verification-attributes
 DynamicCardVerificationCode='{ApplicationTransactionCounter=000B,TrackData=524106000000069D13
 --validation-data 398
```

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

ARQC - CVN14/CVN15

CVN14 および CVN15 は、キー取得の EMV CSK メソッドを使用します。正確なトランザクションデータは、これら 2 つの方法によって異なります。トランザクションデータフィールドの構築の詳細については、スキームドキュメントを参照してください。

ARQC - CVN12/CVN13

CVN12 と CVN13 は、EMV トランザクション用の古い Mastercard 固有の方法で、トランザクションごとの取得に予測不可能な数値が組み込まれ、別のペイロードも使用されます。ペイロードの内容の詳細については、 スキームにお問い合わせください。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN12"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
pw3s6nl62t5ushfk",
                "KeyAttributes": {
                    "KeyUsage": "TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS",
                    "KeyClass": "SYMMETRIC_KEY",
                    "KeyAlgorithm": "TDES_2KEY",
                    "KeyModesOfUse": {
                        "Encrypt": false,
                        "Decrypt": false,
                        "Wrap": false,
                        "Unwrap": false,
                        "Generate": false,
                        "Sign": false,
                        "Verify": false,
                        "DeriveKey": true,
                        "NoRestrictions": false
                    }
                },
                "KeyCheckValue": "08D7B4",
                "KeyCheckValueAlgorithm": "ANSI_X9_24",
                "Enabled": true,
                "Exportable": true,
                "KeyState": "CREATE_COMPLETE",
                "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
                "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
            }
        }
```

arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk など、キーKeyArnを表す を書き留めます。これは次のステップで行います。

ARQC を検証する

Example

この例では、Mastercard CVN12 を使用して生成された ARQC を検証します。

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 31BE5D49F14A5F01 \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
  --major-key-derivation-mode EMV_OPTION_A \
  --transaction-data
 0000000017000000000000000084000800080000840160517000000000093800000B1F220103000000000000000000000
 \
  --session-key-derivation-attributes='{"MastercardSessionKey":
{"ApplicationTransactionCounter":"000B","PanSequenceNumber":"01","PrimaryAccountNumber":"541312
```

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

3DS SPA2 "

SPA2 " (アカウント認証値) は、Mastercard 3DS トランザクション (Mastercard Idenity Check とも呼ばれます) に使用されます。HMAC ベースの MAC 生成を使用した e コマーストランザクションの暗号化認証を提供します。HIPAA は、トランザクション固有のデータと共有シークレットキーを使用して生成されます。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=HMAC_SHA256,KeyUsage=TR31_M7_HMAC_KEY,KeyClass=SYMMETRIC_KEY,KeyModesOfUse='{Gene
 --tags='[{"Key":"KEY_PURPOSE","Value":"SPA2_AAV"},
{"Key":"CARD_BIN","Value":"12345678"}]'
```

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-west-2:111122223333:key/
q5vjtshsg67cz5gn",
```

```
            "KeyAttributes": {
                "KeyUsage": "TR31_M7_HMAC_KEY",
                "KeyClass": "SYMMETRIC_KEY",
                "KeyAlgorithm": "HMAC_SHA256",
                "KeyModesOfUse": {
                    "Encrypt": false,
                    "Decrypt": false,
                    "Wrap": false,
                    "Unwrap": false,
                    "Generate": true,
                    "Sign": false,
                    "Verify": true,
                    "DeriveKey": false,
                    "NoRestrictions": false
                }
            },
            "KeyCheckValue": "C661F9",
            "KeyCheckValueAlgorithm": "HMAC",
            "Enabled": true,
            "Exportable": true,
            "KeyState": "CREATE_COMPLETE",
            "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
            "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
            "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
        }
    }
```

キーKeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-west-2:111122223333:key/q5vjtshsg67cz5gn です。これは次のステップで行います。

SPA2 " を生成する

Example

この例では、HMAC MAC 生成を使用して SPA2 の発行者認証値 (IAV) コンポーネントを生成します。メッセージデータには、認証されるトランザクション固有の情報が含まれます。メッセージデータの形式は Mastercard の SPA2 仕様に従う必要があり、この例では説明していません。

> **Note**
>
> IAV を PVC 値に挿入するフォーマットについては、Mastercard の仕様を確認してください。

```
$ aws payment-cryptography-data generate-mac --key-identifier arn:aws:payment-
cryptography:us-west-2:111122223333:key/q5vjtshsg67cz5gn --message-data
 "2226400099919520FFFFd8b448be65694fe7b42f836bad396e9d" --generation-attributes
 Algorithm=HMAC --region us-west-2
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:111122223333:key/
q5vjtshsg67cz5gn",
    "KeyCheckValue": "C661F9",
    "Mac": "6FB2405E9D8A4C1F7B173F73ADD1A6DC358531CAB0E9994FC5B62012ADDE91FC"
}
```

SPA2 " を検証する

Example

この例では、SPA2 " を検証します。検証には、同じメッセージデータと MAC 値が提供されます。

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

```
$ aws payment-cryptography-data verify-mac --key-identifier arn:aws:payment-
cryptography:us-west-2:111122223333:key/q5vjtshsg67cz5gn --message-
data "2226400099919520FFFFd8b448be65694fe7b42f836bad396e9d" --mac
 "6FB2405E9D8A4C1F7B173F73ADD1A6DC358531CAB0E9994FC5B62012ADDE91FC" --verification-
attributes Algorithm=HMAC --region us-west-2
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-west-2:111122223333:key/
q5vjtshsg67cz5gn",
    "KeyCheckValue": "C661F9"
}
```

American Express 固有の関数

トピック

- CSC1
- CSC2
- iCSC

- 3DS AEVV

CSC1

CSC バージョン 1 は Classic CSC アルゴリズムとも呼ばれます。サービスは、3、4、または 5 桁の数字として提供できます。

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AmexCardSecurityCodeVersion1」を参照してください。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CSC1"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
esh6hn7pxdtttzgq",
                    "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": "8B5077",
                    "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KeyArnを表す を書き留めます。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/esh6hn7pxdtttzgq です。これは次のステップで行います。

CSC1 を生成する

Example

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
esh6hn7pxdtttzgq --primary-account-number=344131234567848 --generation-attributes
 AmexCardSecurityCodeVersion1='{CardExpiryDate=1224}' --validation-data-length 4
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
esh6hn7pxdtttzgq",
    "KeyCheckValue": "8B5077",
    "ValidationData": "3938"
  }
```

CSC1 を検証する

Example

この例では、CSC1 を検証します。

AWS Payment Cryptography が検証できる場合、http/200 が返されます。値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-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/esh6hn7pxdtttzgq
 --primary-account-number=344131234567848 --verification-attributes
 AmexCardSecurityCodeVersion1='{CardExpiryDate=1224}''  --validation-data 3938
```

```
{
```

```
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
esh6hn7pxdtttzgq",
    "KeyCheckValue": "8B5077"
  }
```

CSC2

CSC バージョン 2 は、拡張 CSC アルゴリズムとも呼ばれます。サービスは、3、4、または 5 桁の数字として提供できます。CSC2 のサービスコードは通常 000 です。

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AmexCardSecurityCodeVersion2」を参照してください。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CSC2"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
erlm445qvunmvoda",
                "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": "BF1077",
```

```
                "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"
            }
        }
```

arn:aws:payment-cryptography:us-east-2:111122223333:key/erlm445qvunmvoda など、キーKeyArnを表す を書き留めます。これは次のステップで行います。

CSC2 の生成

この例では、長さが 4 の CSC2 を生成します。CSC は、3、4、または 5 の長さで生成できます。American Express PANs は 15 桁で、34 または 37 で始まる必要があります。有効期限は通常 YYMM の形式です。サービスコードは異なる場合があります - 手動を確認してくださいが、一般的な値は 000、201、または 702 です

Example

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
erlm445qvunmvoda --primary-account-number=344131234567848 --generation-attributes
 AmexCardSecurityCodeVersion2='{CardExpiryDate=2412,ServiceCode=000}' --validation-
data-length 4
```

```
{
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/erlm445qvunmvoda",
"KeyCheckValue": "BF1077",
"ValidationData": "3982"
}
```

CSC2 を検証する

Example

この例では、CSC2 を検証します。

AWS Payment Cryptography が検証できる場合、http/200 が返されます。値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-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/erlm445qvunmvoda
 --primary-account-number=344131234567848 --verification-attributes
 AmexCardSecurityCodeVersion2='{CardExpiryDate=2412,ServiceCode=000}' --validation-data
 3982
```

```
{
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/erlm445qvunmvoda",
"KeyCheckValue": "BF1077"
}
```

iCSC

iCSC は静的 CSC アルゴリズムとも呼ばれ、CSC バージョン 2 を使用して計算されます。サービスは、3、4、または 5 桁の数字として提供できます。

サービスコード 999 を使用して、問い合わせカードの iCSC を計算します。サービスコード 702 を使用して、非接触カードの iCSC を計算します。

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AmexCardSecurityCodeVersion2」を参照してください。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CSC1"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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-1:111122223333:key/7vrybrbvjcvwtunv",
        "KeyAttributes": {
            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
            "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
            },
        },
        "KeyCheckValue": "7121C7",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": true,
        "Exportable": true,
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "CreateTimestamp": "2025-01-29T09:19:21.209000-05:00",
        "UsageStartTimestamp": "2025-01-29T09:19:21.192000-05:00"
        }
    }
```

キーKeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-east-1:111122223333:key/7vrybrbvjcvwtunv です。これは次のステップで行います。

iCSC の生成

この例では、サービスコード 702 を使用して非接触カード用に、長さ 4 の iCSC を生成します。CSC は、3、4、または 5 の長さで生成できます。American Express PANs は 15 桁で、34 または 37 で始まる必要があります。

Example

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-1:111122223333:key/7vrybrbvjcvwtunv
 --primary-account-number=344131234567848 --generation-attributes
 AmexCardSecurityCodeVersion2='{CardExpiryDate=1224,ServiceCode=702}' --validation-
data-length 4
```

```
{
    "KeyArn": arn:aws:payment-cryptography:us-east-1:111122223333:key/7vrybrbvjcvwtunv,
    "KeyCheckValue": 7121C7,
```

```
    "ValidationData": "2365"
}
```

iCSC を検証する

Example

この例では、iCSC を検証します。

AWS Payment Cryptography が検証できる場合、http/200 が返されます。値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-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-1:111122223333:key/7vrybrbvjcvwtunv
 --primary-account-number=344131234567848 --verification-attributes
 AmexCardSecurityCodeVersion2='{CardExpiryDate=1224,ServiceCode=702}' --validation-data
 2365
```

```
{
    "KeyArn": arn:aws:payment-cryptography:us-east-1:111122223333:key/7vrybrbvjcvwtunv,
    "KeyCheckValue": 7121C7
}
```

3DS AEVV

3DS AEVV (3-D Secure Account Verification Value) は、American Express 3-D Secure 認証に使用されます。CSC2 と同じアルゴリズムを使用しますが、入力パラメータは異なります。有効期限フィールドには、予測不可能な (ランダム) 番号を入力する必要があります。サービスコードは、AEVV 認証結果コード (1 桁) と第 2 要素認証コード (2 桁) で構成されます。出力の長さは 3 桁に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AmexCardSecurityCodeVersion2」を参照してください。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"3DS_AEVV"},
{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/
kw8djn5qxvfh3ztm",
        "KeyAttributes": {
            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
            "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
            },
        },
        "KeyCheckValue": "8F3A21",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": true,
        "Exportable": true,
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "CreateTimestamp": "2025-02-02T10:30:15.209000-05:00",
        "UsageStartTimestamp": "2025-02-02T10:30:15.192000-05:00"
        }
    }
```

キーKeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/kw8djn5qxvfh3ztm です。これは次のステップで行います。

3DS AEVV の生成

この例では、長さが 3 の 3DS AEVV を生成します。有効期限フィールドには、予測不可能な (ランダムな) 番号 (1234 など) が含まれ、サービスコードは AEVV 認証結果コード (1 桁) と 2 番目の要素

認証コード (2 桁) で構成されます。例えば、543 は認証結果コード、43 は 2 番目の要素認証コードです。American Express PANs は 15 桁で、34 または 37 で始まる必要があります。

Example

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
kw8djn5qxvfh3ztm --primary-account-number=344131234567848 --generation-attributes
 AmexCardSecurityCodeVersion2='{CardExpiryDate=1234,ServiceCode=543}' --validation-
data-length 3
```

```
{
    "KeyArn": arn:aws:payment-cryptography:us-east-2:111122223333:key/kw8djn5qxvfh3ztm,
    "KeyCheckValue": 8F3A21,
    "ValidationData": "921"
}
```

3DS AEVV を検証する

Example

この例では、3DS AEVV を検証します。

AWS Payment Cryptography が検証できる場合、http/200 が返されます。値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-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/kw8djn5qxvfh3ztm
 --primary-account-number=344131234567848 --verification-attributes
 AmexCardSecurityCodeVersion2='{CardExpiryDate=1234,ServiceCode=543}' --validation-data
 921
```

```
{
    "KeyArn": arn:aws:payment-cryptography:us-east-2:111122223333:key/kw8djn5qxvfh3ztm,
    "KeyCheckValue": 8F3A21
}
```

JCB 固有の関数

トピック

- ARQC - CVN04
- ARQC - CVN01

ARQC - CVN04

JCB CVN04 は、キー取得の CSK メソッドを使用します。トランザクションデータフィールドの構築の詳細については、スキームのドキュメントを参照してください。

ARQC - CVN01

CVN01 は EMV トランザクション用の古い JCB メソッドで、セッション (トランザクションごと) 取得ではなくカードキー取得ごとに を使用し、別のペイロードも使用します。このメッセージは Visa でも使用されるため、要素名には JCB にも使用されますが、その名前が付けられます。ペイロードの内容の詳細については、スキームドキュメントにお問い合わせください。

キーの作成

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN10"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

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/pw3s6nl62t5ushfk",
                            "KeyAttributes": {
                                "KeyUsage": "TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS",
                                "KeyClass": "SYMMETRIC_KEY",
                                "KeyAlgorithm": "TDES_2KEY",
                                "KeyModesOfUse": {
                                    "Encrypt": false,
                                    "Decrypt": false,
                                    "Wrap": false,
                                    "Unwrap": false,
                                    "Generate": false,
                                    "Sign": false,
                                    "Verify": false,
```

```
                            "DeriveKey": true,
                            "NoRestrictions": false
                        }
                    },
                    "KeyCheckValue": "08D7B4",
                    "KeyCheckValueAlgorithm": "ANSI_X9_24",
                    "Enabled": true,
                    "Exportable": true,
                    "KeyState": "CREATE_COMPLETE",
                    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                    "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
                    "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
                }
            }
```

キーKeyArnを表す に注意してください。例えば、arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk です。これは次のステップで行います。

ARQC を検証する

Example

この例では、JCB CVN01 を使用して生成された ARQC を検証します。これは Visa メソッドと同じオプションを使用するため、 パラメータの名前を使用します。

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"
```

```
        }
```

調達と支払いのファシリテーター

通常、アクワイアラー、PSPs、および Payment Facilator には、発行者とは異なる暗号化要件があります。一般的ユースケースには以下が含まれます。

データ復号

データ (特にパンデータ) は、支払いターミナルによって暗号化され、バックエンドによって復号化する必要がある場合があります。データの復号化とデータの暗号化は、TDES、AES、DUKPT 取得技術など、さまざまな方法をサポートしています。 AWS Payment Cryptography サービス自体も PCI P2PE に準拠しており、PCI P2PE 復号コンポーネントとして登録されています。

TranslatePin

PCI PIN への準拠を維持するために、取得するシステムには、安全なデバイスへの入力後にカード所有者ピンが明確に含まれないようにする必要があります。したがって、ターミナルからダウンストリームシステム (支払いネットワークや発行者など) にピンを渡すには、支払いターミナルが使用したキーとは異なるキーを使用してピンを再暗号化する必要があります。Translate Pin は、暗号化されたピンを servicebbb で 1 つのキーから別のキーに安全に変換することで、これを実現します。このコマンドを使用すると、TDES、AES、DUKPT 取得などのさまざまなスキームと、ISO-0, ISO-3ISO-4 などのピンブロック形式の間でピンを変換できます。

VerifyMac

支払いターミナルからのデータは、転送中にデータが変更されていないことを確認するために MAC によって指定される場合があります。Mac と GenerateMac が、ISOISO-9797-1 アルゴリズム ISO-9797-13 (小売 MAC)、CMAC 手法で使用する TDES、AES、DUKPT 導出手法など、対称キーを使用してさまざまな手法をサポートしていることを確認します。

その他のトピック

- 動的キーの使用

動的キーの使用

動的キーを使用すると、 などの暗号化オペレーションに 1 回限りまたは制限付き使用キーを使用できます`EncryptData`。このフローは、キーマテリアルが頻繁にローテーションし (すべてのカード

トランザクションなど)、キーマテリアルをサービスにインポートしないようにしたい場合に使用できます。存続期間の短いキーは、softPOS/Mpoc またはその他のソリューションの一部として使用できます。

> **Note**
>
> これは、 AWS Payment Cryptography を使用する一般的なフローの代わりに使用できます。ここでは、暗号化キーが作成またはサービスにインポートされ、キーはキーエイリアスまたはキー ARN を使用して指定されます。

次のオペレーションは動的キーをサポートしています。

- EncryptData
- DecryptData
- ReEncryptData
- TranslatePin

データの復号

次の例は、復号コマンドとともに動的キーを使用する方法を示しています。この場合のキー識別子は、復号キー (TR-31 形式のラップキーパラメータで提供される) を保護するラップキー (KEK) です。ラップされたキーは、復号コマンドとともに B または D の使用方法とともに使用する D0 の主要な目的とします。

Example

```
$ aws payment-cryptography-data decrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/ov6icy4ryas4zcza
 --cipher-text 1234123412341234123412341234123A --decryption-attributes
 'Symmetric={Mode=CBC,InitializationVector=1234123412341234}'   --wrapped-key
 WrappedKeyMaterial={"Tr31KeyBlock"="D0112D0TN00E0000B05A6E82D7FC68B95C84306634B0000DA4701BE9BC
```

```
{
   "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
   "KeyCheckValue": "0A3674",
```

```
   "PlainText": "2E138A746A0032023BEF5B85BA5060BA"
}
```

ピンの翻訳

次の例は、動的キーと変換ピンコマンドを使用して、動的キーから半静的アクワイアラー作業キー (AWK) に変換する方法を示しています。この場合の受信キー識別子は、TR-31 形式で提供される動的ピン暗号化キー (PEK) を保護するラッピングキー (KEK) です。ラップされたキーは、B または D の使用モードP0とともに の主要な目的とします。送信キー識別子は タイプのキーTR31_P0_PIN_ENCRYPTION_KEYであり、Encrypt=true、Wrap=true の使用モードです。

Example

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-block
 "C7005A4C0FA23E02" --incoming-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}'
 --incoming-key-identifier alias/PARTNER1_KEK  --outgoing-key-
identifier alias/ACQUIRER_AWK_PEK --outgoing-translation-attributes
 IsoFormat0="{PrimaryAccountNumber=171234567890123}"  --incoming-wrapped-key
 WrappedKeyMaterial={"Tr31KeyBlock"="D0112P0TB00S0000EB5D8E63076313162B04245C8CE351C956EA4A16CC
```

```
{
   "PinBlock": "2E66192BDA390C6F",
   "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
   "KeyCheckValue": "0A3674"
}
```

AWS Payment Cryptography のリージョン固有の機能

特定の機能はリージョン固有であり、使用されていない場合があります。これらの機能の詳細については、このセクションで説明します。

AS2805

オーストラリア標準 2805 (AS2805) は、主にカードベースの支払い取引に使用される電子送金の標準です。オーストラリア標準規格によって管理されています。この標準は、メッセージ形式から暗号化標準まで、さまざまなトピックをカバーする 6 冊の本で構成されています。

パート 6 では、host-to-host (node-to-node) 通信や関連する暗号化要件など、キー管理に関するガイダンスを提供しますが、その他の側面については他の部分で説明します。この標準の暗号化はすべて、現在 TDES に基づいています。

> **Note**
>
> AS2805 は現在、ap-southeast-2 リージョンで使用できます。近い将来、追加のリージョンに展開される予定です。

AS2805 には、他の実装といくつかの違いがあります。以下に要約します。

キー保護

TR-31/X9.143 などのキーブロックではなくキーバリアントに依存します。 AWS Payment Cryptography は、すべてのキーを内部的にキーブロックとして保存しますが、AS2805 で定義されたバリアントを使用したインポート、エクスポート、計算を許可します。

一方向キー

AS2805 では、単方向キーの使用が義務付けられています。両方のノードがメッセージ認証コード (MAC) を生成する必要がある場合は、2 つのキーを使用します。

ピンブロック

AS2805 は、トランザクションごとに一意のピン暗号化キーのキー取得手法を定義します。これは DUKPT の代わりに使用できます。AS2805 スキームは、DUKPT によるトランザクションカウンターの使用と比較して、トランザクションデータ (トレース番号とトランザクション量) に依存します。

キー交換の検証

ピンキーなどの作業キーの交換を開始する前に、KEK を検証するプロセスを定義します。他のスキームでは、KEK は頻繁に交換されず、KCV を使用して検証されます。

AS2805 は、キーブロックではなくキーバリアントの概念を使用して、キーが意図した (および唯一の) 目的にのみ使用されるようにします。以下は、 AWS キーを使用して他の暗号化関数をインポート、エクスポート、または実行するときに Payment Cryptography がバリアントとキーブロック間でマッピングする方法です。

AS2805 キータイプ	AWS Payment Cryptography キータイプ
TERMINAL_MAJOR_KEY_VARIANT_00	TR31_K0_KEY_ENCRYPTION_KEY
PIN_ENCRYPTION_KEY_VARIANT_28	TR31_P0_PIN_ENCRYPTION_KEY
MESSAGE_AUTHENTICATION_KEY_VARIANT_24	TR31_M0_ISO_16609_MAC_KEY
DATA_ENCRYPTION_KEY_VARIANT_22	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY
VARIANT_MASK_82、VARIANT_MASK_82C0	KEK 検証プロセスの一部として使用できるオプション。これらのキータイプはエフェメラルであり、 サービスによって保存されません。

node1 と node2 の 2 つのノードがある場合、次の例は node1 の観点から示しています。 AWS Payment Cryptography は、プロセスの両側からの APIs をサポートしています。

トピック

- 初期キー (KEK) 交換
- KEK の検証
- 作業キーの作成と送信
- 作業キーのエクスポート
- ピン翻訳
- Mac の生成と検証

初期キー (KEK) 交換

AS2805 では、各側には独自の KEK があります。KEK (複数可) は、送信側がキーを保護/ラップして node2 に送信する必要がある場合に使用される送信側キーを指します。KEK(r) は、反対側 (node2) 側で作成されたキーです。

> **Note**
>
> これらの用語は相対的です。一方の側がキーを作成し (送信側) 、もう一方の側がキーを受け取ります。したがって、KEY1 を指定すると、node1 では KEK (複数可) 、node2 では KEK (r) と呼ばれます。

AS2805 の KEK は常にキータイプ = TR31_K0_KEY_ENCRYPTION_KEY です。これは、キーブロックではなく暗号文を保護するために使用されるためです。AS2805 6.1 で定義されているように、これは TERMINAL_MAJOR_KEY_VARIANT_00 にマッピングされます

手順:

1. キーを作成する

   CreateKey API を使用してキーを作成します。TR31_K0_KEY_ENCRYPTION_KEY タイプのキーを作成します。

2.node2 とキーを交換する方法を決定する

   KEK をカウンターパーティと交換する方法を決定します。AS2805 の場合、最も一般的な相互運用可能な方法は RSA ラップです。

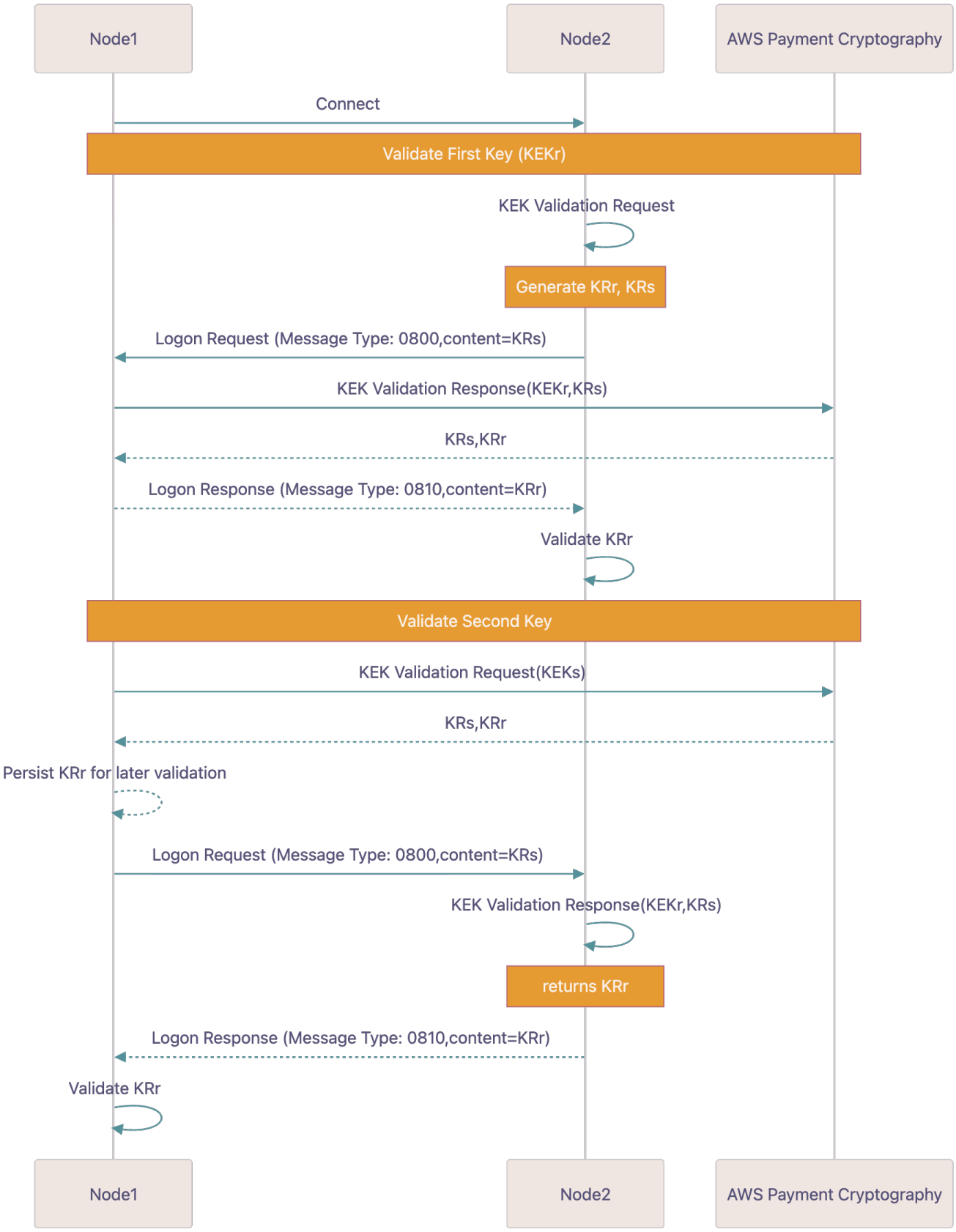
3.KEKsエクスポート

   上記の選択に基づいて、node2 からパブリックキー証明書を受け取ります。その証明書を使用してエクスポートを実行し、キーを保護します (ECDH を使用している場合はキーを取得します) 。

4. KEKr をインポートする

   上記の選択に基づいて、パブリックキー証明書を node2 に送信します。その証明書を使用してインポートを に実行し、ノード 2 の KEKr を サービスにロードします。

KEK の検証


Node1
Node2
AWS Payment Cryptography
Connect
Validate First Key (KEKr)
KEK Validation Request
Generate KRr, KRs
Logon Request (Message Type: 0800,content=KRs)
KEK Validation Response(KEKr,KRs)
KRs,KRr
Logon Response (Message Type: 0810,content=KRr)
Validate KRr
Validate Second Key
KEK Validation Request(KEKs)
KRs,KRr
Persist KRr for later validation
Logon Request (Message Type: 0800,content=KRs)
KEK Validation Response(KEKr,KRs)
returns KRr
Logon Response (Message Type: 0810,content=KRr)
Validate KRr
Node1
Node2
AWS Payment Cryptography

サービス (node1) が node2 に接続すると、各側は KEK 検証と呼ばれるプロセスを使用した後続のオペレーションに同じ KEK を使用していることを確認します。

1. 最初のキーを検証するステップ

1.1 KRs受信

Node2 は KRs を生成し、ログオンプロセスの一環として送信します。Payment Cryptography AWS を使用して、この値または別のソリューションを生成できます。

1.2 KEK 検証レスポンスを生成する

ノードは、ステップ 1 で提供された KEK(r) と KRs として入力を含む KEK 検証レスポンスを生成します。

Example

```
cat >> generate-kek-validation-response.json
{
    "KekValidationType": {
        "KekValidationResponse": {
            "RandomKeySend": "9217DC67B8763BABCFDF3DADFCD0F84A"
        }
    },
    "RandomKeySendVariantMask": "VARIANT_MASK_82",
    "KeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza"
}
```

```
$ aws payment-cryptography-data generate-as2805-kek-validation --cli-input-json
 file://generate-kek-validation-response.json
```

```
{
 "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
 "KeyCheckValue": "0A3674",
 "RandomKeyReceive": "A4B7E249C40C98178C1B856DB7FB76EB",
 "RandomKeySend": "9217DC67B8763BABCFDF3DADFCD0F84A"
}
```

1.3 計算された KRr を返す

計算された KRr を node2 に返します。そのノードは、ステップ 1 の計算値と比較します。

2.2 番目のキーを検証するステップ

2.1 KRr と KRs を生成する

Payment Cryptography を使用して、ノードはランダムな値とこの値の反転 (反転) AWS コピーを生成します。サービスは、KEK (複数可) によってラップされたこれらの値の両方を出力します。これらは KR (複数可) および KR (r) と呼ばれます。

Example

```
cat >> generate-kek-validation-request.json
{
    "KekValidationType": {
        "KekValidationRequest": {
            "DeriveKeyAlgorithm": "TDES_2KEY"
        }
    },
"RandomKeySendVariantMask": "VARIANT_MASK_82",
    "KeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
rhfm6tenpxapkmrv"
}
```

```
$ aws payment-cryptography-data generate-as2805-kek-validation --cli-input-json
 file://generate-kek-validation-request.json
```

```
{
 "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
rhfm6tenpxapkmrv",
 "KeyCheckValue": "DC1081",
 "RandomKeyReceive": "A4B7E249C40C98178C1B856DB7FB76EB",
 "RandomKeySend": "9217DC67B8763BABCFDF3DADFCD0F84A"
}
```

2.2 node2 に KRsを送信する

KRsします。後で検証できるように KRr を保持します。

2.3 Node2 が KEK 検証レスポンスを生成する

Node2 は KEKr と KRs を使用し、KRr を生成してサービスに送り返します。

2.4 レスポンスを検証する

ステップ 1 の KRr とステップ 3 から返された値を比較します。一致する場合は、続行します。

作業キーの作成と送信

AS2805 で使用される一般的な作業キーには、次の 2 つのキーセットが含まれます。

ゾーンピンキー (ZPK)、ゾーン暗号化キー (ZEK)、ゾーン認証キー (ZAK) などのノード間のキー。

DUKPT を使用していない場合、ターミナルメインキー (TMK) やターミナルピンキー (TPK) などのターミナルとノード間のキー。

> **Note**
>
> ターミナルキーごとにキーを最小限に抑え、可能な限り少数のキーを使用する TR-34 や DUKPT などの手法を活用することをお勧めします。

Example

この例では、オプションのタグを使用して、このキーの目的と使用を追跡しています。タグはシステムの暗号化関数の一部としては使用されませんが、分類、財務追跡、IAM ポリシーの適用に使用できます。

```
cat >> create-zone-pin-key.json
{
    "KeyAttributes": {
        "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY",
        "KeyClass": "SYMMETRIC_KEY",
        "KeyAlgorithm": "TDES_2KEY",
        "KeyModesOfUse": {
            "Encrypt": true,
            "Decrypt": true,
            "Wrap": true,
            "Unwrap": true,
            "Generate": false,
            "Sign": false,
            "Verify": false,
            "DeriveKey": false,
            "NoRestrictions": false
        }
    },
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "Exportable": true,
```

```
    "Enabled": true,
    "Tags": [
        {
            "Key": "AS2805_KEYTYPE",
            "Value": "ZONE_PIN_KEY_VARIANT28"
        }
    ]
}
```

```
$ aws payment-cryptography-data create-key --cli-input-json file://create-zone-pin-key.json --region ap-southeast-2
```

```
{
 "Key": {
 "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/alsuwfxug3pgy6xh",
 "KeyAttributes": {
 "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY",
 "KeyClass": "SYMMETRIC_KEY",
 "KeyAlgorithm": "TDES_2KEY",
 "KeyModesOfUse": {
 "Encrypt": true,
 "Decrypt": true,
 "Wrap": true,
 "Unwrap": true,
 "Generate": false,
 "Sign": false,
 "Verify": false,
 "DeriveKey": false,
 "NoRestrictions": false
 }
 },
 "KeyCheckValue": "9A325B",
 "KeyCheckValueAlgorithm": "ANSI_X9_24",
 "Enabled": true,
 "Exportable": true,
 "KeyState": "CREATE_COMPLETE",
 "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
 "CreateTimestamp": "2025-12-17T09:05:27.586000-08:00",
 "UsageStartTimestamp": "2025-12-17T09:05:27.570000-08:00"
 }
}
```

作業キーのエクスポート

他の当事者との互換性を維持するために、 AWS Payment Cryptography はTR-31 などのキーブロックの代わりにキーバリアントを使用する AS2805 対称キーラッピング手法をサポートしています。複数のキーが当事者間で共有されている場合は、それぞれを個別にエクスポートする必要があります。データが双方向で送信される場合、各側でメッセージ認証コードを生成するために使用される ZAK (複数可) や ZAK (r) など、同じタイプの当事者間に 2 つのキーが存在する可能性があります。

これらの形式でインポートおよびエクスポートする追加のパラメータは、 コマンドで指定します。

```
cat >> export-zone-pin-key.json
{
    "ExportKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
    "KeyMaterial": {
        "As2805KeyCryptogram": {
            "WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/rhfm6tenpxapkmrv",
            "As2805KeyVariant: "PIN_ENCRYPTION_KEY_VARIANT_28"
        }
    }
}
```

```
$ aws payment-cryptography-data export-key --cli-input-json file://export-zone-pin-
key.json --region ap-southeast-2
```

```
{
    "WrappedKey": {
        "KeyCheckValue": "DC1081",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyMaterial": "HDC10AEF038E695DDD72AF08DC1BB422D",
        "WrappedKeyMaterialFormat": "KEY_CRYPTOGRAM",
        "WrappingKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
rhfm6tenpxapkmrv"
    }
}
```

ピン翻訳

AS2805 では、セクション 6.4 でセッション固有のキー取得モードについて説明します。これは DUKPT と同様の目的を果たし、どちらのアルゴリズムも DUKPT がセクション 6.7 で説明

されているように使用できます。このスキームでは、セッションピンキー (KPE と呼ばれる) は、SystemTraceAuditNumber(STAN) と TransactionAmount を派生データとして使用してターミナルピンキーから派生します。

翻訳ピンは、さまざまな形式との間で翻訳できる一般的な関数です。この例では、ピンを支払いネットワークに送信するときなどに、KPE からピン暗号化キー (PEK) に変換します。

```
cat >> translate-pin-as2805.json
{
    "EncryptedPinBlock": "B3B34B43BAB5F81A",
    "IncomingKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
    "IncomingTranslationAttributes": {
        "IsoFormat0": {
            "PrimaryAccountNumber": "9999179999900013"
        }
    },
      "IncomingAs2805Attributes": {
        "SystemTraceAuditNumber": "000348",
        "TransactionAmount": "000000000328"
    },
    "OutgoingKeyIdentifier": "",
    "OutgoingTranslationAttributes": {
        "IsoFormat0": {
            "PrimaryAccountNumber": "9999179999900013"
        }
    }
}
```

```
$ aws payment-cryptography-data translate-pin-data --cli-input-json file://translate-
pin-as2805.json  --region ap-southeast-2
```

```
{
    "WrappedKey": {
        "KeyCheckValue": "DC1081",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyMaterial": "HDC10AEF038E695DDD72AF08DC1BB422D",
        "WrappedKeyMaterialFormat": "KEY_CRYPTOGRAM",
        "WrappingKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
rhfm6tenpxapkmrv"
    }
```

```
}
```

Mac の生成と検証

MAC コマンドの生成と検証は、MACs、CMAC、EMV MAC など、さまざまな MAC をサポートしています。AS2805 の場合、AS2805.4.1 で定義されている追加のバリエーションがあります。通常、AS2805 では、受信メッセージはこの MAC を使用して検証され、送信メッセージには MAC も含まれます。

```
cat verify-mac.json
{
    "KeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6",
    "Mac": "86304058",
    "MessageData": "73D8BA54D3852951DAEA41",
    "VerificationAttributes": {
        "Algorithm": "AS2805_4_1"
    }
}
```

```
$ aws payment-cryptography-data verify-mac --cli-input-json file://verify-mac.json --
region ap-southeast-2
```

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

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を活用できます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。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 — AWS 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AWS 責任があります。 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。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。 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。
- クラウド内のセキュリティ — 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、ユーザーは、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 を使用する際に責任共有モデルを適用する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ように 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 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
- ハイブリッドポスト量子 TLS の使用
- 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

AWS Payment Cryptography でのデータ保護には、 AWS 責任共有モデルが適用されます。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。ユーザーは、このインフラストラクチャでホストされ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に関する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「AWS 責任共有モデルおよび GDPR」のブログ記事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 アイデンティティセンターまたは 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 は必須ですが、TLS 1.3 を推奨します。
- 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。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中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 などの高度な管理された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 140-3 検証済み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) 140-3」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を、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含め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 タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する

場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

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 が生成またはロードする支払いキーのキーマテリアルは、 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、キー識別子を使用して API オペレーションのキーを表します。

ただし、一部の API オペレーションでは、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します。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。

Payment AWS 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 で「強力な暗号化」として定義された 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 1.2 ~ 1.3 およびハイブリッドポスト量子 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。

ネットワーク間の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と一連の API オペレーションをサポートしています。これにより、支払いキーを作成および管理し、暗号化オペレーションで使用できます。

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仮想プライベートクラウド (VPC) から AWS Payment Cryptography に直接接続するには、AWS PrivateLink を搭載した VPC エンドポイントを使用します。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、フォールトトレランス、および拡張性が優れています。

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別サービスです。

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

データと認可情報が、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手順で保護されています。

AWS が公開した 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、市販の PCI PTS HSM 認定ハードウェアセキュリティモジュール (HSM) 専用のハードウェアセキュリティモジュール (HSM) によってサポートされています。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場合、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 接続を使用せずに、VPC を AWS Payment Cryptography に直接接続します。VPC 内のインスタンスは、 AWS Payment Cryptography と通信するためにパブリック IP アドレスを必要としません。

Regions

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 用の VPC エンドポイントの作成
- AWS Payment Cryptography VPC エンドポイントへの接続
- VPC エンドポイントへのアクセスの制御
- ポリシーステートメントでの VPC エンドポイントの使用
- VPC エンドポイントのログ記録

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

> **Note**
>
> VPC エンドポイントでは、わずか 1 つのアベイラビリティーゾーン (AZ) でサービスに接続できますが、高可用性と冗長性の目的で 3 つのアベイラビリティーゾーンに接続することをお勧めします。

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 からすべての AWS Payment Cryptography コントロールプレーンオペレーションと AWS Payment Cryptography データプレーン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。一方または両方のサブサービスを設定できます AWS PrivateLink 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 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
```

たとえば、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、アプリケーションやコマンドで 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 https://
vpce-1234abcdf5678c90a-09p7654s-us-east-1a.ec2.us-east-1.vpce.amazonaws.com
```

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を変更す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、ModifyVpcattribute 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」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 CreateVPcendPoint オペレーションまたは ModifyVPcendPoint 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」でインターフェイスエンドポイントの作成方法およびインターフェイスエンドポイントの変更方法を確認してください。

JSON ポリシードキュメントの記述と書式設定については、『 IAM ユーザーガイド』の「IAM JSON ポリシーリファレンス 」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 リクエストを成功させるには、プリンシパルに 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、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ためのアクセス許可をプリンシパルに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。

たとえば、キーポリシーは、特定の AWS Payment Cryptography キーで Decrypt を呼び出すアクセス許可をプリンシパルに付与する場合があります。ただし、VPC 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`Decrypt`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は、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 を呼び出すことを許可する場合があります。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。

Payment Cryptography の VPC AWS エンドポイントポリシーの例を次に示します。このポリシーを VPC エンドポイントにアタッチすると、 ExampleUserは VPC エンドポイントを使用して、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ができます。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ます。

```
{
   "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キーに対する 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 マネジメントコンソール または DescribeVpcEndpoints 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==`vpce-1234abcdf5678c90a`].[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 から送信された場合、または VPC エンドポイントを使用している場合、 AWS Payment Cryptography 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 エンドポイントと VPC エンドポイントサービスのアイデンティティおよびアクセス管理のページ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ない CreateKey などのオペレーションへのアクセスを制御できます。

たとえば、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 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 AWS PrivateLink 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。

JSON

```
{
    "Id": "example-key-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableIAMPolicies",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "payment-cryptography:*"
            ],
            "Resource": "*"
```

```
        },
        {
            "Sid": "RestrictUsageToMyVPCEndpoint",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "payment-cryptography:EncryptData",
                "payment-cryptography:DecryptData"
            ],
            "Resource": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": "vpce-1234abcdf5678c90a"
                }
            }
        }

    ]
}
```

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 キーを から取得した場合にのみ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。

JSON

```
{
    "Id": "example-key-2",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAdminActionsFromVPC12345678",
```

```
            "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": "AllowKeyUsageFromVPC2b2b2b2b",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "payment-cryptography:Encrypt*",
                "payment-cryptography:Decrypt*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-2b2b2b2b"
                }
            }
        },
        {
            "Sid": "AllowListReadActionsFromEverywhere",
            "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、許可されていたリクエストは に記録されませんAWS CloudTrail。

たとえば、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:i-98761b8890c09a34a",
          "arn": "arn:aws:sts::111122223333:assumed-role/samplerole/
i-98761b8890c09a34a",
          "accountId": "111122223333",
          "accessKeyId": "TESTXECZ5U2ZULLHHMJG",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "TESTXECZ5U9M4LGF2N6Y5",
                  "arn": "arn:aws:iam::111122223333:role/samplerole",
                  "accountId": "111122223333",
                  "userName": "samplerole"
              },
              "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,
```

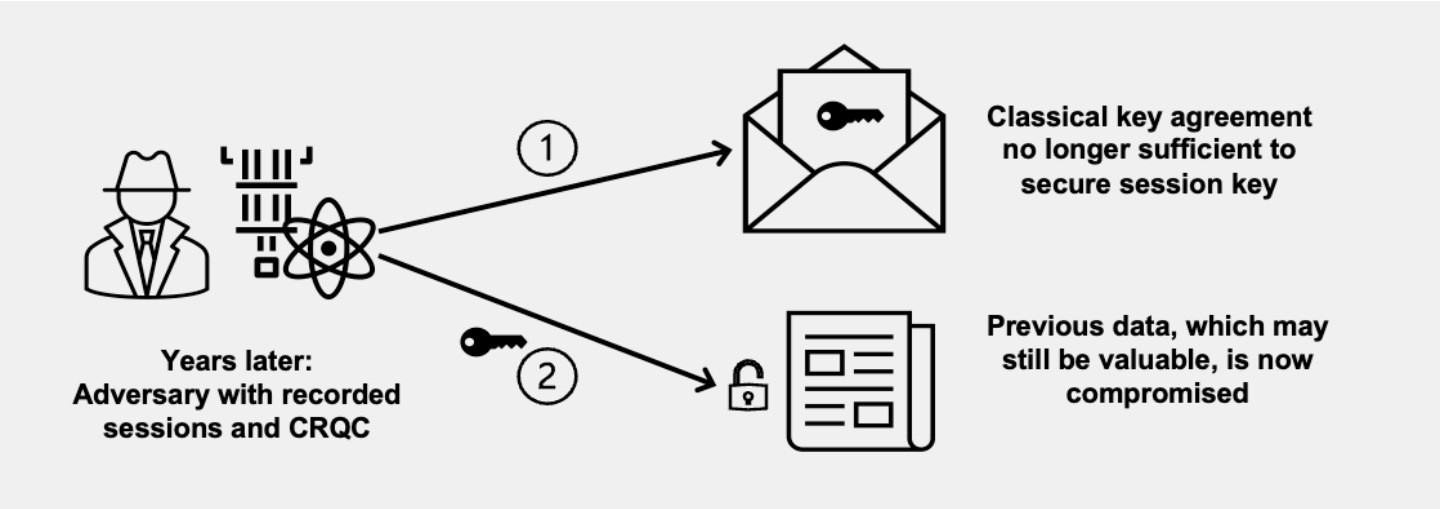
```
                    "verify": true,
                    "deriveKey": false,
                    "noRestrictions": false
                }
            },
            "keyCheckValue": "A486ED",
            "keyCheckValueAlgorithm": "ANSI_X9_24",
            "enabled": true,
            "exportable": 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": "vpce-1234abcdf5678c90a",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "vpce-1234abcdf5678c90a-
oo28vrvr.controlplane.payment-cryptography.us-east-1.vpce.amazonaws.com"
    }
}
```


ハイブリッドポスト量子 TLS の使用

AWS Payment Cryptography およびその他の多くのサービスは、Transport Layer Security (TLS) ネットワーク暗号化プロトコルのハイブリッドポスト量子キー交換オプションをサポートしています。この TLS オプションは、API エンドポイントに接続するとき、または AWS SDKs を使用するときに使用できます。これらのオプションのハイブリッドポスト量子キー交換機能は、現在使用している TLS 暗号化と同等以上に安全であり、セキュリティ上のさらなる長期的な利点をもたらす可能性があります。

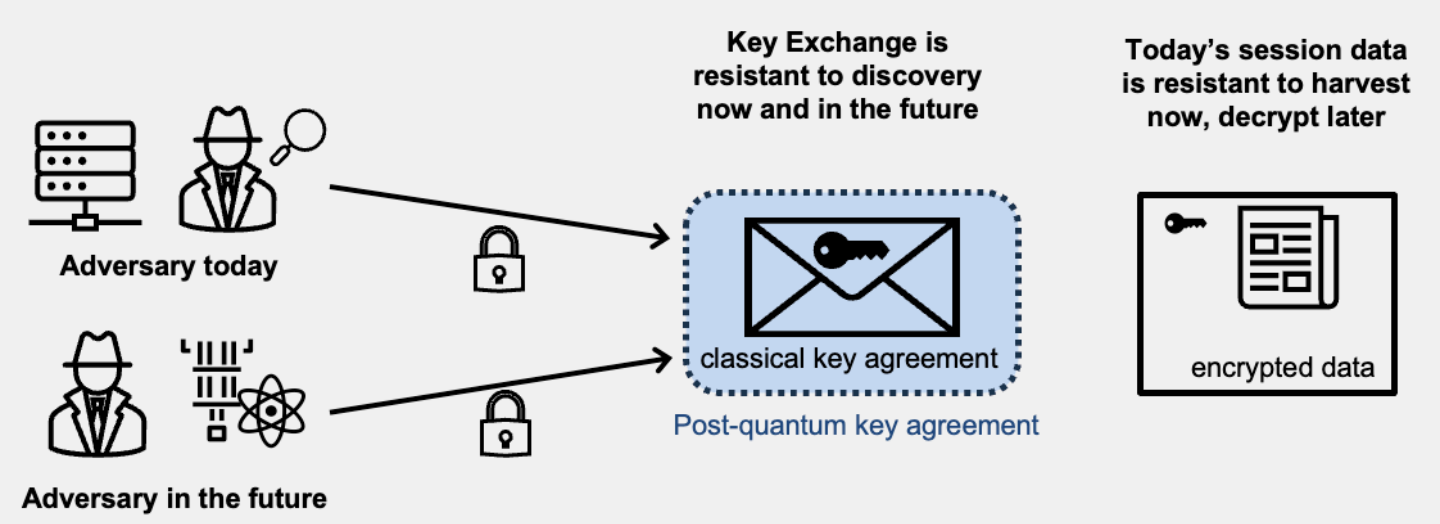
有効なサービスに送信するデータは、Transport Layer Security (TLS) 接続によって提供される暗号化によって転送中に保護されます。 AWS Payment Cryptography が TLS セッションでサポートする

RSA と ECC に基づく従来の暗号スイートは、現在のテクノロジーではキー交換メカニズムに対するブルートフォース攻撃を実行不可能にします。ただし、大規模または暗号に関連する量子コンピュータ (CRQC) が将来実用的になった場合、既存の TLS キー交換メカニズムはこれらの攻撃の影響を受けやすくなります。攻撃者は、今後復号化できることを期待して、暗号化されたデータの収集を開始する可能性があります (今すぐ収集し、後で復号化します) 。TLS 接続を介して渡されるデータの長期的な機密性に依存するアプリケーションを開発している場合は、大規模な量子コンピュータが使用できるようになる前に、ポスト量子暗号化に移行する計画を検討する必要があります。 AWS は将来に備えており、準備も万端です。



現在暗号化されているデータを潜在的な将来の攻撃から保護するために、 AWS は量子耐性アルゴリズムまたはポスト量子アルゴリズムの開発に暗号コミュニティに参加しています。 AWS は、クラシック要素とポスト量子要素を組み合わせたハイブリッドポスト量子キー交換暗号スイートを実装し、TLS 接続がクラシック暗号スイートと同等以上に強力であることを確認します。

これらのハイブリッド暗号スイートは、最新バージョンの AWS SDKs を使用する場合に、本稼働ワークロードで使用できます。この動作を有効/無効にする方法の詳細については、「」を参照してください。 ???



TLS におけるハイブリッドポスト量子キー交換について

が AWS 使用するアルゴリズムは、TLS で現在使用されている従来のキー交換アルゴリズムである Elliptic Curve Diffie-Hellman (ECDHML-KEM) と、米国国立標準技術研究所 (NIST) が最初の標準ポスト量子キーアグリーメントアルゴリズムとして指定したパブリックキー暗号化およびキー確立アルゴリズムである Module-Lattice-Based Key-Encapsulation Mechanism () を組み合わせたハイブリッドです。このメカニズムは、各アルゴリズムを独立して使用してキーを生成します。次に、2つのキーを暗号的に組み合わせます。

PQC の詳細

米国国立標準技術研究所 (NIST) のポスト量子暗号プロジェクトの詳細については、「Post-Quantum Cryptography」(ポスト量子暗号化) を参照してください。

NIST ポスト量子暗号標準化については、「Post-Quantum Cryptography Standardization」(ポスト量子暗号標準化) を参照してください。

ハイブリッドポスト量子 TLS の有効化

AWS SDKsとツールには、言語とランタイムによって異なる暗号化機能と設定があります。AWS SDK またはツールが現在 PQ TLS サポートを提供する方法は 3 つあります。

トピック

- PQ TLS がデフォルトで有効になっている SDKs
- オプトイン PQ TLS サポート

- System OpenSSL に依存する SDKs
- PQ TLS をサポートする予定がない AWS SDKs とツール

PQ TLS がデフォルトで有効になっている SDKs

> **Note**
>
> 6-Nov-2025 日現在、MacOS および Windows 用の AWS SDK とその基盤となる CRT ライブラリは TLS 用のシステムライブラリを使用するため、これらのプラットフォームでの PQ TLS 機能は一般的にシステムレベルのサポートによって決まります。

AWS SDK for Go

AWS SDK for Go は、標準ライブラリが提供する Golang 独自の TLS 実装を使用します。Golang は v1.24 の時点で PQ TLS をサポートおよび優先しているため、AWS SDK for Go ユーザーは Golang を v1.24 にアップグレードするだけで PQ TLS を有効にできます。

AWS SDK for JavaScript (ブラウザ)

AWS SDK for JavaScript (ブラウザ) はブラウザの TLS スタックを使用するため、ブラウザランタイムがサポートし、希望する場合、SDK は PQ TLS をネゴシエートします。Firefox は v132.0 で PQ TLS のサポートを開始しました。Chrome は v131 での PQ TLS のサポートを発表しました。Edge は、デスクトップの場合は v120、Android の場合は 140 でオプトイン PQ TLS をサポートしています。

AWS SDK for Node.js

Node.js v22.20 (LTS) および v24.9.0 以降、Node.js は OpenSSL 3.5 を静的にリンクしてバンドルします。つまり、PQ TLS は、これらのバージョン以降ではデフォルトで有効になっており、優先されます。

AWS SDK for Kotlin

Kotlin SDK は、v1.5.78 の時点で Linux での PQ TLS をサポートしており、優先しています。AWS SDK for Kotlin の CRT ベースのクライアントは MacOS および Windows の TLS のシステムライブラリに依存しているため、PQ TLS のサポートは基盤となるシステムライブラリによって異なります。

AWS SDK for Rust

AWS SDK for Rust は、サービスクライアントごとに個別のパッケージ (Rust エコシステムでは「木箱」と呼ばれます) を配布します。これらはすべて統合された GitHub リポジトリで管理されますが、各サービスクライアントは独自のバージョンとリリース頻度に従います。統合 SDK は 8/29/25 に PQ TLS 設定をリリースしたため、それ以降にリリースされた個々のサービスクライアントバージョンはデフォルトで PQ TLS をサポートし、優先します。

特定のサービスクライアントの PQ TLS をサポートする最小バージョンを確認するには、関連する crates.io バージョン URL に移動し (たとえば、 AWS Payment Cryptography の がここにあります)、29-Aug-25 日以降に発行された最初のバージョンを見つけます。29-Aug-25 日以降に発行されたサービスクライアントバージョンでは、デフォルトで PQ TLS が有効になり、優先されます。

オプトイン PQ TLS サポート

AWS SDK for C++

デフォルトでは、C++ SDK は libcurl や WinHttp などのプラットフォームネイティブクライアントを使用します。Libcurl は通常、TLS のシステム OpenSSL に依存しているため、PQ TLS はシステム OpenSSL が ≥ v3.5 の場合にのみデフォルトで有効になります。このデフォルトは、C++ SDK v1.11.673 以降で上書きし、デフォルトで PQ TLS をサポートおよび有効にする AwsCrtHttpClient にオプトインできます。

オプトイン PQ TLS の構築に関する注意事項 このスクリプトを使用して SDK の CRT 依存関係を取得できます。ソースからの SDK の構築については、ここで説明しますが、追加の CMake フラグが必要になる場合があります。

```
    -DUSE_CRT_HTTP_CLIENT=ON \
    -DUSE_TLS_V1_2=OFF \
    -DUSE_TLS_V1_3=ON \
    -DUSE_OPENSSL=OFF \
```

AWS SDK for Java

v2 の時点で、AWS SDK for Java は、PQ TLS を実行するように設定できる AWS Common Runtime (AWS CRT) HTTP クライアントを提供します。v2.35.11 以降、AwsCrtHttpClient は、使用するすべての場所でデフォルトで PQ TLS を有効にし、優先します。

System OpenSSL に依存する SDKs

いくつかの AWS SDKs とツールは、TLS 用のシステムの libcrypto/libssl ライブラリに依存しています。最もよく使用されるシステムライブラリは OpenSSL です。バージョン 3.5 での OpenSSL 対応 PQ TLS サポートであるため、PQ TLS 用にこれらの SDKs とツールを設定する最も簡単な方法は、少なくとも OpenSSL 3.5 がインストールされているオペレーティングシステムディストリビューションで使用することです。

OpenSSL 3.5 を使用して Docker をサポートするすべてのシステムで PQ TLS を有効にするように Docker コンテナを設定することもできます。Python 用に設定する例については、Python でポスト量子 TLS を参照してください。

AWS CLI

AWS CLI インストーラでの PQ TLS サポートは間もなく開始されます。をすぐに有効にするには、オペレーティングシステムによって異なる AWS CLI の代替インストーラーを使用し、PQ TLS を有効にできます。

MacOS の場合は、Homebrew 経由で AWS CLI をインストールし、Homebrew 提供の OpenSSL がバージョン 3.5 以降にアップグレードされていることを確認します。これは「brew install openssl@3.6」で実行し、「brew list | grep openssl」で検証できます。

Ubuntu または Debian Linux の場合: 使用している Linux ディストリビューションに OpenSSL 3.5+ がシステム OpenSSL としてインストールされていることを確認します。次に、apt または PyPI を使用して AWS CLI をインストールします。これらの前提条件により、apt または PyPI によって提供された AWS CLI は、PQ-TLS をネゴシエートするように設定されます。インストールを検証する step-by-stepについては、「github repository and accompanying blog post」を参照してください。

AWS SDK for PHP

AWS SDK for PHP はシステム libssl/libcrypto に依存しています。PQ TLS を使用するには、少なくとも OpenSSL 3.5 がインストールされているオペレーティングシステムディストリビューションでこの SDK を使用します。

「AWS SDK for Python (Boto3)」

AWS SDK for Python (Boto3) は、システム libssl/libcrypto に依存しています。PQ TLS を使用するには、少なくとも OpenSSL 3.5 がインストールされているオペレーティングシステムディストリビューションでこの SDK を使用します。

AWS SDK for Ruby

AWS SDK for Ruby は、システム libssl/libcrypto に依存しています。PQ TLS を使用するには、少なくとも OpenSSL 3.5 がインストールされているオペレーティングシステムディストリビューションでこの SDK を使用します。

PQ TLS をサポートする予定がない AWS SDKs とツール

現在、以下の言語 SDKs とツールをサポートする予定はありません。

- AWS SDK for .NET
- Swift 用 AWS SDK
- AWS Tools for Windows PowerShell

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

AWS Payment Cryptography は、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。これらの機能は、暗号化キーの保護を強化し、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、キーブロックに関する PCI PIN ルールの組み込み適用など、意図した目的に使用できます。

> **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。

- Key Usage and Modes of Use: AWS Payment Cryptography は、ANSI X9 TR 31-2018 Interoperable Secure Key Exchange Key Block Specification で説明されているように、PCI PIN セキュリティ要件 18-3 に準拠し、キーの使用と使用モードの制限を適用します。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) をキーマテリアル自体に暗号的にバインドできます。 AWS Payment Cryptography は、キー暗号化キー (TR31_K0_KEY_ENCRYPTION_KEY) をデータ復号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 では、対称キーマテリアルや非対称プライベートキーマテリアルがクリアに公開されることはありません。
- 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

他の AWS サービスと同様に、お客様はセキュリティとコンプライアンスの責任共有モデルを明確に理解する必要があります。支払いを特にサポートするサービスとして、 AWS Payment Cryptography のお客様にとって、適用可能な PCI 標準への準拠は特に重要です。 AWS PCI DSS および PCI 3DS 評価には AWS Payment Cryptography が含まれます。これらのレポート AWS Artifact については、「 責任共有ガイド」の「サービスへの参照」を参照してください。PCI PIN セキュリティとPoint-to-Point暗号化 (P2PE) の評価は AWS Payment Cryptography に固有です。

このセクションでは、サービスコンプライアンスのステータスと範囲に関する情報と、アプリケーションの PCI PIN セキュリティと PCI P2PE 評価の計画に役立つ情報を提供します。

トピック

- サービスのコンプライアンス
- PIN コンプライアンス計画
- P2PE ソリューションでの AWS Payment Cryptography Decryption コンポーネントの使用

サービスのコンプライアンス

サードパーティーの監査者は、複数のコンプライアンスプログラムの一環として AWS Payment Cryptography のセキュリティと AWS コンプライアンスを評価します。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 と PCI 3DS 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。利用可能な認証とコンプライアンスガイド AWS Artifact については、「」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コンプライアンスプログラム」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。

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 AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

AWS Payment Cryptography を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- 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 コンプライアンスリソース — このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 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 CSPM— この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準と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。

PIN コンプライアンス計画

このガイドでは、 AWS Payment Cryptography を使用する PIN 処理アプリケーションの PCI PIN 評価を準備する必要があるドキュメントと証拠について説明します。

他の AWS のサービス およびコンプライアンス標準と同様に、サービスを安全に使用し、アクセスコントロールを設定し、PCI PIN 要件に従ってセキュリティパラメータを使用するのはお客様の責任です。このガイドでは、要件を満たすために適切な場合、これらの設定について説明します。

トピック

- 一般的なトピック
- 評価範囲
- トランザクション処理オペレーション

一般的なトピック

アプリケーションを HSM への接続から AWS Payment Cryptography などのマネージドサービスに移行すると、顧客と評価者に共通の問題と概念が生じます。このセクションでは、サービスの安全な使用がこれらの状況にどのように対処するかを明確にする情報を提供します。

トピック

- 共有責任
- 最小 HSM 設定
- 顧客と Modbus 間のキー交換

共有責任

アプリケーションに対する完全なセキュリティとコンプライアンスの責任を引き受けたお客様は、 AWS Payment Cryptography のキー管理、セキュリティコントロール、マネージド HSM 機能 (「サービス」) を活用できるようにコンプライアンスを再構築します。これにより、 AWS Payment Cryptography のサードパーティー評価で証明されているように AWS、一部の要件が完全に に移行されます。一部の要件は、お客様のアプリケーションと サービス間で共有されます。アプリケーションは以下を担当します。

- サービスに正確な情報を提供する
- サービスの推奨事項と PCI PIN セキュリティ要件に従ったセキュリティコントロールの使用
- サービスが提供するツールを使用して必要なセキュリティコントロールを実装する

お客様とその評価者は、 のコンプライアンス認証とともに公開された責任共有ガイドと実装ガイドを使用して AWS Artifact 、コントロールとコントロールモニタリングを実装し、評価を計画して完了します。

最小 HSM 設定

他の PCI 標準の基本標準である PCI データセキュリティ標準では、すべてのシステムをその機能に必要な最小限の機能で設定する必要があります。PCI PIN、P2PE、およびその他のソリューション標準は、この要件をソリューションHSMs に適用します。HSMs、ソリューションに必要な関数のみを有効にする必要があります。

AWS サービスはシステムとして扱われ、必要最小限の機能に設定する必要があります。Payment Card Industry Data Security Standard (PCI DSS) v4.0 on AWS では、IAM を使用して、ソリューションで使用される各 AWS サービスの最小機能を設定することをお勧めします。これは AWS Payment Cryptography にも適用されます。IAM ポリシーは、詳細なアクセス許可を有効にして、暗号化関数に依存するアプリケーションコンポーネントのみに制限します。

顧客と Modbus 間のキー交換

PIN PIN セキュリティ要件 8-4 および 15-2 では、キーの交換とロードにパブリックキーが認証され、整合性が保護されている必要があります。ANSI/ASC X9 TR-34 で機能的に記述され、PCI PIN Annex A によって管理される POI のリモートキーロードの場合、パブリックキーは Annex A2-compliantの認証局によって署名された証明書で最もよく伝達されます。組織間の交換の場合、パブリックキーは真正性と整合性のために他のメカニズムを使用します。

顧客と AWS 間のすべてのやり取りは AWS APIs を介して行われます。これにより、各 API コールが相互に認証され、TLS を使用したコールとレスポンスの整合性が保証されます。カスタマーアプリケーションの認証は、セキュリティトークンや SigV4 などのメカニズムを使用して AWS Identity and Access Management によって管理されます。AWS API エンドポイントは、AWS SDKs。その後、TLS は顧客と各 AWS API の間で渡されるすべてのデータの機密性と完全性を保証します。

Modbus APIs GetParametersForImport と ImportKey は、顧客からサービスへのキー転送を実装します。GetParametersForImport が提供する認証機関 (CA) は Annex A2-compliantしていませんが、安全でアカウントに固有です。この CA は要件 8-4 および 15-2 への準拠には依存できませんが、インポートされたキーの整合性検証を提供します。GetCertificateSigningRequest API を活用して、独自の CA を使用することもできます。

パブリックキー認証と整合性保証を提供するメカニズムは次のとおりです。

- AWS API 認証によって提供される認証
- キーの整合性は、証明書の ID 情報が信頼されていない場合でも、GetParametersForImport によって提供される証明書の MAC 機能によって提供されます。キーの整合性は、顧客と AWS 間のセッションを保護する TLS によって使用される MAC によっても保証されます。

Modbus が提供する証明書とキーブロックは、非対称メソッドによる証明書とキー保護の要件を指定する Annex A1 に準拠しています。

評価範囲

評価を計画する最初のステップは、範囲を文書化することです。PCI PIN の場合、範囲は PINs を保護するシステムとプロセスです。これには、暗号化キーとそれらを保護するデバイスの保護、points-of-interaction (POI)、HSMs、その他の安全な暗号化デバイス (SCD) とも呼ばれる支払いターミナルが含まれます。

お客様が全責任を負う要件には対応しません。これらの要件は、サービスの範囲外の領域に対処するためです。例えば、支払いターミナルの設定とプロビジョニングなどです。 AWS 「 Payment Cryptography Shared Responsibility Guide for PCI PIN」を参照してください。 AWS Artifact

トピック

- 共有責任
- 高レベルのネットワーク図
- キーテーブル
- ドキュメントリファレンス

共有責任

AWS Payment Cryptography は、Visa PIN セキュリティプログラムで定義され、「Amazon Web Services, LLC」の Visa グローバルサービスプロバイダーレジストリにリストされている Encryption and Support Organization (ESO) および PIN-Acquiring Third-Party Servicer (TPS) です。つまり、このサービスは、顧客の PIN 評価者 (PCI 認定 PIN 評価者または PCI QPA) によるさらなる評価を必要とせずに、PIN 取得サードパーティーの VisaNet プロセッサ (VNP)、PIN 取得クライアント VisaNet プロセッサ、およびその他の TPS および ESO プロバイダーが使用できます。

他のカードブランドや支払いネットワークプロバイダーは、 Visa PIN セキュリティプログラムに依存するか、独自のプログラムを持つ場合があります。他の支払いネットワークプログラムのサービスコンプライアンスに関する質問 AWS サポート については、 にお問い合わせください。

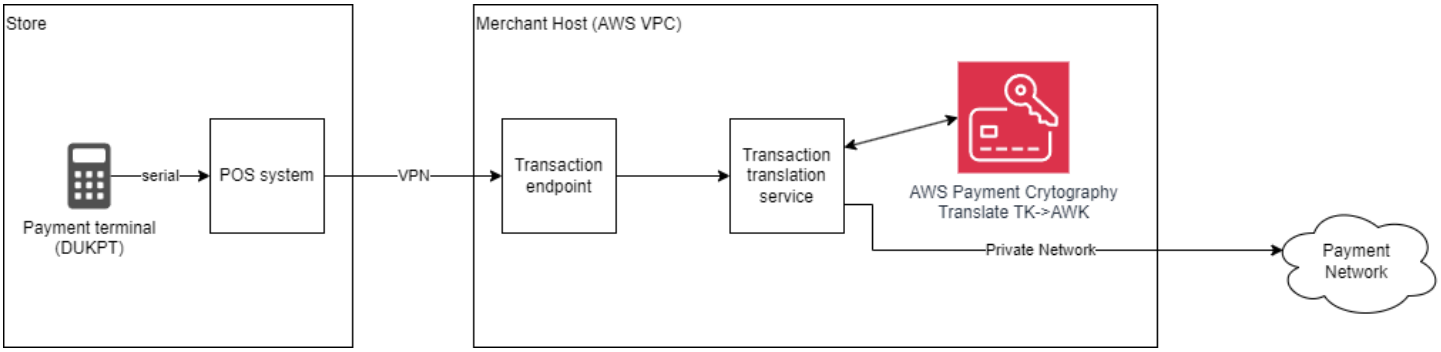
AWS には、PCI PIN Security Attestation of Compliance (AOC) と AWS Payment Cryptography に関する責任共有ガイドが用意されています AWS Artifact。PIN 処理でのサービスプロバイダーの使用は長年一般的ですが、PCI PIN セキュリティ標準はバージョン 3.1 までは、サードパーティーのサービスプロバイダー管理に対応していません。Visa PIN セキュリティプログラムも行いません。Customer QPA は、PCI DSS AOC および 責任共有ガイドで確立されたモデルに従って、該当する要件のテストが成功 AWSしたと言及しています。

高レベルのネットワーク図

PCI PIN レポートテンプレートでは、「PIN ベースのトランザクションの処理に関与するエンティティの場合、関連するキータイプの使用状況を含む PIN ベースのトランザクションフローを説明するネットワーク図を提供します。さらに、非対称手法を使用してリモートキー配布を行う KIFs とエンティティは、キーマテリアルフローを提供する必要があります。

AWS Payment Cryptography は、PCI PIN 評価の内部サービス構造を報告しています。図は、PIN 処理のためのサービス APIsの呼び出しを示しています。

AWS Payment Cryptography を使用した PIN アプリケーションの高レベルのネットワーク図の例:



キーテーブル

このレポートでは、PINs を保護するすべてのキーを直接または間接的に一覧表示する必要があります。サービスに存在するキーは、ListKeysAPI で一覧表示できます。

アプリケーションのキーを所有するすべてのリージョンとアカウントのキーリストを必ず指定してください。

ドキュメントリファレンス

AWS Payment Cryptography を安全に使用するためのベンダーのドキュメントと推奨事項は、「 ユーザーガイド」と「 API リファレンス」に記載されています。 https://docs.aws.amazon.com/payment-cryptography/latest/APIReference/Welcome.htmlこれらは、このガイダンスで必要に応じてリンクされています。

トランザクション処理オペレーション

PCI PIN の要件は、コントロール目標にまとめられています。各コントロール目標は、PINs。

トピック

- コントロールの目的 1: これらの要件によって管理されるトランザクションで使用される PINsは、安全性を確保する機器と方法論を使用して処理されます。
- コントロールの目的 2: PIN の暗号化/復号化および関連するキー管理に使用される暗号化キーは、キーを予測できないか、特定のキーが他のキーよりも可能性が高いと判断できないようにするプロセスを使用して作成されます。
- コントロールの目的 3: キーは安全な方法で伝達または送信されます。
- コントロールの目的 4: HSMsおよび POI PIN 承認デバイスへのキーロードは、安全な方法で処理されます。
- コントロールの目的 5: キーは、不正な使用を防止または検出する方法で使用されます。
- コントロールの目的 6: キーは安全な方法で管理されます。
- コントロールの目的 7: PINsとキーの処理に使用される機器は、安全な方法で管理されます。

コントロールの目的 1: これらの要件によって管理されるトランザクションで使用される PINsは、安全性を確保する機器と方法論を使用して処理されます。

要件 1: AWS Payment Cryptography で使用される HSMs は、PCI PIN 評価の一環として評価されました。サービスを使用しているお客様の場合、要件 1 ~ 3 および 1 ~ 4 は、サービスによって管理さ

れる HSM に対する「インプレース」です。HSM の結果には、テストが AWS QPA によって証明されたことが示されます。PIN 準拠証明書を参照できます AWS Artifact。POI などのソリューション内の他の SCD は、インベントリ化して参照する必要があります。

要件 2: 手順のドキュメントでは、担当者への漏洩、実装された PINs 翻訳プロトコル (複数可) 、オンラインおよびオフライン処理中の保護に関して、カード所有者 PIN を保護する方法を指定する必要があります。さらに、ドキュメントには、各ゾーンで使用される暗号化キー管理方法の概要が含まれている必要があります。

要件 3: POI は、安全な PIN 暗号化と送信用に設定する必要があります。 AWS Payment Cryptography は、要件 3-3 で指定された PIN ブロック変換のみをサポートします。

要件 4: アプリケーションは PIN ブロックを保存してはいけません。PIN ブロックは、暗号化されていても、トランザクションジャーナルやログに保持しないでください。サービスは PIN ブロックを保存せず、PIN 評価はそれらがログにないことを検証します。

PCI PIN セキュリティ標準は、標準に記載されているように、「ATM およびpoint-of-sale端末でのオンラインおよびオフラインの支払いカード取引処理中の個人識別番号 (PIN) ATMs」の取得に適用されます。ただし、この標準は、意図した範囲外の支払いの暗号化キー管理を評価するためによく使用されます。これには、PINsが保存される発行者のユースケースが含まれる場合があります。これらのケースの要件の例外は、評価の対象となる対象者と合意する必要があります。

コントロールの目的 2: PIN の暗号化/復号化および関連するキー管理に使用される暗号化キーは、キーを予測できないか、特定のキーが他のキーよりも可能性が高いと判断できないようにするプロセスを使用して作成されます。

要件 5: AWS Payment Cryptography によるキー生成は、PCI PIN 評価の一環として評価されました。これは、キーテーブルの「Generated by」列で指定できます。

要件 6: AWS Payment Cryptography に保持されているキーのセキュリティコントロールは、サービスの PCI PIN 評価の一部として評価されました。アプリケーション内および他のサービスプロバイダーとのキー生成に関連するセキュリティコントロールの説明を含めます。

要件 7: キーの生成方法を指定するキー生成ポリシードキュメントが必要であり、影響を受けるすべての当事者がこれらの手順/ポリシーを認識している必要があります。Modbus API を使用したキー作成の手順には、キー作成のアクセス許可を持つロールの使用と、キーを作成するスクリプトやその他のコードの実行の承認を含める必要があります。 AWS CloudTrail ログには、日付と時刻、キー ARN、ユーザー ID を含むすべての CreateKey イベントが含まれます。物理メディアにアクセスするための HSM シリアル番号とログは、サービスの PIN 評価の一部として評価されました。

コントロールの目的 3: キーは安全な方法で伝達または送信されます。

要件 8: AWS Payment Cryptography によるキー伝達は、PCI PIN 評価の一環として評価されました。 AWS Payment Cryptography へのインポート前とエクスポート後の転送のキー保護メカニズムを文書化する必要があります。このサービスは、正しい伝達を検証するためのすべてのキーのキーチェック値を提供します。

要件 8-4 では、パブリックキーの整合性と信頼性を保護する方法でパブリックキーを伝達する必要があります。アプリケーションと 間の伝達 AWS は AWS、 AWS Identity and Access Management TLS サーバー証明書を介したアプリケーションへの AWS API エンドポイント認証に対するアプリケーションの認証によって制御されます。さらに、 AWS Payment Cryptography からエクスポートまたはインポートされたパブリックキーには、一時的な顧客固有の CAs によって署名された証明書があります (GetPublicKeyCertificate」、GetParametersForImport」、および GetParametersForExport」を参照してください)。これらの CAsは PCI PIN Security Annex A2 に準拠していないため、認証の唯一の方法として使用することはできません。ただし、証明書は、認証を提供する IAM でパブリックキーの整合性を保証します。

非対称方法を使用してビジネスパートナーとパブリックキーを交換する場合は、安全なファイル交換ウェブサイトなどを使用して、通信チャネルを介してビジネスの認証を提供する必要があります。

要件 9: サービスはクリアテキストキーコンポーネントを使用または直接サポートしていません。

要件 10: サービスは、伝達のためにキーを保護する相対的なキー強度を適用します。 AWS Payment Cryptography へのインポート前とエクスポート後のキー伝達、およびキーのインポート、エクスポート、生成に正確な API と TR-31 パラメータの使用については、お客様の責任となります。キー伝達メカニズムと伝達に使用される暗号化キーのリストを説明する手順が文書化されている必要があります。

要件 11: 手順のドキュメントでは、キーの伝達方法を指定する必要があります。 AWS Payment Cryptography API を使用したキー伝達の手順には、キーのインポートとエクスポートのアクセス許可を持つロールの使用と、キーを作成するスクリプトやその他のコードの実行の承認を含める必要があります。 AWS CloudTrail ログには、すべての ImportKey イベントと ExportKey イベントが含まれます。

コントロールの目的 4: HSMsおよび POI PIN 承認デバイスへのキーロードは、安全な方法で処理されます。

要件 12: コンポーネントまたは共有からキーをロードする責任があります。HSM メインキーの管理は、サービスの PIN 評価の一部として評価されました。 AWS Payment Cryptography は、個々の共

有またはコンポーネントからキーをロードしません。「暗号化の詳細」セクションを参照してください。

要件 13 および 14: サービスからのインポート前とエクスポート後の転送のキー保護を記述する必要があります。

要件 15: AWS Payment Cryptography は、サービス内のすべてのキーのキーチェック値と、パブリックキーの整合性保証を提供します。アプリケーションは、これらのチェックを使用して、サービスへのインポートまたはサービスからのエクスポート後にキーを検証する責任があります。検証メカニズムが設定されていることを確認する手順を文書化する必要があります。

要件 15-2 では、パブリックキーの整合性と信頼性を保護する方法でロードする必要があります。ImportKey は GetParametersForImport とともに、提供された署名証明書の検証を提供します。提供された証明書が自己署名の場合、認証は安全なファイル交換などの別のメカニズムで提供する必要があります。

要件 16: 手順のドキュメントでは、キーをサービスにロードする方法を指定する必要があります。API を使用したキーのインポートの手順には、キーのインポート許可を持つロールの使用と、キーをロードするスクリプトやその他のコードの実行の承認を含める必要があります。 AWS CloudTrail ログには、すべての ImportKey イベントが含まれます。ログ記録メカニズムをドキュメントに含める必要があります。このサービスは、正しいキーロードを検証するために、すべてのキーのキーチェック値を提供します。

コントロールの目的 5: キーは、不正な使用を防止または検出する方法で使用されます。

要件 17: このサービスは、キー共有関係の追跡を可能にするキーのタグやエイリアスなどのメカニズムを提供します。さらに、キーが共有されるときに既知のキー値またはデフォルトのキー値が使用されないことを示すために、キーチェック値を個別に保持する必要があります。

要件 18: このサービスは、GetKey および ListKeys を介したキー整合性チェックと、 を介したキー管理イベントを提供します。これは、不正な置換を検出したり AWS CloudTrail、当事者間のキーの同期を監視したりするために使用できます。このサービスは、キーをキーブロックにのみ保存します。Payment Cryptography AWS へのインポート前とエクスポート後のキーストレージと使用は、お客様の責任となります。

PIN ベースのトランザクションまたは予期しないキー管理イベントの処理中に不一致が発生した場合は、即時調査の手順を設定する必要があります。

要件 19: サービスはキーブロックでのみキーを使用し、すべてのオペレーションに KeyUsage、KeyModeOfUse、およびその他のキー属性を適用します。これには、プライベートキーオペレーションの制限が含まれます。パブリックキーは、暗号化やデジタル署名の検証などの単一目的に使用する必要がありますが、両方に使用することはできません。本番システムとテスト/開発システムには、別々のアカウントを使用する必要があります。

要件 20: この要件については、お客様が責任を負います。

コントロールの目的 6: キーは安全な方法で管理されます。

要件 21: AWS Payment Cryptography でのキーストレージと使用は、サービスの PCI PIN 評価の一部として評価されました。キーコンポーネント関連のストレージ要件については、21-2 および 21-3 で説明されているように保存する必要があります。サービスへのインポート前およびサービスからのエクスポート後に、ポリシードキュメントでキー保護メカニズムを記述する必要があります。

要件 22: AWS Payment Cryptography の主要な侵害手順は、サービスの PCI PIN 評価の一部として評価されました。モニタリングや通知への対応 AWSなど、主要な侵害の検出と対応の手順を説明する必要があります。

要件 23: AWS Payment Cryptography は、バリアントやその他の可逆的なキー計算方法をサポートしていません。お客様が Modbus メインキーまたはそれらによって暗号化されたキーを利用することはできません。可逆的なキー計算の使用は、サービスの PCI PIN 評価の一部として評価されました。

要件 24: 内部シークレットキーとプライベートキーの破棄プラクティス AWS Payment Cryptography は、サービスの PCI PIN 評価の一部として評価されました。Python へのインポート前とエクスポート後に、キーのキー破棄手順を説明する必要があります。キーコンポーネント関連の破棄要件 (24-2.2 および 24-2.3) は引き続きお客様の責任となります。

要件 25: AWS Payment Cryptography 内のシークレットキーとプライベートキーへのアクセスは、サービスの PCI PIN 評価の一環として評価されました。 AWS Payment Cryptography へのインポート前とエクスポート後のキーのアクセスコントロールのプロセスとドキュメントが必要です。

要件 26: サービス外で使用されるキー、キーコンポーネント、または関連マテリアルへのアクセスのログ記録を記述する必要があります。アプリケーションで サービスで行うすべてのキー管理アクティビティのログは、 経由で入手できます AWS CloudTrail。

要件 27: サービス外で使用されるキー、キーコンポーネント、または関連マテリアルのバックアップ手順を説明する必要があります。

要件 28: API を使用したすべてのキー管理の手順には、キー管理アクセス許可を持つロールの使用と、キーを管理するスクリプトやその他のコードの実行の承認を含める必要があります。 AWS CloudTrail ログには、すべてのキー管理イベントが含まれます。

コントロールの目的 7: PINsとキーの処理に使用される機器は、安全な方法で管理されます。

要件 29: HSMsは、 AWS Payment Cryptography を使用して満たされます。

要件 30: アプリケーションは、POI デバイス要件のすべての物理的および論理的な保護について責任を負います。

要件 31: AWS Payment Cryptography で使用されるセキュア暗号化デバイス (SCD) の保護は、サービスの PCI PIN 評価の一環として評価されました。アプリケーションで使用される他の SCDsの保護を実証する必要があります。

要件 32: AWS Payment Cryptography で使用される SCDs の使用は、サービスの PCI PIN 評価の一部として評価されました。アプリケーションで使用される他の SCDs のアクセスコントロールと保護を実証する必要があります。

要件 33: 管理下にある PIN 処理機器の保護について説明する必要があります。

P2PE ソリューションでの AWS Payment Cryptography Decryption コンポーネントの使用

PCI P2PE ソリューションは AWS Payment Cryptography Decryption コンポーネントを使用できます。これは、PCI ウェブサイトにある「PCI Point-to-Point暗号化: セキュリティ要件とテスト手順」、セクション P2PE ソリューションとサードパーティーの使用および/または P2PE コンポーネントプロバイダー：「ソリューションプロバイダー (またはソリューションプロバイダーとしてのマーチャント) は、特定の P2PE 関数を PCI リストの P2PE コンポーネントプロバイダーにアウトソースし、検証に関する P2PE レポート (P-ROV P2PE)」で報告されています。

他の AWS のサービスやコンプライアンス標準と同様に、サービスを安全に使用し、アクセスコントロールを設定し、PCI P2PE 要件に従ってセキュリティパラメータを使用するのはお客様の責任です。で入手できる AWS Payment Cryptography P2PE Decryption Component User's Guide AWS Artifactには、 AWS Payment Cryptography を PCI P2PE ソリューションに統合するための詳細な手順と、コンプライアンスレポートに必要な年次復号コンポーネントレポートが含まれています。

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 リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「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)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「AWS Payment Cryptography と IAM の連携方法」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「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)

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。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」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、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 AWS アカウント root ユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。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。ルートユーザー認証情報を必要とするタスクについては、「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、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用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「ID プロバイダーとのフェデレーションを使用して にアクセスする必要がある AWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」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を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス 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、ID AWS またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられたときにアクセス許可を定義します。は、プリンシパル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。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、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

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のポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーと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「管理ポリシーとインラインポリシーのいずれかを選択する」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。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。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。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。

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アクセス許可を持つかを制御します。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、および Amazon VPC は AWS WAF、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、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。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アクセス許可を理解するのがさらに難しくなります。が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 アイデンティティベースのポリシー
- AWS Payment Cryptography タグに基づく認可

AWS Payment Cryptography 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、許可または拒否されたアクションとリソース、およびアクションが許可または拒否され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 JSON AWS 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対してどのような条件下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。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アクセス許可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 オペレーションを使用してパラメータを作成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。

単一のステートメントに複数のアクションを指定するには次のようにコンマで区切ります。

```
"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 JSON AWS 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対してどのような条件下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。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。リソースレベルのアクセス許可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 タグに基づく認可

AWS Payment Cryptography リソースにタグをアタッチすることも、 AWS Payment Cryptography へのリクエストでタグを渡すこともできます。タグに基づいてアクセスを管理するには、`payment-cryptography:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

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 ユーザーとロールには 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 マネジメントコンソール、 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「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

- ポリシーに関する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する機能
- 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

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内の AWS Payment Cryptography リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。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の付与を開始するには、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することで、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 ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。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、最小特権アクセス許可とも呼ばれています。IAM を使用して許可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 でのポリシーとアクセス許可 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。たとえば、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、サービスアクションが な

どの特定の を通じて使用されている場合に 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 CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。

- IAM アクセスアナライザー 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 アクセスアナライザー 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 でポリシーを検証する 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、MFA をオンにしてセキュリティを強化します。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 を使用した安全な API アクセス 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する機能

> **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と、 `ControlPlane` オペレーションと `DataPlane`オペレーションの両方を含むすべての AWS Payment Cryptography API を呼び出す機能を付与します。

JSON

```
{
      "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 `GenerateCardValidationData`と の 2 APIs で使用します`VerifyCardValidationData`。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。

JSON

```
   {
      "Version":"2012-10-17",
      "Statement": [
            {
               "Effect": "Allow",
               "Action": [
                  "payment-cryptography:VerifyCardValidationData",
                  "payment-cryptography:GenerateCardValidationData"
               ],
               "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 アカウントの IAM ユーザーに AWS Payment Cryptography キーへのアクセスを許可しますが、1 つの特定のキーに対するアクセス許可を拒否します。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。

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
```

```
            {
                "Effect": "Allow",
                "Action": [
                    "payment-cryptography:VerifyCardValidationData",
                    "payment-cryptography:GenerateCardValidationData"
                ],
                "Resource": [
                  "arn:aws:payment-cryptography:us-east-2:111122223333:key/*"
                ]
            },
            {
                "Effect": "Deny",
                "Action": [
                  "payment-cryptography:GenerateCardValidationData"
                ],
                "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 およびその他の 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な部分です。AWS には、 AWS Payment Cryptography をモニタリングし、問題が発生したときに報告し、必要に応じて自動アクションを実行するための以下のモニタリングツールが用意されています。

- Amazon CloudWatch は、 AWS リソースと AWS で実行され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 で特定の APIsしたり、 AWS Payment Cryptography のクォータに近づいている場合に通知したりできます。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。
- AWS CloudTrail は、 AWS アカウントによって、またはアカウントに代わって行われた API コールおよび関連イベントをキャプチャし、指定した Amazon S3 バケットにログファイルを配信します。が呼び出したユーザーとアカウント AWS、 が呼び出したエンドポイント、使用されたリソース (キー) 、呼び出し元のソース 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 ユーザーガイドをご参照ください。

トピック

- を使用した AWS Payment Cryptography API コールのログ記録 AWS CloudTrail

を使用した AWS Payment Cryptography API コールのログ記録 AWS CloudTrail

AWS Payment Cryptography は AWS CloudTrail、 AWS Payment Cryptography のユーザー、ロール、または サービスによって実行されたアクションを記録する AWS サービスである 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。キャプチャされたコールには、 コンソールのコールと、 API オペレーション

へのコードのコールが含まれます。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 イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールのイベント履歴で最新の管理 (コントロールプレーン) イベントを表示できます。CloudTrail で収集された情報を使用して、 AWS Payment Cryptography に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

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。

トピック

- AWS CloudTrail での Payment Cryptography 情報
- CloudTrail のコントロールプレーンイベント
- CloudTrail のデータイベント
- AWS Payment Cryptography Control Plane ログファイルエントリについて
- AWS Payment Cryptography データプレーンのログファイルエントリについて

AWS CloudTrail での Payment Cryptography 情報

CloudTrail は、 AWS アカウントの作成時にアカウントで有効になります。 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イベント履歴の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。最近のイベントは、 AWS アカウントで表示、検索、ダウンロードできます。詳細については、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 のイベントなど、 AWS アカウント内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。証跡により、ログファイルを CloudTrail で Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、すべての AWS リージョンに証跡が適用されます。証跡は、 AWS パーティション内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、それに基づいて対応するため、他の AWS サービスを構成できます。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」
- CloudTrail ログファイルの複数のリージョンからの受け取り

- 複数のアカウントから CloudTrail ログファイルを受け取る

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または AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- 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 サービスによって行われたかどうか。

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

CloudTrail のコントロールプレーンイベント

CloudTrail は、CreateKey、ImportKey、DeleteKey、ListKeys、TagResource、およびその他の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 オペレーションをログに記録します。

CloudTrail のデータイベント

データイベントは、ペイロードの暗号化やピンの変換など、リソース上またはリソース内で実行されるリソースオペレーションに関する情報を提供します。データイベントは大量のアクティビティで、CloudTrail はデフォルトではログに記録しません。CloudTrail API またはコンソールを使用して、 AWS Payment Cryptography データプレーンイベントのデータイベント APIs アクションログを有効にできます。詳細については、「AWS CloudTrail ユーザーガイド」の「データイベントをログ記録する」を参照してください。

CloudTrail では、高度なイベントセレクタを使用して、ログに記録および記録される AWS Payment Cryptography API アクティビティを決定する必要があります。 AWS Payment Cryptography データプレーンイベントをログに記録するには、リソースタイプ `AWS Payment Cryptography key`とを含める必要があります`AWS Payment Cryptography alias`。これを設定したら、`eventName` フィルタを使用して `EncryptData` イベントを追跡するなど、記録する特定のデータイベントを選択して、ロギング設定をさらに絞り込むことができます。詳細については、「AWS CloudTrail API リファレンス」の「`AdvancedEventSelector`」を参照してください。

> **Note**
>
> AWS Payment Cryptography データイベントをサブスクライブするには、高度なイベントセレクタを使用する必要があります。すべてのイベントを確実に受信できるように、キーイベントとエイリアスイベントにサブスクライブすることをお勧めします。

AWS Payment Cryptography データイベント:

- DecryptData
- EncryptData
- GenerateCardValidationData
- GenerateMac
- GeneratePinData
- ReEncryptData
- TranslatePinData
- VerifyAuthRequestCryptogram
- VerifyCardValidationData
- VerifyMac
- VerifyPinData

追加の変更がイベントデータに適用されます。詳細については、「AWS CloudTrail の料金」を参照してください。

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

「トレイル」は、指定した 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 のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典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、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API 呼び出し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。

次の例は、 AWS Payment Cryptography `CreateKey`アクションを示す CloudTrail ログエントリを示しています。

```
      {
    CloudTrailEvent: {
      tlsDetails= {
        TlsDetails: {
          cipherSuite=TLS_AES_128_GCM_SHA256,
          tlsVersion=TLSv1.3,
          clientProvidedHostHeader=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/
ControlPlane-IntegTest-68211a2a-3e9d-42b7-86ac-c682520e0410,
            invokedBy=null,
            accessKeyId=TESTXECZ5U2ZULLHHMJG,
            type=AssumedRole,
            sessionContext={
              SessionContext: {
                sessionIssuer={
                  SessionIssuer: {arn=arn:aws:iam::111122223333:role/TestAssumeRole-us-
west-2,
                  type=Role,
                  accountId=111122223333,
                  userName=TestAssumeRole-us-west-2,
                  principalId=TESTXECZ5U9M4LGF2N6Y5}
                },
                attributes={
                  SessionContextAttributes: {
                    creationDate=Sun May 21 18:58:31 UTC 2023,
                    mfaAuthenticated=false
                  }
                },
                webIdFederationData=null
```

```
              }
            },
          username=null,
          principalId=TESTXECZ5U9M4LGF2N6Y5: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={
    }
  }
}
```

次の例は、マルチリージョンキーレプリケーションを有効にする AWS Payment Cryptography を示す CloudTrail ログエントリを示しています。

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "payment-cryptography.amazonaws.com"
    },
    "eventTime": "2025-08-15T17:50:41Z",
    "eventSource": "payment-cryptography.amazonaws.com",
    "eventName": "SynchronizeMultiRegionKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "payment-cryptography.amazonaws.com",
    "userAgent": "payment-cryptography.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "55c0fcbc-5b2e-4bd2-a976-99305be6e6fc",
    "readOnly": false,
```

```
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "keyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/key-id",
        "replicationRegion": "us-east-2"
    },
    "eventCategory": "Management"
}
```

AWS Payment Cryptography データプレーンのログファイルエントリについて

データプレーンイベントは、オプションでコントロールプレーンログと同様に設定して機能できますが、通常ははるかに大きいボリュームです。 AWS Payment Cryptography データプレーンオペレーションへの一部の入力と出力の機密性を考慮すると、「*** Sensitive Data Redacted ***」というメッセージを含む特定のフィールドが見つかる場合があります。これは設定できず、機密データがログや証跡に表示されないようにすることを目的としています。

次の例は、 AWS Payment Cryptography `EncryptData`アクションを示す CloudTrail ログエントリを示しています。

```
      {
        "Records": [
            {
                "eventVersion": "1.09",
                "userIdentity": {
                    "type": "AssumedRole",
                    "principalId": "TESTXECZ5U2ZULLHHMJG:DataPlane-User",
                    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/DataPlane-
User",
                    "accountId": "111122223333",
                    "accessKeyId": "TESTXECZ5U2ZULLHHMJG",
                    "userName": "",
                    "sessionContext": {
                        "sessionIssuer": {
                            "type": "Role",
                            "principalId": "TESTXECZ5U9M4LGF2N6Y5",
                            "arn": "arn:aws:iam::111122223333:role/Admin",
                            "accountId": "111122223333",
                            "userName": "Admin"
```

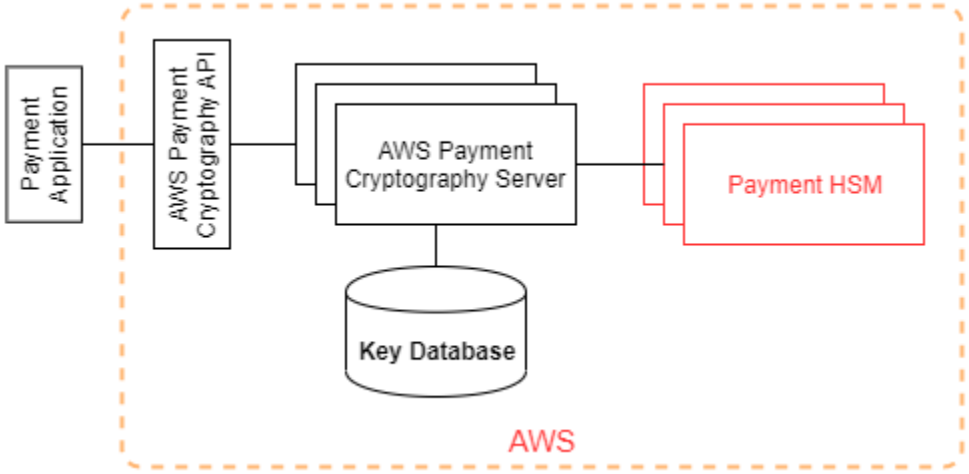
```
                        },
                        "attributes": {
                            "creationDate": "2024-07-09T14:23:05Z",
                            "mfaAuthenticated": "false"
                        }
                    }
                },
                "eventTime": "2024-07-09T14:24:02Z",
                "eventSource": "payment-cryptography.amazonaws.com",
                "eventName": "GenerateCardValidationData",
                "awsRegion": "us-east-2",
                "sourceIPAddress": "192.158.1.38",
                "userAgent": "aws-cli/2.17.6 md/awscrt#0.20.11 ua/2.0 os/macos#23.4.0
 md/arch#x86_64 lang/python#3.11.8 md/pyimpl#CPython cfg/retry-mode#standard md/
installer#exe md/prompt#off md/command#payment-cryptography-data.generate-card-
validation-data",
                "requestParameters": {
                    "key_identifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp",
                    "primary_account_number": "*** Sensitive Data Redacted ***",
                    "generation_attributes": {
                        "CardVerificationValue2": {
                            "card_expiry_date": "*** Sensitive Data Redacted ***"
                        }
                    }
                },
                "responseElements": null,
                "requestID": "f2a99da8-91e2-47a9-b9d2-1706e733991e",
                "eventID": "e4eb3785-ac6a-4589-97a1-babdd3d4dd95",
                "readOnly": true,
                "resources": [
                    {
                        "accountId": "111122223333",
                        "type": "AWS::PaymentCryptography::Key",
                        "ARN": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp"
                    }
                ],
                "eventType": "AwsApiCall",
                "managementEvent": false,
                "recipientAccountId": "111122223333",
                "eventCategory": "Data",
                "tlsDetails": {
                    "tlsVersion": "TLSv1.3",
```

```
                    "cipherSuite": "TLS_AES_128_GCM_SHA256",
                    "clientProvidedHostHeader": "dataplane.payment-cryptography.us-
east-2.amazonaws.com"
                }
            }
        ]
    }
```

暗号化の詳細

AWS Payment Cryptography は、支払いトランザクションの暗号化キーを生成および管理するためのウェブインターフェイスを提供します。 AWS Payment Cryptography は、一元的な管理と監査に使用できる標準キー管理サービスおよび支払いトランザクション暗号化とツールを提供します。このドキュメントでは、 AWS Payment Cryptography で使用できる暗号化オペレーションの詳細な説明を提供し、サービスが提供する機能の評価を支援します。

AWS Payment Cryptography には、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分散フリート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、および を介した RESTful API を含む AWS マネジメントコンソール) が含まれています。



AWS Payment Cryptography は、ウェブ向け AWS Payment Cryptography ホストと HSMs。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 プロトコル (TLS) を介して行われ、 AWS Payment Cryptography ホストで終了する必要があります。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 AWS API オペレーションで使用できる IAM の同じ認証情報とポリシーメカニズムを使用して、リクエストを認証および認可します。

AWS Payment Cryptography サーバーは、プライベートな非仮想ネットワークを介して基盤となる 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。

トピック

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 HSMs と同期できません。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。したがって、顧客の AWS Payment Cryptography キーは、顧客のアカウント内で運用されている AWS Payment Cryptography サービス以外では使用できません。
- 低レイテンシーと高スループット — AWS Payment Cryptography は、支払い暗号化キーの管理と支払いトランザクションの処理に適したレイテンシーとスループットレベルで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 — 強力な暗号化は真に予測不可能な乱数生成に依存するため、 AWS Payment Cryptography は高品質で検証済みの乱数のソースを提供します。 AWS Payment Cryptography のすべてのキー生成では、PCI モードで動作する PCI PTS HSM リストの HSM が使用されます。
- 監査 — AWS Payment Cryptography は、Amazon CloudWatch 経由で利用可能な CloudTrail 口グとサービスログに暗号化キーの使用と管理を記録します。CloudTrail 口グを使用して、キーを共有しているアカウントによるキーの使用など、暗号化キーの使用を検査できます。 AWS Payment Cryptography は、該当する PCI、カードブランド、およびリージョンの支払いセキュリティ標準

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。

- Elastic — AWS Payment Cryptography は、需要に応じてスケールアウトおよびスケールインします。HSM 容量を予測して予約する代わりに、 AWS Payment Cryptography はオンデマンドで支払い暗号化を提供します。 AWS Payment 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、パラメータ対応の標準暗号化アルゴリズムを使用して、アプリケーションがユースケースに必要なアルゴリズムを実装できるようにします。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。

注: リストされている関数の一部は、標準プロトコルとデータ構造をサポートするために内部的に使用されます。特定のアクションでサポートされているアルゴリズムについては、 API ドキュメントを参照してください。

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、完全な前方秘匿性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。

注: リストされている関数の一部は、標準プロトコルとデータ構造をサポートするために内部的に使用されます。特定のアクションでサポートされているアルゴリズムについては、 API ドキュメントを参照してください。

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じくらい強力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じくらい強力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 には、PCI PIN Security と 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があります。

キーエクスポート

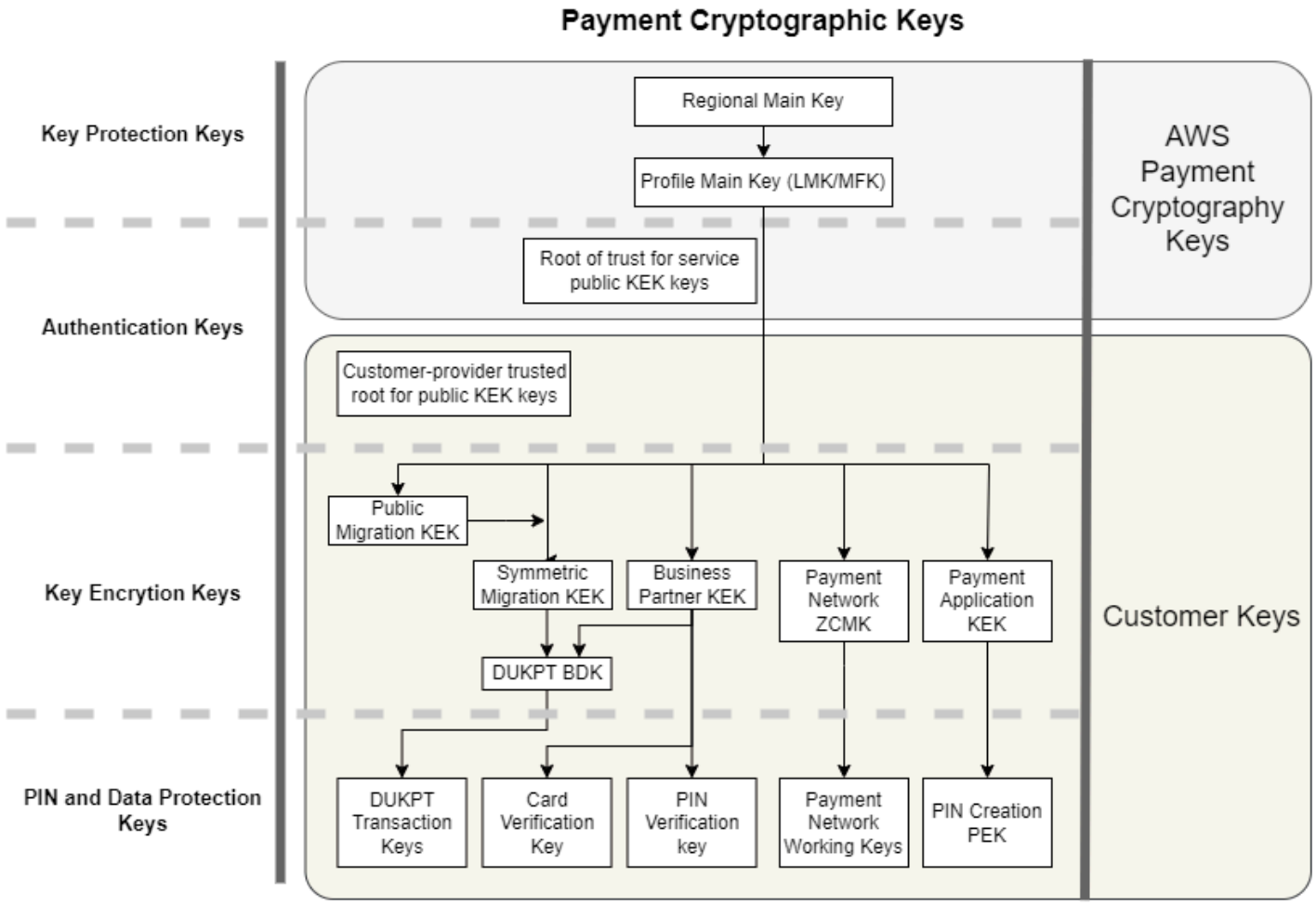
キーは、適切な KeyUsage を使用して、エクスポートするキーと同等かそれ以上の強度を持つキーによってエクスポートおよび保護できます。

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 ベース派生キー (BDK) で をサポートします。

キー階層

AWS Payment Cryptography キー階層は、キーが保護するキーと同等以上の強度を持つキーによって常に保護されるようにします。



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

キー	説明
	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) キーの信頼のルート	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。

キー	説明
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。
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。

キー	説明
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。
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。

トピック

HSM 保護

HSM の仕様とライフサイクル

AWS Payment Cryptography は、市販HSMs のフリートを使用します。HSMs は FIPS 140-2 Level 3 検証済みであり、PCI HSM v3 準拠として PCI Security Standards Council 承認済み PCI PTS デバイスリストに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。

サードパーティーの評価者は、HSM がモデル、ファームウェア、設定、ライフサイクル物理管理、変更管理、オペレーターアクセスコントロール、メインキー管理、HSM および HSMs オペレーションに関連するすべての PCI PIN および P2PE 要件を作成することを確認します。

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、または返しは提供されません。

HSM デバイスの物理的セキュリティ

サービスで使用できるのは、配信前に製造元によって AWS Payment Cryptography 認証局 (CA) によって署名されたデバイスキーを持つ HSMs のみです。 AWS Payment Cryptography は、HSM 製造元および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、PCI PIN Security Annex A および PCI P2PE Annex A への準拠を証明しています。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 の指定されたレシーバー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 演算子は、インストールされた HSM のリストを製造元のリストと照合してから、 AWS Payment Cryptography キーの受信を許可された HSM のリストにシリアル番号を追加します。

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。

HSM ごとに、個別の説明責任を伴う完全な管理過程が維持され、監視されます。

HSM の初期化

HSM は、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 Payment AWS 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

廃止作業は HSM の耐用年数終了または故障により行われます。HSM は、機能している場合は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破棄されます。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) リスト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に関連する場合、またはお客様が新しいバージョンの機能からメリットを得ることができると判断された場合に適用されます。AWS Payment Cryptography HSMs は、PCI PTS HSM リストバージョンに一致する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記録はすべて審査され、プロセスの順守状況や HSM 監視、コンソール以外のアクセス管理プロセス、オペレータートレーニングに変更が生じる可能性がないか確認されます。

一般的なキー管理

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。

[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 と一貫しています。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を除きます。

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。

- リージョン内の AWS Payment Cryptography 処理に対して検証された HSM のみが、そのリージョンのマスターキーを受信できます。
- 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、通常停止を引き起こす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 リージョン全体で複数レベルの冗長モデルを使用します。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。

HSM メインキーへのオペレーターアクセス

メインキーは、 サービスによって管理され、安全な AWS 施設で保護されている HSM にのみ存在します。メインキーを HSM からエクスポートしたり、製造元によって初期化されていない HSM に同期してサービスで使用することはできません。AWS 演算子は、サービスによって管理されていない HSM にロードされる可能性のある形式のメインキーを取得できません。

カスタマーキーの管理

AWS顧客の信頼が最優先事項です。AWS アカウントの サービスでインポートまたは作成するキーを完全に制御できます。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、長年の支払いサービスプロバイダーと同様に、HSMs を使用して顧客に代わってキーを管理するサービスプロバイダーです。サービスには、HSM の物理的および論理的なセキュリティに対する完全な責任があります。キー管理の責任は、サービスと顧客の間で共有されます。これは、顧客がサービスによって作成またはインポートされたキーに関する正確な情報を提供する必要があるためです。このキーは、サービスがキーの適切な使用と管理を強制するために使用するためです。AWS データ分離保護は、ある AWS アカウントに属するキーの侵害が別のアカウントに属するキーを侵害しないようにするために使用されます。

AWS Payment Cryptography は、サービスによって管理されるキーの HSM 物理コンプライアンスとキー管理に全責任を負います。これには、HSM メインキーの所有権と管理、および AWS Payment Cryptography によって管理されるカスタマーキーの保護が必要です。

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、キーを所有するアカウントへのプリンシパルの制限など、すべてのキー使用に対してキーポリシーを適用します。

AWS アカウントは、さまざまなデータセンターでのクラウド以外の実装に類似した、顧客またはアプリケーション間の完全な環境分離を提供します。各アカウントは、独立したアクセスコントロール、ネットワーク、コンピューティングリソース、データストレージ、データ保護と支払いトランザクション用の暗号化キー、およびすべての AWS リソースを提供します。Organizations や Control Tower などの AWS のサービスを使用すると、エンタープライズデータセンター内のケージや部屋と同様に、個別のアプリケーションアカウントのエンタープライズ管理が可能になります。

カスタマーキーへのオペレーターアクセス

サービスによって管理されるカスタマーキーは、パーティションメインキーによって保護されて保存され、所有者がキー共有用に特別に設定した所有のカスタマーアカウントまたはアカウントでのみ使用できます。AWS オペレーターは、AWS 手動オペレーターアクセスメカニズムによって管理される サービスへの手動アクセスを使用して、カスタマーキーを使用してキー管理または暗号化オペレーションをエクスポートまたは実行することはできません。

カスタマーキーの管理と使用を実装するサービスコードは、AWS PCI DSS 評価に従って評価される AWS の安全なコードプラクティスの対象となります。

バックアップとリカバリ

リージョンのサービスによって内部的に保存されたキーとキー情報は、 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 AWS。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 X9.143 形式のキーブロックに保存され、処理されます。

キーは、ImportKey が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。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、サービスによって設定された KeyUsage 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 Security および PCI P2PE では、PINs またはカードデータを暗号化するキーを共有する組織には、それらのキーの共有に使用されるキー交換キー (KEK) を含め、他の組織と同じキーを共有しないことが必要です。対称キーは、同じ組織内を含め、単一の目的で 2 つの当事者間でのみ共有すること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および適用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。別の例としては、決済ネットワークで共有されるキーに「Network」=「PayCard」というタグを付けることが挙げられます。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、ユーザーが設定できる期間が過ぎると、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、RestoreKey 以外の操作では使用できません。

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。

通信セキュリティ

外部

AWS Payment Cryptography API エンドポイントは、1.2 以降の TLS や、リクエストの認証と整合性のための署名バージョン 4 などの AWS セキュリティ基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 VPCs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モニタリングされます。

ログ記録とモニタリング

内部サービスログには以下が含まれます。

- サービスによって行われた AWS サービスコールの CloudTrail ログ
- CloudWatch ログに直接記録されたイベントと HSM からのイベントの両方の 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 の物理的なコンプライアンスについて全責任を負います。また、このサービスは安全なキーストアを提供し、PCI 標準で許可され、作成またはインポート中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
- キーのインポート
- キーをエクスポートする
- キーの削除
- キーローテーション

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実施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、キーインポートの二重制御を実施し、実証することができます。

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 は、クリアテキストのキーコンポーネントを提供または表示しません。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実施し、実証することができます。

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
Aliases	サポートされている各リージョン: 2,000	あり	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	あり	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	あり	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	あり	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	あり	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
新機能 - AS2805	AS2805 リージョンサポートをサポートするアルゴリズムとフローのサポート	2025 年 12 月 17 日
新機能 - マルチリージョンキーレプリケーション	マルチリージョンキーレプリケーションを使用すると、AWS Payment Cryptography キーを複数の にレプリケートできます AWS リージョン。	2025 年 9 月 10 日
新機能 - ECDH	このリリースでは、ECDH を使用して共有 KEK を確立し、さらなるキー交換を行うことができます。	2025 年 3 月 30 日
新しいキー交換ガイダンス	キー交換に関する新しいガイダンスが提供されています。一般的な JCB コマンドに関する情報も追加されました。	2025 年 1 月 31 日
新しいリージョンの起動	欧州 (フランクフルト) 、欧州 (アイルランド) 、アジアパシフィック (シンガポール) 、アジアパシフィック (東京) での新しいリージョンのローンチにエンドポイントを追加	2024 年 7 月 31 日
データプレーンと動的キーの CloudTrail	データプレーン (暗号化) オペレーションに CloudTrail を使用する方法についての情報	2024 年 7 月 10 日

	を例を含めて追加しました。Payment AWS Cryptography にインポートすべきではない 1 回限りの使用キーまたは制限付き使用キーをより適切にサポートするために、特定の関数に動的キーを使用する方法に関する情報を追加しました。	
更新された例	カード発行の新しい例を追加	2024 年 7 月 1 日
機能リリース	VPC エンドポイント (PrivateLink) と iCVV の例に関する情報を追加します。	2024 年 5 月 30 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。