Benutzer-Leitfaden

AWS Kryptografie im Zahlungsverkehr



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Kryptografie im Zahlungsverkehr: Benutzer-Leitfaden

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

Was ist AWS Zahlungskryptografie? 1
Konzepte 2
Branchenterminologie 4
Allgemeine Schlüsseltypen 5
Andere Begriffe 8
Zugehörige Services 14
Weitere Informationen 15
Endpunkte 15
Endpunkte der Steuerebene 15
Endpunkte der Datenebene 18
Erste Schritte 22
Voraussetzungen 22
Schritt 1: Erstellen Sie einen Schlüssel 23
Schritt 2: Generieren Sie mit dem Schlüssel einen CVV2 Wert 24
Schritt 3: Überprüfen Sie den in Schritt 2 generierten Wert 24
Schritt 4: Führen Sie einen negativen Test durch 25
Schritt 5: (Optional) Aufräumen 25
Verwalten von Schlüsseln 27
Erstellen von Schlüsseln 28
Erstellen eines 2KEY-TDES-Basisableitungsschlüssels 28
Erstellen eines 2KEY TDES-Schlüssels für CVV/CVV2 31
Einen HMAC-Schlüssel erstellen 32
Einen Schlüssel erstellen AES-256 33
Einen PIN-Verschlüsselungsschlüssel (PEK) erstellen 34
Erstellen eines asymmetrischen Schlüssels (RSA) 35
Erstellen eines PVV-Schlüssels (PIN Verification Value) 36
Erstellen eines asymmetrischen ECC-Schlüssels 37
Schlüssel auflisten 38
Aktivieren und Deaktivieren von -Schlüsseln 40
Starten Sie die Schlüsselnutzung 40
Stoppen Sie die Schlüsselnutzung 42
Replizieren von Schlüsseln 44
Vorteile der Schlüsselreplikation Multi-Region 44
Multi-Region So funktioniert die Schlüsselreplikation 44

Einschränkungen und Überlegungen ... 45
Multi-Region Schlüsselreplikation aktivieren ... 46
Deaktivieren der Schlüsselreplikation Multi-Region ... 48
Sicherheitsüberlegungen ... 49
Bewährte Methoden ... 49
Preisgestaltung ... 50
Löschen von -Schlüsseln ... 50
Über die Wartezeit ... 51
Schlüssel importieren und exportieren ... 55
Schlüssel importieren ... 57
Schlüssel exportieren ... 83
Fortgeschrittene Themen ... 107
Verwenden von Aliassen ... 126
Über Aliasse ... 127
Verwenden von Aliassen in Ihren Anwendungen ... 130
Verbundene APIs ... 131
Holen Sie sich Schlüssel ... 131
Ordnet die Öffentlichkeit key/certificate einem Schlüsselpaar zu ... 133
Tagging von Schlüsseln ... 134
Informationen zu Stichwörtern in AWS Kryptografie im Zahlungsverkehr ... 134
Wichtige Tags in der Konsole anzeigen ... 136
Verwaltung von Schlüsseltags mit API-Vorgängen ... 136
Zugriffssteuerung mit Tags ... 139
Verwendung von Tags zur Steuerung des Zugriffs auf Schlüssel ... 143
Die wichtigsten Attribute verstehen ... 147
Symmetrische Schlüssel ... 147
Asymmetrische Schlüssel ... 150
Datenoperationen ... 152
Daten verschlüsseln, entschlüsseln und erneut verschlüsseln ... 152
Daten verschlüsseln ... 153
Daten entschlüsseln ... 159
Kartendaten generieren und verifizieren ... 163
Kartendaten generieren ... 164
Überprüfen Sie die Kartendaten ... 166
PIN-Daten generieren, übersetzen und verifizieren ... 168
PIN-Daten übersetzen ... 169

Generieren Sie PIN-Daten ... 171
Überprüfen Sie die PIN-Daten ... 175
Generieren und verifizieren Sie ARQC ... 179
Kryptogramm für Authentifizierungsanfragen (ARQC) generieren ... 180
Kryptogramm für Authentifizierungsanfragen (ARQC) verifizieren ... 182
MAC generieren und verifizieren ... 186
MAC generieren ... 188
Überprüfen Sie den MAC ... 192
Schlüsseltypen für bestimmte Datenoperationen ... 194
GenerateCardData ... 195
VerifyCardData ... 196
GeneratePinData (für VISA/ABA Schemata) ... 197
GeneratePinData (für IBM3624) ... 198
VerifyPinData (für VISA/ABA Schemata) ... 199
VerifyPinData (für IBM3624) ... 200
Daten entschlüsseln ... 201
Encrypt Data ... 202
Pin-Daten übersetzen ... 204
Generate/Verify MAC ... 205
GenerateMacEmvPinChange ... 207
VerifyAuthRequestCryptogram ... 208
Import/Export Schlüssel ... 208
Unbenutzte Schlüsseltypen ... 209
Häufige Anwendungsfälle ... 210
Emittenten und Emittentenverarbeiter ... 210
Allgemeine Funktionen ... 210
Netzwerkspezifische Funktionen ... 244
Akquisitions- und Zahlungsvermittler ... 277
Dynamische Schlüssel verwenden ... 278
Regionsspezifische Funktionen ... 281
ALS 2805 ... 281
Austausch des Anfangsschlüssels (KEK) ... 283
Validierung von KEK ... 285
Erstellung und Übertragung von Arbeitstasten ... 288
Funktionstüchtige Schlüssel exportieren ... 290
Pin-Übersetzung ... 291

Mac-Generierung und Validierung ... 292
Sicherheit ... 294
Datenschutz ... 295
Schutz von Schlüsselmaterial ... 296
Datenverschlüsselung ... 296
Verschlüsselung im Ruhezustand ... 297
Verschlüsselung während der Übertragung ... 297
Richtlinie für den Datenverkehr zwischen Netzwerken ... 297
Ausfallsicherheit ... 298
Regionale Isolierung ... 298
Multi-tenant entwerfen ... 299
Sicherheit der Infrastruktur ... 300
Isolierung physischer Hosts ... 300
Verwenden Sie Amazon VPC und AWS PrivateLink ... 301
Überlegungen für AWS VPC-Endpunkte für Zahlungskryptografie ... 302
Erstellen eines VPC-Endpunkts für AWS Kryptografie im Zahlungsverkehr ... 302
Herstellen einer Verbindung mit einem VPC-Endpunkt ... 303
Steuern des Zugriffs auf einen VPC-Endpunkt ... 304
Verwenden eines VPC-Endpunkts in einer Richtlinienanweisung ... 308
Protokollieren des VPC-Endpunkts ... 311
Hybrid-Post-Quantum-TLS ... 314
Über Post-Quantum-TLS ... 316
Verwendung ... 316
Über PQC ... 316
Verwendung ... 316
Bewährte Methoden für die Gewährleistung der Sicherheit ... 316
Compliance-Validierung ... 319
Konformität des Dienstes ... 319
Einhaltung der PIN-Vorschriften ... 320
Allgemeine Themen ... 321
Umfang der Bewertung ... 323
Vorgänge zur Transaktionsverarbeitung ... 325
P2PE-Konformität ... 331
Identity and Access Management ... 332
Zielgruppe ... 332
Authentifizierung mit Identitäten ... 333

AWS-Konto Root-Benutzer ... 333
IAM-Benutzer und -Gruppen ... 333
IAM-Rollen ... 333
Verwalten des Zugriffs mit Richtlinien ... 334
Identity-based Richtlinien ... 334
Resource-based Richtlinien ... 335
Zugriffssteuerungslisten (ACLs) ... 335
Weitere Richtlinientypen ... 335
Mehrere Richtlinientypen ... 336
Wie AWS Zahlungskryptografie funktioniert mit IAM ... 336
AWS Richtlinien für Zahlungskryptografie Identity-based ... 336
Autorisierung basiert auf AWS Kryptografie-Tags für Zahlungen ... 338
Identity-based Beispiele für politische Maßnahmen ... 339
Best Practices für Richtlinien ... 339
Verwenden der Konsole ... 341
Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer ... 341
Möglichkeit, auf alle Aspekte von zuzugreifen AWS Kryptografie für Zahlungen ... 342
Möglichkeit, APIs mit bestimmten Schlüsseln aufzurufen ... 343
Fähigkeit, eine Ressource gezielt abzulehnen ... 344
Resource-based Richtlinien ... 345
Überlegungen ... 346
Verwaltung ressourcenbasierter Richtlinien ... 347
Resource-based Beispiele für Richtlinien ... 348
Multi-party Zulassung ... 350
-Übersicht ... 350
Geschützte Operationen ... 351
Voraussetzungen ... 351
MPA aktivieren und deaktivieren ... 352
Erste Schritte ... 353
Beispiel: Importieren Sie ein Stammzertifikat mit aktiviertem MPA ... 353
AWS CloudTrail Protokollierung von MPA-Ereignissen ... 355
Überprüfen des Anforderungsstatus und Behandlung von Fehlern ... 357
Fehlerbehebung ... 359
Überwachen ... 360
CloudTrail protokolliert ... 361
AWS Informationen zur Zahlungskryptografie in CloudTrail ... 361

Ereignisse auf Kontrollebene in CloudTrail ... 362
Datenereignisse in CloudTrail ... 362
Grundlegendes zu den Protokolldateieinträgen der AWS Payment Cryptography Control Plane ... 364
Grundlegendes AWS zur Zahlungskryptografie: Einträge in Protokolldateien auf Datenebene ... 367
Kryptografische Details ... 370
Designziele ... 371
Grundlagen ... 372
Kryptografische Primitive ... 373
Entropie und Zufallszahlengenerierung ... 373
Symmetrische Tastenoperationen ... 373
Asymmetrische Schlüsseloperationen ... 373
Speicher für Schlüssel ... 374
Schlüsselimport mit symmetrischen Schlüsseln ... 374
Schlüsselimport mit asymmetrischen Schlüsseln ... 374
Export von Schlüsseln ... 375
DUKPT (Derived Unique Key Per Transaction) -Protokoll ... 375
Schlüsselhierarchie ... 375
Interne Operationen ... 380
HSM-Schutz ... 380
Allgemeine Schlüsselverwaltung ... 383
Verwaltung von Kundenschlüsseln ... 387
Sicherheit der Kommunikation ... 390
Protokollierung und Überwachung ... 390
Geschäftsbetrieb für Kunden ... 391
Generieren von Schlüsseln ... 391
Importieren von Schlüsseln ... 392
Exportieren von Schlüsseln ... 393
Löschen von -Schlüsseln ... 393
Rotieren von -Schlüsseln ... 393
Kontingente ... 394
Dokumentverlauf ... 396
... cccxcix

Was ist AWS Zahlungskryptografie?

AWS Payment Cryptography ist ein verwalteter AWS Dienst, der Zugriff auf kryptografische Funktionen und Schlüsselverwaltung bietet, die bei der Zahlungsabwicklung gemäß den PCI-Standards (Payment Card Industry) verwendet werden, ohne dass Sie spezielle HSM-Instanzen für Zahlungen erwerben müssen. AWS Payment Cryptography bietet Kunden, die Zahlungsfunktionen ausführen, wie Acquirern, Zahlungsvermittlern, Netzwerken, Switches, Prozessoren und Banken, die Möglichkeit, ihre kryptografischen Zahlungsvorgänge näher an Anwendungen in der Cloud zu verlagern und die Abhängigkeit von zusätzlichen Rechenzentren oder Colocation-Einrichtungen mit dedizierten Zahlungen zu minimieren HSMs.

Der Service ist so konzipiert, dass er die geltenden Branchenregeln wie PCI PIN, PCI P2PE und PCI DSS erfüllt, und der Service nutzt Hardware, die nach PCI PTS HSM V3 und FIPS 140-2 Level 3 zertifiziert ist. Er ist so konzipiert, dass er niedrige Latenzzeiten und ein hohes Maß an Verfügbarkeit und Belastbarkeit unterstützt. AWS Die Zahlungskryptografie ist vollständig elastisch und macht viele betriebliche Anforderungen vor Ort überflüssig HSMs, wie z. B. die Bereitstellung von Hardware, die sichere Verwaltung von Schlüsselmaterial und die Aufbewahrung von Notfall-Backups in sicheren Einrichtungen. AWS Die Zahlungskryptografie bietet Ihnen auch die Möglichkeit, Schlüssel elektronisch mit Ihren Partnern zu teilen, sodass Sie keine Klartext-Komponenten in paper austauschen müssen.

Sie können die AWS Payment Cryptography Control Plane API verwenden, um Schlüssel zu erstellen und zu verwalten.

Sie können die AWS Payment Cryptography Data Plane API verwenden, um Verschlüsselungsschlüssel für die Zahlungsabwicklung und die damit verbundenen kryptografischen Vorgänge zu verwenden.

AWS Die Zahlungskryptografie bietet wichtige Funktionen, mit denen Sie Ihre Schlüssel verwalten können:

- Erstellen und verwalten Sie symmetrische und asymmetrische Schlüssel für die AWS Zahlungskryptografie, einschließlich TDES-, AES- und RSA-Schlüsseln, und geben Sie deren Verwendungszweck an, z. B. für die CVV-Generierung oder die DUKPT-Schlüsselableitung.
- Speichern Sie Ihre Schlüssel zur AWS Zahlungskryptografie automatisch und sicher, geschützt durch Hardware-Sicherheitsmodule ()HSMs, und sorgen Sie gleichzeitig für eine Trennung der Schlüssel zwischen den einzelnen Anwendungsfällen.

- Erstellen, löschen, listen und aktualisieren Sie Aliase. Dabei handelt es sich um „benutzerfreundliche Namen“, mit denen Sie auf Ihre AWS Payment Cryptography Keys zugreifen oder den Zugriff darauf kontrollieren können.
- Kennzeichnen Sie Ihre Kryptografie-Schlüssel für AWS Zahlungen zur Identifizierung, Gruppierung, Automatisierung, Zugriffskontrolle und Kostenverfolgung.
- Importieren und exportieren Sie symmetrische Schlüssel zwischen AWS Payment Cryptography und Ihrem HSM (oder Drittanbietern) mithilfe von Key Encryption Keys (KEK) gemäß TR-31 (Interoperable Secure Key Exchange Key Block Specification).
- Importieren und exportieren Sie symmetrische Schlüsselverschlüsselungsschlüssel (KEK) zwischen AWS Payment Cryptography und anderen Systemen, die asymmetrische Schlüsselpaare verwenden, und verwenden Sie anschließend elektronische Mittel wie TR-34 (Methode zur Verteilung symmetrischer Schlüssel mithilfe asymmetrischer Techniken).

Sie können Ihre kryptografischen AWS Zahlungsschlüssel für kryptografische Operationen verwenden, z. B. für:

- Verschlüsseln, entschlüsseln und verschlüsseln Sie Daten mit symmetrischen oder asymmetrischen Schlüsseln für die Zahlungskryptografie. AWS
- Übersetzen Sie vertrauliche Daten (wie Karteninhaber-PINS) auf sichere Weise zwischen Verschlüsselungsschlüsseln, ohne dass der Klartext gemäß den PCI-PIN-Regeln offengelegt wird.
- Generieren oder validieren Sie Karteninhaberdaten wie CVV oder ARQC. CVV2
- Generieren und validieren Sie Karteninhaber-Pins.
- Generieren oder validieren Sie MAC-Signaturen.

Konzepte

Lernen Sie die grundlegenden Begriffe und Konzepte der AWS Zahlungskryptografie kennen und erfahren Sie, wie Sie sie zum Schutz Ihrer Daten verwenden können.

Alias

> Ein benutzerfreundlicher Name, der mit einem AWS Zahlungskryptografie-Schlüssel verknüpft ist. Der Alias kann in vielen API-Vorgängen für AWS Payment Cryptography synonym mit dem Schlüssel-ARN verwendet werden. Mithilfe von Aliasen können Schlüssel rotiert oder auf andere Weise geändert werden, ohne dass sich dies auf Ihren Anwendungscode auswirkt. Der

Aliasname besteht aus einer Zeichenfolge mit bis zu 256 Zeichen. Es identifiziert eindeutig einen zugehörigen AWS Zahlungskryptografie-Schlüssel innerhalb eines Kontos und einer Region. In der AWS Zahlungskryptografie beginnen Aliasnamen immer mit. `alias/`

Das Format eines Aliasnamens lautet wie folgt:

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

Zum Beispiel:

```
alias/sampleAlias2
```

Schüssel-ARN

Der Schlüssel-ARN ist der Amazon-Ressourcenname (ARN) eines Schlüsseleintrags in AWS Payment Cryptography. Es handelt sich um eine eindeutige, vollqualifizierte Kennung für den AWS Payment Cryptography Key. Ein Schlüssel-ARN umfasst eine AWS-Konto, eine Region und eine zufällig generierte ID. Der ARN steht nicht im Zusammenhang mit dem Schlüsselmaterial oder leitet sich davon ab. Da sie bei Erstellungs- oder Importvorgängen automatisch zugewiesen werden, sind diese Werte nicht idempotent. Wenn Sie denselben Schlüssel mehrmals importieren, erhalten Sie mehrere Schlüssel ARNs mit jeweils eigenem Lebenszyklus.

Das Format eines Schlüssel-ARN lautet wie folgt:

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

Im Folgenden finden Sie ein Beispiel für einen Schlüssel-ARN:

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

Schlüssel-ID

Eine Schlüssel-ID ist ein Verweis auf einen Schlüssel, und einer (oder mehrere) von ihnen sind typische Eingaben für AWS kryptografische Zahlungsvorgänge. Gültige Schlüsselkennungen können entweder ein Schlüssel oder ein Schlüsselalias sein.

AWS Kryptografie-Schlüssel für Zahlungen

AWS Kryptografie-Schlüssel (Schlüssel) für Zahlungen werden für alle kryptografischen Funktionen verwendet. Schlüssel werden entweder direkt von Ihnen mit dem Befehl create

key generiert oder dem System hinzugefügt, indem Sie den Schlüsselimport aufrufen. Der Ursprung eines Schlüssels kann anhand des Attributs bestimmt werden KeyOrigin. AWS Die Zahlungskryptografie unterstützt auch abgeleitete Schlüssel oder Zwischenschlüssel, die bei kryptografischen Vorgängen verwendet werden, wie sie beispielsweise von DUKPT verwendet werden.

Diese Schlüssel haben sowohl unveränderliche als auch veränderbare Attribute, die bei der Erstellung definiert wurden. Attribute wie Algorithmus, Länge und Verwendung werden bei der Erstellung definiert und können nicht geändert werden. Andere, wie z. B. das Gültigkeitsdatum oder das Ablaufdatum, können geändert werden. Eine vollständige Liste der Schlüsselattribute für AWS Zahlungskryptografie finden Sie in der API-Referenz für AWS Zahlungskryptografie.

AWS Für Schlüssel zur Zahlungskryptografie gibt es Schlüsseltypen, die hauptsächlich durch ANSI X9 TR 31 definiert sind. Sie beschränken ihre Verwendung auf den vorgesehenen Zweck, wie in PCI-PIN v3.1 Requirement 19 festgelegt.

Attribute werden mithilfe von Schlüsselblöcken an Schlüssel gebunden, wenn sie gespeichert, mit anderen Konten geteilt oder exportiert werden, wie in PCI-PIN v3.1 Anforderung 18-3 spezifiziert.

Schlüssel werden auf der AWS Payment Cryptography Platform anhand eines eindeutigen Werts identifiziert, der als Amazon Resource Name (ARN) als Schlüssel bezeichnet wird.

> **Note**
>
> Der Schlüssel ARN wird generiert, wenn ein Schlüssel zum ersten Mal erstellt oder in den AWS Payment Cryptography Service importiert wird. Wenn also dasselbe Schlüsselmaterial mithilfe der Schlüsselimportfunktion mehrmals hinzugefügt wird, befindet sich dasselbe Schlüsselmaterial unter mehreren Schlüsseln, die ARNS jedoch jeweils einen anderen Schlüssellebenszyklus haben.

Branchenterminologie

Themen

- Allgemeine Schlüsseltypen
- Andere Begriffe

Allgemeine Schlüsseltypen

AWS Kryptografie-Schlüssel für Zahlungen

Ein kryptografischer Schlüssel für AWS Zahlungen ist in einem einzigen Schlüssel enthalten. AWS-Region Er besteht aus wichtigen Metadaten und Materialien, die im AWS Payment Cryptography Service gespeichert sind. Ein Schlüssel kann als TR-31 Schlüsselblock aus einer externen Quelle importiert oder vom AWS Payment Cryptography Service generiert werden.

AWK

Ein Acquirer Working Key (AWK) ist ein Schlüssel, der typischerweise für den Datenaustausch zwischen einem acquirer/acquirer Prozessor und einem Netzwerk (wie Visa oder Mastercard) verwendet wird. In der Vergangenheit nutzte AWK 3DES zur Verschlüsselung und wurde als TR31_P0_PIN_ENCRYPTION_KEY dargestellt.

BDK

Ein Basisableitungsschlüssel (Base Derivation Key, BDK) ist ein funktionierender Schlüssel, der häufig als Teil des PCI-PIN- und PCI P2PE-DUKPT-Prozesses verwendet wird. Er wird als TR31_B0_BASE_DERIVATION_KEY bezeichnet.

CMK

Ein Kartenhauptschlüssel (CMK) ist ein oder mehrere kartenspezifische Schlüssel, die in der Regel von einem Issuer Master Key, PAN und PSN abgeleitet werden und sind in der Regel 3DES-Schlüssel. Diese Schlüssel werden während der Personalisierung auf dem EMV-Chip gespeichert. Beispiele für CMKs sind AC-, SMI- und SMC-Schlüssel.

CMK-AC

Ein Anwendungskryptogrammschlüssel (AC) wird als Teil von EMV-Transaktionen verwendet, um das Transaktionskryptogramm zu generieren. Dabei handelt es sich um eine Art Kartenhauptschlüssel.

CMK-SMI

Ein SMI-Schlüssel (Secure Messaging Integrity) wird als Teil von EMV verwendet, um die Integrität von Payloads zu überprüfen, die mithilfe von MAC an die Karte gesendet werden, z. B. von Pin-Aktualisierungsskripten. Es handelt sich um eine Art Hauptschlüssel für Karten.

CMK-SMC

Ein SMC-Schlüssel (Secure Messaging Confidentiality) wird als Teil von EMV verwendet, um an die Karte gesendete Daten zu verschlüsseln, z. B. PIN-Updates. Es handelt sich um eine Art Hauptschlüssel für Karten.

CVK

Ein Kartenverifikationsschlüssel (CVK) ist ein Schlüssel, der zur Generierung von CVV-, CVV2- und ähnlichen Werten unter Verwendung eines definierten Algorithmus sowie zur Validierung einer Eingabe verwendet wird. Er wird als TR31_C0_CARD_VERIFICATION_KEY bezeichnet.

IMK

Ein Issuer Master Key (IMK) ist ein Masterschlüssel, der im Rahmen der Personalisierung von EMV-Chipkarten verwendet wird. In der Regel gibt es 3 IMKs — jeweils einen für AC-Schlüssel (Kryptogramm), SMI- (Skript-Masterschlüssel für integrity/signature) und SMC-Schlüssel (Skript-Masterschlüssel für). confidentiality/encryption

IK

Ein Anfangsschlüssel (IK) ist der erste Schlüssel, der im DUKPT-Prozess verwendet wird. Er wird vom Base Derivation Key (BDK) abgeleitet. Mit diesem Schlüssel werden keine Transaktionen verarbeitet, aber er wird verwendet, um future Schlüssel abzuleiten, die für Transaktionen verwendet werden. Die Ableitungsmethode für die Erstellung eines IK wurde in definiert. X9.24-1:2017 Wenn ein TDES-BDK verwendet wird, gilt der geltende Standard und IK X9.24-1:2009 wird durch den Initial Pin Encryption Key (IPEK) ersetzt.

IPEK

Ein initialer PIN-Verschlüsselungsschlüssel (IPEK) ist der erste Schlüssel, der im DUKPT-Prozess verwendet wird. Er wird vom Base Derivation Key (BDK) abgeleitet. Mit diesem Schlüssel werden keine Transaktionen verarbeitet, aber er wird verwendet, um future Schlüssel abzuleiten, die für Transaktionen verwendet werden. IPEK ist eine Fehlbezeichnung, da dieser Schlüssel auch zur Ableitung von Datenverschlüsselung und Mac-Schlüsseln verwendet werden kann. Die Ableitungsmethode für die Erstellung eines IPEK wurde in definiert. X9.24-1:2009 Wenn ein AES-BDK verwendet wird, X9.24-1:2017 gilt der geltende Standard und IPEK wird durch Initial Key (IK) ersetzt.

IWK

Ein Issuer Working Key (IWK) ist ein Schlüssel, der typischerweise für den Datenaustausch zwischen einem issuer/issuer Prozessor und einem Netzwerk (wie Visa oder Mastercard)

verwendet wird. In der Vergangenheit nutzte IWK 3DES zur Verschlüsselung und wurde als TR31_P0_PIN_ENCRYPTION_KEY dargestellt.

KBPK

Ein Key Block Protection Key (KBPK) ist eine Art symmetrischer Schlüssel, der zum Schutz von Schlüsselblöcken und damit wrap/encrypt anderen Schlüsseln verwendet wird. Ein KBPK ähnelt einem KEK, aber ein KEK schützt das Schlüsselmaterial direkt, wohingegen in TR-31 und ähnlichen Schemata das KBPK den Arbeitsschlüssel nur indirekt schützt. Bei der Verwendung ist TR31_K1_KEY_BLOCK_PROTECTION_KEY der richtige Schlüsseltyp *TR-31*, obwohl TR31_K0_KEY_ENCRYPTION_KEY für historische Zwecke synonym unterstützt wird.

KEK

Ein Key Encryption Key (KEK) ist ein Schlüssel, mit dem andere Schlüssel entweder für die Übertragung oder Speicherung verschlüsselt werden. Schlüssel, die zum Schutz anderer Schlüssel bestimmt sind, haben laut Standard in der Regel den Wert TR31_K0_KEY_ENCRYPTION_KEY. KeyUsage TR-31

SPITZE

Ein PIN-Verschlüsselungsschlüssel (PEK) ist eine Art Arbeitsschlüssel, der zur Verschlüsselung von PINs entweder für die Speicherung oder Übertragung zwischen zwei Parteien verwendet wird. IWK und AWK sind zwei Beispiele für spezifische Anwendungen von PIN-Verschlüsselungsschlüsseln. Diese Schlüssel werden als TR31_P0_PIN_ENCRYPTION_KEY dargestellt.

PGK

PGK (Pin Generation Key) ist ein anderer Name für einen Pin-Bestätigungsschlüssel. Es wird nicht wirklich zur Generierung von Pins verwendet (bei denen es sich standardmäßig um kryptografische Zufallszahlen handelt), sondern stattdessen zur Generierung von Bestätigungswerten wie PVV.

PRK

Der Schlüssel für die primäre Region ist die autoritative Replikationsquelle für einen bestimmten Zahlungskryptografie-Schlüssel, für den die Replikation aktiviert wurde. PRK ist ein Verweis auf eine Schlüsselrolle bei der Zahlungskryptografie als Quelle in einer Multi-Region Konfiguration für die Schlüsselreplikation. Wenn die Replikation für einen Payment Cryptography Key aktiviert ist, wird sie für diese spezielle Schlüsselreplikationskonfiguration als PRK bezeichnet.

PVK

Ein PIN-Bestätigungsschlüssel (PVK) ist eine Art Arbeitsschlüssel, der zur Generierung von PIN-Bestätigungswerten wie PVV verwendet wird. Die beiden gängigsten Typen sind TR31_V1_IBM3624_PIN_VERIFICATION_KEY, der zum Generieren von IBM3624-Offsetwerten verwendet wird, und TR31_V2_VISA_PIN_VERIFICATION_KEY, der für Bestätigungswerte verwendet wird. Visa/ABA Dies kann auch als Pin-Generierungsschlüssel bezeichnet werden.

RRK

Bei den Regionsschlüsseln für Replikate handelt es sich um das replizierte Schlüsselmaterial und die Metadaten, die sicher aus dem PRK in ein konfiguriertes Replikat kopiert wurden. AWS-Region Ein RRK ist eine schreibgeschützte Replik eines Zahlungskryptografie-Schlüssels. RRK ist eine Referenz für die Rolle, die ein bestimmter Schlüssel in einer Multi-Region Konfiguration für die Schlüsselreplikation spielt. Alle wichtigen Änderungen an den Metadaten, einschließlich der Replikationseinstellungen, müssen auf das PRK angewendet werden.

Andere Begriffe

ARQC

Das Authorization Request Cryptogram (ARQC) ist ein Kryptogramm, das während der Transaktion mit einer EMV-Standard-Chipkarte (oder einer gleichwertigen kontaktlosen Implementierung) generiert wird. In der Regel wird ein ARQC durch eine Chipkarte generiert und zur Überprüfung bei der Transaktion an einen Emittenten oder dessen Vertreter weitergeleitet.

CVV

Ein Kartenprüfwert ist ein statischer geheimer Wert, der traditionell in einen Magnetstreifen eingebettet wurde und zur Überprüfung der Echtheit einer Transaktion verwendet wurde. Der Algorithmus wird auch für andere Zwecke wie iCVV, CAVV, CVV2 verwendet. Für andere Anwendungsfälle darf er nicht auf diese Weise eingebettet werden.

CVV2

Ein Kartenprüfwert 2 ist ein statischer geheimer Wert, der traditionell auf der Vorder- (oder Rückseite) einer Zahlungskarte aufgedruckt wurde und zur Überprüfung der Echtheit von Zahlungen ohne Karte verwendet wird (z. B. am Telefon oder online). Er verwendet denselben Algorithmus wie CVV, der Servicecode ist jedoch auf 000 gesetzt.

iCVV

iCVV ist ein CVV2-like Wert, der jedoch in die Track2-äquivalenten Daten auf einer EMV-Karte (Chip) eingebettet ist. Dieser Wert wird anhand des Servicecodes 999 berechnet und unterscheidet sich von dem, CVV1/CVV2 um zu verhindern, dass gestohlene Informationen verwendet werden, um neue Zahlungsinformationen eines anderen Typs zu erstellen. Wenn beispielsweise Chip-Transaktionsdaten abgerufen wurden, ist es nicht möglich, diese Daten zur Generierung eines Magnetstreifens (CVV1) oder für Online-Käufe (CVV2) zu verwenden.

Es verwendet einen Schlüssel ???

DUMPT

Derived Unique Key Per Transaction (DUKPT) ist ein Schlüsselverwaltungsstandard, der in der Regel verwendet wird, um die Verwendung von einmalig verwendbaren Verschlüsselungsschlüsseln auf physischen Geräten zu definieren. POS/POI In der Vergangenheit nutzte DUKPT 3DES für die Verschlüsselung. Der Industriestandard für DUKPT ist in ANSI definiert. X9.24-3-2017

ECC

ECC (Elliptic Curve Cryptography) ist ein Kryptografiesystem mit öffentlichen Schlüsseln, das die Mathematik elliptischer Kurven zur Erstellung von Verschlüsselungsschlüsseln verwendet. ECC bietet das gleiche Sicherheitsniveau wie herkömmliche Methoden wie RSA, jedoch mit viel kürzeren Schlüssellängen, wodurch gleichwertige Sicherheit auf effizientere Weise gewährleistet wird. Dies ist besonders relevant für Anwendungsfälle, in denen RSA keine praktische Lösung ist (RSA-Schlüssellänge > 4096 Bit). AWS Die Zahlungskryptografie unterstützt von NIST definierte Kurven zur Verwendung in ECDH-Operationen.

ECDH

ECDH (Elliptic Curve Diffie-Hellman) ist ein wichtiges Vertragsprotokoll, das es zwei Parteien ermöglicht, ein gemeinsames Geheimnis (z. B. ein KEK oder ein PEK) festzulegen. In ECDH haben Partei A und B jeweils ihre eigenen öffentlich-privaten Schlüsselpaare und tauschen untereinander öffentliche Schlüssel (in Form von Zertifikaten für AWS Zahlungskryptografie) sowie Metadaten zur Schlüsselableitung (Ableitungsmethode, Hashtyp und gemeinsame Informationen) aus. Beide Parteien multiplizieren ihren privaten Schlüssel mit dem öffentlichen Schlüssel des jeweils anderen, und aufgrund der Eigenschaften elliptischer Kurven können beide Parteien den resultierenden Schlüssel ableiten (generieren).

EMV

EMV (ursprünglich Europay, Mastercard, Visa) ist ein technisches Gremium, das mit Interessenträgern im Zahlungsverkehr zusammenarbeitet, um interoperable Zahlungsstandards und -technologien zu entwickeln. Ein Beispiel für Standards sind chip/contactless Karten und die Zahlungsterminals, mit denen sie interagieren, einschließlich der verwendeten Kryptografie. Die EMV-Schlüsselableitung bezieht sich auf Verfahren zur Generierung eindeutiger Schlüssel für jede Zahlungskarte auf der Grundlage eines anfänglichen Schlüsselsatzes wie einem `IMK`

HSM

Ein Hardware-Sicherheitsmodul (HSM) ist ein physisches Gerät, das kryptografische Operationen (z. B. Verschlüsselung, Entschlüsselung und digitale Signaturen) sowie die zugrunde liegenden Schlüssel, die für diese Operationen verwendet werden, schützt.

KCAAS

A Key Custodian As A Service (KCAAS) bietet eine Vielzahl von Dienstleistungen rund um das Schlüsselmanagement an. Bei Zahlungsschlüsseln können sie in der Regel papierbasierte Schlüsselkomponenten in elektronische Formulare umwandeln, die von AWS Payment Cryptography unterstützt werden, oder elektronisch geschützte Schlüssel in papierbasierte Komponenten umwandeln, die möglicherweise von bestimmten Anbietern benötigt werden. Sie können auch wichtige Treuhanddienste für Schlüssel anbieten, deren Verlust sich nachteilig auf Ihren laufenden Betrieb auswirken würde. KCAAS-Anbieter sind in der Lage, Kunden dabei zu unterstützen, den betrieblichen Aufwand für die Verwaltung von Schlüsselmaterial außerhalb eines sicheren Dienstes wie AWS Zahlungskryptografie auf eine Weise zu übernehmen, die den PCI-DSS-, PCI-PIN- und PCI P2PE-Standards entspricht. AWS Payment Cryptography bietet Physischer Schlüsselaustausch als integrierte KCAAS-Funktion die Umwandlung von papierbasierten Schlüsselkomponenten in ein elektronisches Format.

KCV

Key Check Value (KCV) bezieht sich auf eine Vielzahl von Prüfsummenmethoden, die hauptsächlich dazu verwendet werden, Schlüssel miteinander zu vergleichen, ohne Zugriff auf das eigentliche Schlüsselmaterial zu haben. KCV wurde auch für die Integritätsprüfung verwendet (insbesondere beim Austausch von Schlüsseln), obwohl diese Rolle jetzt Teil von Schlüsselblockformaten wie z. `TR-31` Bei TDES-Schlüsseln wird der KCV berechnet, indem 8 Byte, jedes mit dem Wert Null, verschlüsselt werden, wobei der zu prüfende Schlüssel und die 3 höchstwertigen Byte des verschlüsselten Ergebnisses beibehalten werden. Bei AES-Schlüsseln wird der KCV mithilfe eines CMAC-Algorithmus berechnet, bei dem die Eingabedaten aus 16

Byte Null bestehen und die 3 Byte der höchsten Ordnung des verschlüsselten Ergebnisses beibehalten werden. Bei HMAC-Schlüsseln wird der KCV anhand des Hashs berechnet, der bei der Schlüsselerstellung für eine Nachricht mit der Länge Null ausgewählt wurde, wobei die 3 Byte ganz links verwendet werden.

KDH

Ein Key Distribution Host (KDH) ist ein Gerät oder System, das Schlüssel in einem Schlüsselaustauschprozess sendet, wie z. TR-34 Beim Senden von Schlüsseln aus AWS Payment Cryptography wird dies als KDH betrachtet.

KIF

Eine Key Injection Facility (KIF) ist eine sichere Einrichtung zur Initialisierung von Zahlungsterminals, einschließlich des Ladens dieser mit Verschlüsselungsschlüsseln.

KRD

Ein Schlüsselempfangsgerät (KRD) ist ein Gerät, das Schlüssel in einem Schlüsselaustauschprozess empfängt, wie z. TR-34 Beim Senden von Schlüsseln an AWS Payment Cryptography wird es als KRD betrachtet.

KSN

Eine Schlüsselseriennummer (KSN) ist ein Wert, der als Eingabe für DUKPT verwendet wird, um eindeutige Verschlüsselungsschlüssel pro Transaktion encryption/decryption zu erstellen. Die KSN besteht in der Regel aus einer BDK-Kennung, einer halbeindeutigen Terminal-ID sowie einem Transaktionszähler, der bei jedem auf einem bestimmten Zahlungsterminal verarbeiteten Übergang inkrementiert wird. Gemäß X9.24 TDES besteht das 10-Byte-KSN typischerweise aus 24 Bit für die Key Set ID, 19 Bit für die Terminal ID und 21 Bit für den Transaktionszähler, obwohl die Grenze zwischen Key Set ID und Terminal ID keinen Einfluss auf die Funktion der Zahlungskryptografie hat. AWS Bei AES besteht das 12-Byte-KSN typischerweise aus 32 Bit für die BDK-ID, 32 Bit für den Derivation Identifier (ID) und 32 Bit für den Transaktionszähler.

mPoC

mPOC (Mobile Point of Sale on Commercial Hardware) ist ein PCI-Standard, der die Sicherheitsanforderungen für Lösungen berücksichtigt, die es Händlern ermöglichen, PINs von Karteninhabern oder kontaktlose Zahlungen mit einem Smartphone oder anderen handelsüblichen Mobilgeräten (COTS) zu akzeptieren.

PAN

Eine primäre Kontonummer (PAN) ist eine eindeutige Kennung für ein Konto, z. B. eine Kredit- oder Debitkarte. In der Regel 13-19 Ziffern lang. Die ersten 6-8 Ziffern identifizieren das Netzwerk und die ausstellende Bank.

PIN-Block

Ein Datenblock, der während der Verarbeitung oder Übertragung eine PIN sowie andere Datenelemente enthält. PIN-Blockformate standardisieren den Inhalt des PIN-Blocks und die Art und Weise, wie er verarbeitet werden kann, um die PIN abzurufen. Die meisten PIN-Blocks bestehen aus der PIN und der PIN-Länge und enthalten häufig einen Teil oder die gesamte PAN. AWS Die Zahlungskryptografie unterstützt die ISO 9564-1-Formate 0, 1, 3 und 4. Format 4 ist für AES-Schlüssel erforderlich. Bei der Überprüfung oder Übersetzung von PINs muss der PIN-Block der eingehenden oder ausgehenden Daten angegeben werden.

POI

Point of Interaction (POI), auch häufig anonym bei Point of Sale (POS) verwendet, ist das Hardwaregerät, mit dem der Karteninhaber interagiert, um seine Zahlungsdaten vorzuzeigen. Ein Beispiel für einen POI ist das physische Terminal an einem Händlerstandort. Eine Liste der zertifizierten PCI-PTS-POI-Terminals finden Sie auf der PCI-Website.

PSN

Die PAN-Sequenznummer (PSN) ist ein numerischer Wert, der zur Unterscheidung mehrerer Karten verwendet wird, die mit derselben PAN ausgegeben wurden.

Öffentlicher Schlüssel

Bei der Verwendung asymmetrischer Chiffren (RSA, ECC) ist der öffentliche Schlüssel die öffentliche Komponente eines öffentlich-privaten key pair. Der öffentliche Schlüssel kann freigegeben und an Entitäten verteilt werden, die Daten für den Besitzer des öffentlich-privaten Schlüsselpaars verschlüsseln müssen. Bei digitalen Signaturvorgängen wird der öffentliche Schlüssel verwendet, um die Signatur zu überprüfen.

Privater Schlüssel

Bei der Verwendung asymmetrischer Chiffren (RSA, ECC) ist der private Schlüssel die private Komponente eines öffentlich-privaten key pair. Mit dem privaten Schlüssel werden dann Daten entschlüsselt oder digitale Signaturen erstellt. Ähnlich wie bei symmetrischen Schlüsseln für die AWS Zahlungskryptografie werden private Schlüssel von HSMs sicher erstellt. Sie werden nur in

den flüchtigen Speicher des HSM entschlüsselt und nur für die Zeit, die für die Bearbeitung Ihrer kryptografischen Anfrage benötigt wird.

PVV

Ein PIN-Bestätigungswert (PVV) ist eine Art kryptografischer Ausgabe, mit der eine PIN verifiziert werden kann, ohne dass die tatsächliche PIN gespeichert wird. Obwohl es sich um einen Oberbegriff handelt, bezieht sich PVV im Zusammenhang mit AWS Zahlungskryptografie auf die Visa- oder ABA-PVV-Methode. Bei dieser PVV handelt es sich um eine vierstellige Zahl, deren Eingaben die Kartennummer, die Pan-Sequenznummer, die Pan selbst und ein PIN-Bestätigungsschlüssel sind. Während der Validierungsphase erstellt AWS Payment Cryptography die PVV intern anhand der Transaktionsdaten neu und vergleicht sie erneut mit dem Wert, der vom Payment Cryptography-Kunden gespeichert wurde. AWS Auf diese Weise ähnelt es konzeptionell einem kryptografischen Hash oder MAC.

RSA Wrap/Unwrap

RSA Wrap verwendet einen asymmetrischen Schlüssel, um einen symmetrischen Schlüssel (z. B. einen TDES-Schlüssel) für die Übertragung an ein anderes System zu umschließen. Nur das System mit dem passenden privaten Schlüssel kann die Nutzdaten entschlüsseln und den symmetrischen Schlüssel laden. Umgekehrt entschlüsselt RSA Unwrap einen mit RSA verschlüsselten Schlüssel sicher und lädt den Schlüssel dann in die Zahlungskryptografie. AWS RSA Wrap ist eine Low-Level-Methode für den Austausch von Schlüsseln, bei der Schlüssel nicht im Schlüsselblockformat übertragen werden und auch keine Payload-Signierung durch die sendende Partei verwendet wird. Alternative Kontrollen sollten in Betracht gezogen werden, um sicherzustellen, dass Providence und Schlüsselattribute nicht verändert sind.

TR-34 verwendet RSA auch intern, ist jedoch ein eigenständiges Format und daher nicht interoperabel.

Service-Code

Ein Servicecode ist ein dreistelliger Wert, der auf dem Magnetstreifen und dem Chip einer Zahlungskarte kodiert ist und Kartenattribute definiert, z. B. wo die Karte verwendet werden kann und ob eine zusätzliche Autorisierung erforderlich ist. Der Servicecode wird als Eingabe bei der Berechnung von Bestätigungswerten verwendet. Verschiedene Servicecodewerte führen zu unterschiedlichen Typen von Bestätigungswerten wie CVV (unter Verwendung des Servicecodes aus dem Magnetstreifen der Karte), iCVV (Servicecode 999 für Chipkarten) und CVV2 (Servicecode 000).

TR-31

TR-31 (formal definiert als ANSI X9 TR 31) ist ein Schlüsselblockformat, das vom American National Standards Institute (ANSI) definiert wurde, um die Definition von Schlüsselattributen in derselben Datenstruktur wie die Schlüsseldaten selbst zu unterstützen. Das TR-31 Schlüsselblockformat definiert eine Reihe von Schlüsselattributen, die an den Schlüssel gebunden sind, sodass sie zusammengehalten werden. AWS Bei der Zahlungskryptografie werden wann immer möglich TR-31 standardisierte Begriffe verwendet, um eine korrekte Trennung der Schlüssel und den Hauptzweck zu gewährleisten. TR-31 wurde durch ANSI ersetzt. X9.143-2022

TR-34

TR-34 ist eine Implementierung von ANSI X9.24-2 , in der ein Protokoll zur sicheren Verteilung symmetrischer Schlüssel (wie 3DES und AES) mithilfe asymmetrischer Techniken (wie RSA) beschrieben wurde. AWS Die Zahlungskryptografie verwendet TR-34 Methoden, um einen sicheren Import und Export von Schlüsseln zu ermöglichen.

X9.143

X9.143 ist ein Schlüsselblockformat, das vom American National Standards Institute (ANSI) definiert wurde, um die Sicherung eines Schlüssels und von Schlüsselattributen in derselben Datenstruktur zu unterstützen. Das Schlüsselblockformat definiert eine Reihe von Schlüsselattributen, die an den Schlüssel gebunden sind, sodass sie zusammengehalten werden. AWS Bei der Zahlungskryptografie werden wann immer möglich X9.143 standardisierte Begriffe verwendet, um eine korrekte Trennung der Schlüssel und den Hauptzweck zu gewährleisten. X9.143 ersetzt den früheren TR-31Vorschlag, obwohl sie in den meisten Fällen rückwärts- und vorwärtskompatibel sind und die Begriffe häufig synonym verwendet werden.

Zugehörige Services

AWS Key Management Service

AWS Der Key Management Service (AWS KMS) ist ein verwalteter Dienst, mit dem Sie die kryptografischen Schlüssel, die zum Schutz Ihrer Daten verwendet werden, auf einfache Weise erstellen und kontrollieren können. AWS KMS verwendet Hardware-Sicherheitsmodule (HSMs), um Ihre AWS KMS-Schlüssel zu schützen und zu validieren.

AWS CloudHSM

AWS CloudHSM bietet Kunden spezielle HSM-Instanzen für allgemeine Zwecke in der AWS Cloud. AWS CloudHSM kann eine Vielzahl von kryptografischen Funktionen bereitstellen, z. B.

das Erstellen von Schlüsseln, das Signieren von Daten oder das Verschlüsseln und Entschlüsseln von Daten.

Weitere Informationen

- Informationen zu den in der Zahlungskryptografie verwendeten Begriffen und Konzepten finden Sie unter AWS Konzepte der AWS Zahlungskryptografie.
- Informationen zur AWS Payment Cryptography Control Plane API finden Sie unter AWS Payment Cryptography Control Plane API-Referenz.
- Informationen zur AWS Payment Cryptography Data Plane API finden Sie unter AWS Payment Cryptography Data Plane API-Referenz.
- Ausführliche technische Informationen darüber, wie die AWS Zahlungskryptografie Kryptografie verwendet und die Schlüssel für die AWS Zahlungskryptografie schützt, finden Sie unter Kryptografische Details.

Endpunkte für AWS Payment Cryptography

Um programmgesteuert eine Verbindung herzustellen AWS Payment Cryptography, verwenden Sie einen Endpunkt, die URL des Einstiegspunkts für den Dienst. Die AWS SDKs und die Befehlszeilentools verwenden automatisch den Standardendpunkt für den Service in einem auf der Region AWS-Region basierenden Kontext einer Anfrage, sodass es in der Regel nicht erforderlich ist, diese Werte explizit festzulegen. Bei Bedarf können Sie einen anderen Endpunkt für Ihre API-Anfragen angeben.

Endpunkte der Steuerebene

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
USA Ost (Ohio)	us-east-2	controlplane.payment-cryptography.us-east-2.amazonaws.com controlplane.payment-cryptography.us-east-2.api.aws	HTTPS HTTPS	

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
USA Ost (Nord-Virginia)	us-east-1	controlplane.payment-cryptography.us-east-1.amazonaws.com controlplane.payment-cryptography.us-east-1.api.aws	HTTPS HTTPS	
USA West (Oregon)	us-west-2	controlplane.payment-cryptography.us-west-2.amazonaws.com controlplane.payment-cryptography.us-west-2.api.aws	HTTPS HTTPS	
Afrika (Kapstadt)	af-south-1	controlplane.payment-cryptography.af-south-1.amazonaws.com controlplane.payment-cryptography.af-south-1.api.aws	HTTPS HTTPS	
Asien-Pazifik (Hyderabad)	ap-south-2	controlplane.payment-cryptography.ap-south-2.amazonaws.com controlplane.payment-cryptography.ap-south-2.api.aws	HTTPS HTTPS	
Asien-Pazifik (Mumbai)	ap-south-1	controlplane.payment-cryptography.ap-south-1.amazonaws.com controlplane.payment-cryptography.ap-south-1.api.aws	HTTPS HTTPS	
Asien-Pazifik (Osaka)	ap-northeast-3	controlplane.payment-cryptography.ap-northeast-3.amazonaws.com controlplane.payment-cryptography.ap-northeast-3.api.aws	HTTPS HTTPS	

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
Asien-Pazifik (Singapur)	ap-southeast-1	controlplane.payment-cryptography.ap-southeast-1.amazonaws.com controlplane.payment-cryptography.ap-southeast-1.api.aws	HTTPS HTTPS	
Asien-Pazifik (Sydney)	ap-southeast-2	controlplane.payment-cryptography.ap-southeast-2.amazonaws.com controlplane.payment-cryptography.ap-southeast-2.api.aws	HTTPS HTTPS	
Asien-Pazifik (Tokio)	ap-northeast-1	controlplane.payment-cryptography.ap-northeast-1.amazonaws.com controlplane.payment-cryptography.ap-northeast-1.api.aws	HTTPS HTTPS	
Kanada (Zentral)	ca-central-1	controlplane.payment-cryptography.ca-central-1.amazonaws.com controlplane.payment-cryptography.ca-central-1.api.aws	HTTPS HTTPS	
Europa (Frankfurt)	eu-central-1	controlplane.payment-cryptography.eu-central-1.amazonaws.com controlplane.payment-cryptography.eu-central-1.api.aws	HTTPS HTTPS	
Europa (Irland)	eu-west-1	controlplane.payment-cryptography.eu-west-1.amazonaws.com controlplane.payment-cryptography.eu-west-1.api.aws	HTTPS HTTPS	

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
Europa (London)	eu-west-2	controlplane.payment-cryptography.eu-west-2.amazonaws.com controlplane.payment-cryptography.eu-west-2.api.aws	HTTPS HTTPS	
Europa (Paris)	eu-west-3	controlplane.payment-cryptography.eu-west-3.amazonaws.com controlplane.payment-cryptography.eu-west-3.api.aws	HTTPS HTTPS	
Südamerika (São Paulo)	sa-east-1	controlplane.payment-cryptography.sa-east-1.amazonaws.com controlplane.payment-cryptography.sa-east-1.api.aws	HTTPS HTTPS	

Endpunkte der Datenebene

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
USA Ost (Ohio)	us-east-2	dataplane.payment-cryptography.us-east-2.amazonaws.com dataplane.payment-cryptography.us-east-2.api.aws	HTTPS HTTPS	
USA Ost (Nord-Virginia)	us-east-1	dataplane.payment-cryptography.us-east-1.amazonaws.com	HTTPS HTTPS	

Name der Region	Region	Endpunkt	Protocol (Protokoll)
		dataplane.payment-cryptography.us-east-1.api.aws	
USA West (Oregon)	us-west-2	dataplane.payment-cryptography.us-west-2.amazonaws.com dataplane.payment-cryptography.us-west-2.api.aws	HTTPS HTTPS
Afrika (Kapstadt)	af-south-1	dataplane.payment-cryptography.af-south-1.amazonaws.com dataplane.payment-cryptography.af-south-1.api.aws	HTTPS HTTPS
Asien-Pazifik (Hyderabad)	ap-south-2	dataplane.payment-cryptography.ap-south-2.amazonaws.com dataplane.payment-cryptography.ap-south-2.api.aws	HTTPS HTTPS
Asien-Pazifik (Mumbai)	ap-south-1	dataplane.payment-cryptography.ap-south-1.amazonaws.com dataplane.payment-cryptography.ap-south-1.api.aws	HTTPS HTTPS
Asien-Pazifik (Osaka)	ap-northeast-3	dataplane.payment-cryptography.ap-northeast-3.amazonaws.com dataplane.payment-cryptography.ap-northeast-3.api.aws	HTTPS HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
Asien-Pazifik (Singapur)	ap-southeast-1	dataplane.payment-cryptography.ap-southeast-1.amazonaws.com dataplane.payment-cryptography.ap-southeast-1.api.aws	HTTPS HTTPS	
Asien-Pazifik (Sydney)	ap-southeast-2	dataplane.payment-cryptography.ap-southeast-2.amazonaws.com dataplane.payment-cryptography.ap-southeast-2.api.aws	HTTPS HTTPS	
Asien-Pazifik (Tokio)	ap-northeast-1	dataplane.payment-cryptography.ap-northeast-1.amazonaws.com dataplane.payment-cryptography.ap-northeast-1.api.aws	HTTPS HTTPS	
Kanada (Zentral)	ca-central-1	dataplane.payment-cryptography.ca-central-1.amazonaws.com dataplane.payment-cryptography.ca-central-1.api.aws	HTTPS HTTPS	
Europa (Frankfurt)	eu-central-1	dataplane.payment-cryptography.eu-central-1.amazonaws.com dataplane.payment-cryptography.eu-central-1.api.aws	HTTPS HTTPS	
Europa (Irland)	eu-west-1	dataplane.payment-cryptography.eu-west-1.amazonaws.com dataplane.payment-cryptography.eu-west-1.api.aws	HTTPS HTTPS	

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
Europa (London)	eu-west-2	dataplane.payment-cryptography.eu-west-2.amazonaws.com dataplane.payment-cryptography.eu-west-2.api.aws	HTTPS HTTPS	
Europa (Paris)	eu-west-3	dataplane.payment-cryptography.eu-west-3.amazonaws.com dataplane.payment-cryptography.eu-west-3.api.aws	HTTPS HTTPS	
Südamerika (São Paulo)	sa-east-1	dataplane.payment-cryptography.sa-east-1.amazonaws.com dataplane.payment-cryptography.sa-east-1.api.aws	HTTPS HTTPS	

Erste Schritte mit AWS Payment Cryptography

Um mit der AWS Zahlungskryptografie zu beginnen, sollten Sie zunächst Schlüssel erstellen und diese dann für verschiedene kryptografische Operationen verwenden. Das folgende Tutorial bietet einen einfachen Anwendungsfall für die Generierung eines Schlüssels, der für generating/verifying CVV2 Werte verwendet werden soll. Um andere Beispiele auszuprobieren und Bereitstellungsmuster innerhalb von AWS zu untersuchen, probieren Sie bitte den folgenden AWS Payment Cryptography Workshop aus oder schauen Sie sich unser Beispielprojekt an unter GitHub

Dieses Tutorial führt Sie durch die Erstellung eines einzelnen Schlüssels und die Durchführung kryptografischer Operationen mit diesem Schlüssel. Anschließend löschen Sie den Schlüssel, wenn Sie ihn nicht mehr benötigen, wodurch der Schlüssellebenszyklus abgeschlossen ist.

> **Warning**
>
> In den Beispielen in diesem Benutzerhandbuch können Beispielwerte verwendet werden. Es wird dringend empfohlen, in einer Produktionsumgebung keine Beispielwerte wie Seriennummern von Schlüsseln zu verwenden.

Topics

- Voraussetzungen
- Schritt 1: Erstellen Sie einen Schlüssel
- Schritt 2: Generieren Sie mit dem Schlüssel einen CVV2 Wert
- Schritt 3: Überprüfen Sie den in Schritt 2 generierten Wert
- Schritt 4: Führen Sie einen negativen Test durch
- Schritt 5: (Optional) Aufräumen

Voraussetzungen

Bevor Sie beginnen, stellen Sie Folgendes sicher:

- Sie haben die Erlaubnis, auf den Service zuzugreifen. Weitere Informationen finden Sie unter IAM-Richtlinien.

- Sie haben das AWS CLIinstalliert. Sie können auch AWS SDKsoder verwenden AWS APIs, um auf AWS Payment Cryptography zuzugreifen, aber die Anweisungen in diesem Tutorial verwenden die AWS CLI.

Schritt 1: Erstellen Sie einen Schlüssel

Der erste Schritt besteht darin, einen Schlüssel zu erstellen. Für dieses Tutorial erstellen Sie einen CVK-3DES-Schlüssel mit doppelter Länge (2KEY TDES) zum Generieren und Überprüfen von CVV/-Werten. CVV2

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
    "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"
    }
}
```

Notieren Sie sich den Wert, der den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east - 2:111122223333:key/tqv5yij6wtxx64pi. Das benötigen Sie im nächsten Schritt.

Schritt 2: Generieren Sie mit dem Schlüssel einen CVV2 Wert

In diesem Schritt generieren Sie mithilfe des Schlüssels aus Schritt 1 einen Wert CVV2 für ein bestimmtes PAN Ablaufdatum.

```
$  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"
}
```

Notieren Sie sich diecardDataValue, in diesem Fall die 3-stellige Zahl 144. Das brauchst du im nächsten Schritt.

Schritt 3: Überprüfen Sie den in Schritt 2 generierten Wert

In diesem Beispiel validieren Sie den CVV2 aus Schritt 2 stammenden Schlüssel mit dem Schlüssel, den Sie in Schritt 1 erstellt haben.

Führen Sie den folgenden Befehl aus, um das zu validieren CVV2.

```
$  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"
}
```

Der Dienst gibt eine HTTP-Antwort von 200 zurück, um anzuzeigen, dass er die überprüft hat CVV2.

Schritt 4: Führen Sie einen negativen Test durch

In diesem Schritt erstellen Sie einen negativen Test, bei dem der nicht korrekt CVV2 ist und nicht validiert wird. Sie versuchen, CVV2 mit dem Schlüssel, den Sie in Schritt 1 erstellt haben, einen falschen zu validieren. Dies ist ein erwarteter Vorgang, z. B. wenn ein Karteninhaber an der CVV2 Kasse einen falschen Wert eingegeben hat.

```
$  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.
```

Der Dienst gibt eine HTTP-Antwort von 400 mit der Meldung „Überprüfung der Kartenvalidierungsdaten fehlgeschlagen“ und dem Grund INVALID_VALIDATION_DATA zurück.

Schritt 5: (Optional) Aufräumen

Jetzt können Sie den Schlüssel löschen, den Sie in Schritt 1 erstellt haben. Um die Anzahl der nicht wiederherstellbaren Änderungen zu minimieren, beträgt die Standardlöschfrist für Schlüssel sieben Tage.

```
$ 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"
    }
}
```


Notieren Sie sich zwei Felder in der Ausgabe. Die `deletePendingTimestamp` ist standardmäßig auf sieben Tage in der future festgelegt. Der KeyState ist auf eingestellt. `DELETE_PENDING` Sie können diesen Löschvorgang jederzeit vor dem geplanten Löschtermin stornieren, indem Sie anrufen. `restore-key`

Verwalten von Schlüsseln

Um mit der AWS Zahlungskryptografie zu beginnen, erstellen Sie einen Schlüssel für die AWS Zahlungskryptografie.

In diesem Abschnitt wird erklärt, wie Sie verschiedene Schlüsseltypen für AWS Zahlungskryptografie während ihres gesamten Lebenszyklus erstellen und verwalten. Sie erfahren, wie Sie Schlüssel erstellen, anzeigen und bearbeiten und wie Sie Schlüssel taggen, Schlüsselaliase erstellen und Schlüssel aktivieren oder deaktivieren.

Ein AWS Payment Cryptography Key ist eine regionale Ressource. Wenn Sie beabsichtigen, einen bestimmten Schlüssel in mehreren Schlüsseln zu verwenden AWS-Regionen, können Sie die Multi-Region Schlüsselreplikation aktivieren, bei der Schlüsselmaterial und Metadaten, die AWS-Regionen Sie angeben, auf sichere Weise innerhalb derselben AWS Partition und desselben Kontos kopiert werden. Der Quellschlüssel bei der Multi-Region Schlüsselreplikation wird als Primärregionsschlüssel (PRK) bezeichnet. Dieser ist nach wie vor die maßgebliche Quelle für alle wichtigen Verwaltungsaktivitäten. Der replizierte Schlüssel wird als Replikatregionsschlüssel (RRK) bezeichnet. Dabei handelt es sich um ein schreibgeschütztes Replikat des PRK. Sie sollten erwägen, Multi-Region Schlüssel zusammen mit Ihren Schlüsseln zu verwenden, um die Designziele in Bezug auf Verfügbarkeit, Notfallwiederherstellung und niedrige Latenz zu erreichen.

Themen

- Erstellen von Schlüsseln
- Schlüssel auflisten
- Aktivieren und Deaktivieren von -Schlüsseln
- Replikation AWS Schlüssel zur Verschlüsselung von Zahlungen
- Löschen von -Schlüsseln
- Schlüssel importieren und exportieren
- Verwenden von Aliassen
- Holen Sie sich Schlüssel
- Tagging von Schlüsseln
- Grundlegendes zu Schlüsselattributen für AWS Kryptografie-Schlüssel für Zahlungen

Erstellen von Schlüsseln

Sie können mithilfe des CreateKey API-Vorgangs Schlüssel für die AWS Zahlungskryptografie erstellen. Wenn Sie einen Schlüssel erstellen, geben Sie Attribute wie den Schlüsselalgorithmus, die Schlüsselverwendung, zulässige Operationen an und ob er exportierbar ist. Sie können diese Eigenschaften nicht mehr ändern, nachdem Sie den AWS Payment Cryptography-Schlüssel erstellt haben.

> **Note**
>
> Wenn die Multi-Region Schlüsselreplikation für Sie aktiviert ist AWS-Konto und Sie einen Payment Cryptography Key erstellen, wird dieser Schlüssel automatisch zu einem Primary Region Key (PRK). PRK wird repliziert, auch wenn Sie den `--replication-regions` Parameter nicht im Befehl angeben. CreateKey Weitere Informationen finden Sie unter Multi-Region So funktioniert die Schlüsselreplikation.

Beispiele

- Erstellen eines 2KEY-TDES-Basisableitungsschlüssels
- Erstellen eines 2KEY TDES-Schlüssels für CVV/CVV2
- Einen HMAC-Schlüssel erstellen
- Einen Schlüssel erstellen AES-256
- Einen PIN-Verschlüsselungsschlüssel (PEK) erstellen
- Erstellen eines asymmetrischen Schlüssels (RSA)
- Erstellen eines PVV-Schlüssels (PIN Verification Value)
- Erstellen eines asymmetrischen ECC-Schlüssels

Erstellen eines 2KEY-TDES-Basisableitungsschlüssels

> **Note**
>
> Obwohl 3KEY TDES für KeyUsage = TR31_B0_BASE_DERIVATION_KEY unterstützt wird, unterstützen derzeit keine Operationen diesen Schlüsseltyp. Der X9.24 ANSI-Standard (DUKPT) definiert die Unterstützung nur für TDES_2KEY- und AES-Schlüsseltypen. ???

Example

Mit diesem Befehl wird ein 2KEY-TDES-Ableitungsschlüssel für DUKPT erstellt, der von den Regionen USA West (Oregon) nach USA Ost (Ohio) repliziert wird. Die Antwort umfasst die Anforderungsparameter, einen Amazon-Ressourcennamen (ARN) für nachfolgende Aufrufe und einen Key Check Value (KCV).

```
$ aws payment-cryptography create-key --exportable --key-attributes \
     "KeyUsage=TR31_B0_BASE_DERIVATION_KEY, \
     KeyClass=SYMMETRIC_KEY,KeyAlgorithm=TDES_2KEY, \
     KeyModesOfUse={DeriveKey=true}" \
     --replication-regions us-east-2 --region us-west-2
```

Beispielausgabe:

```
{
    "Key": {
        "CreateTimestamp": "2022-10-26T16:04:11.642000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_2KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": true,
                "Encrypt": false,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": false,
                "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"
```

```
}
```

Erstellen eines 2KEY TDES-Schlüssels für CVV/CVV2

Example

Dieser Befehl erstellt einen 2KEY-TDES-Schlüssel zum Generieren und Überprüfen von Werten. CVV/CVV2 Die Antwort umfasst die Anforderungsparameter, einen Amazon-Ressourcennamen (ARN) für nachfolgende Aufrufe und einen Key Check Value (KCV).

```
$ 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}'
```

Beispielausgabe:

```
{
    "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"
    }
}
```

Einen HMAC-Schlüssel erstellen

Example

HMAC-Schlüssel werden zum Generieren oder Überprüfen von Hash-Nachrichten-Authentifizierungscodes (HMAC) verwendet. Bei HMAC-Schlüsseln wird der Hashtyp zum Zeitpunkt der Schlüsselerstellung zugewiesen (z. B. HMAC_SHA224 und HMAC_SHA512) und kann nicht geändert werden.

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

Beispielausgabe:

```
{
 "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"
 }
}
```

Einen Schlüssel erstellen AES-256

Example

Dieser Befehl erstellt einen AES-256 symmetrischen Schlüssel für die Verschlüsselung und Entschlüsselung von Daten. AES-Schlüssel bieten eine starke Verschlüsselung für sensible Daten und werden häufig bei der Zahlungsabwicklung zur Verschlüsselung von Karteninhaberdaten und anderen vertraulichen Informationen verwendet. TDES wird jedoch häufiger für Anwendungsfälle von Emittenten wie EMV verwendet.

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

Beispielausgabe:

```
{
    "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"
    }
}
```

Einen PIN-Verschlüsselungsschlüssel (PEK) erstellen

Example

Mit diesem Befehl wird ein 3KEY-TDES-Schlüssel zum Verschlüsseln von PIN-Werten erstellt, obwohl PIN-Schlüssel je nach Interoperabilitätsanforderungen auch AES-Schlüssel sein können. Sie können diesen Schlüssel verwenden, um PINs sicher zu speichern oder PINs während der Überprüfung, z. B. bei einer Transaktion, zu entschlüsseln. Die Antwort umfasst die Anforderungsparameter, einen ARN für nachfolgende Aufrufe und einen KCV.

```
$ 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}'
```

Beispielausgabe:

```
{
    "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"
    }
}
```

Erstellen eines asymmetrischen Schlüssels (RSA)

Example

Dieser Befehl generiert ein neues asymmetrisches RSA 2048-Bit-Schlüsselpaar. Es erstellt einen neuen privaten Schlüssel und den zugehörigen öffentlichen Schlüssel. Sie können den öffentlichen Schlüssel mithilfe der PublicCertificate Get-API abrufen.

```
$ 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}'
```

Beispielausgabe:

```
{
    "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"
    }
}
```

Erstellen eines PVV-Schlüssels (PIN Verification Value)

Example

Mit diesem Befehl wird ein 3KEY-TDES-Schlüssel zum Generieren von PVV-Werten erstellt. Sie können diesen Schlüssel verwenden, um einen PVV zu generieren, der mit einem später berechneten PVV verglichen werden kann. Die Antwort enthält die Anforderungsparameter, einen ARN für nachfolgende Aufrufe und einen KCV.

```
$ 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}'
```

Beispielausgabe:

```
{
    "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"
    }
}
```

Erstellen eines asymmetrischen ECC-Schlüssels

Example

Dieser Befehl generiert ein ECC-Schlüsselpaar zum Erstellen einer ECDH-Schlüsselvereinbarung (Elliptic Curve Diffie-Hellman) zwischen zwei Parteien. Bei ECDH generiert jede Partei ihr eigenes ECC-Schlüsselpaar mit dem Schlüsselzweck K3 und der Verwendungsart X, und sie tauschen öffentliche Schlüssel aus. Beide Parteien verwenden dann ihren privaten Schlüssel und den empfangenen öffentlichen Schlüssel, um einen gemeinsamen abgeleiteten Schlüssel zu erstellen.

Um das Prinzip der einmaligen Verwendung kryptografischer Schlüssel bei Zahlungen beizubehalten, empfehlen wir, ECC-Schlüsselpaare nicht für mehrere Zwecke wiederzuverwenden, z. B. für die Ableitung und Signierung von ECDH-Schlüsseln.

```
$ 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}'
```

Beispielausgabe:


```
{
    "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"
    }
}
```

Schlüssel auflisten

Verwenden Sie den ListKeys Vorgang, um eine Liste der Schlüssel abzurufen, auf die Sie in Ihrem Konto und Ihrer Region zugreifen können.

Example

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

Beispielausgabe:

```
{
    "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"
        }
    ]
}
```

Aktivieren und Deaktivieren von -Schlüsseln

Sie können AWS Payment Cryptography Keys deaktivieren und wieder aktivieren. Wenn Sie einen Schlüssel erstellen, ist er standardmäßig aktiviert. Wenn Sie einen Schlüssel deaktivieren, kann er für keine kryptografischen Operationen verwendet werden, bis Sie ihn erneut aktivieren. Start/stop Verwendungsbefehle werden sofort wirksam. Es wird daher empfohlen, die Verwendung zu überprüfen, bevor Sie eine solche Änderung vornehmen. Mithilfe des optionalen `timestamp` Parameters können Sie auch festlegen, dass eine Änderung (Start oder Stopp der Nutzung) in Zukunft wirksam wird.

Da die Deaktivierung eines AWS Payment Cryptography-Schlüssels temporär ist und leicht rückgängig gemacht werden kann, ist sie eine sicherere Alternative zum Löschen eines AWS Payment Cryptography-Schlüssels. Diese Aktion ist zerstörerisch und irreversibel. Wenn Sie erwägen, einen AWS Payment Cryptography-Schlüssel zu löschen, deaktivieren Sie ihn zuerst und stellen Sie sicher, dass Sie den Schlüssel in Zukunft nicht mehr zum Verschlüsseln oder Entschlüsseln von Daten verwenden müssen.

Themen

- Starten Sie die Schlüsselnutzung
- Stoppen Sie die Schlüsselnutzung

Starten Sie die Schlüsselnutzung

Die Schlüsselverwendung muss aktiviert sein, um einen Schlüssel für kryptografische Operationen verwenden zu können. Wenn ein Schlüssel nicht aktiviert ist, können Sie diesen Vorgang verwenden, um ihn nutzbar zu machen. Das Feld `UsageStartTimestamp` gibt an, wann der Schlüssel became/will aktiv wird. Dies ist bei aktivierten Tokens in der Vergangenheit der Fall und bei ausstehender Aktivierung in der Zukunft.

Example

In diesem Beispiel wird angefordert, dass ein Schlüssel für die Schlüsselnutzung aktiviert wird. Die Antwort enthält die wichtigsten Informationen und das Aktivierungs-Flag wurde auf „Wahr“ geändert. Dies wird sich auch im Antwortobjekt List-Keys widerspiegeln.

```
$ 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"
  }
}
```

Stoppen Sie die Schlüsselnutzung

Wenn Sie nicht mehr vorhaben, einen Schlüssel zu verwenden, können Sie die Verwendung des Schlüssels beenden, um weitere kryptografische Operationen zu verhindern. Dieser Vorgang ist nicht dauerhaft, Sie können ihn also rückgängig machen, indem Sie den Startschlüssel verwenden. Sie können einen Schlüssel auch so einstellen, dass er in Zukunft deaktiviert wird. Das Feld `UsageStopTimestamp` gibt an, wann der Schlüssel deaktiviert became/will wird.

Example

In diesem Beispiel wird es aufgefordert, die Verwendung von Schlüsseln in Zukunft zu beenden. Nach der Ausführung kann dieser Schlüssel nicht für kryptografische Operationen verwendet werden, es sei denn, er wird über die Verwendung des Startschlüssels erneut aktiviert. Die Antwort enthält die Schlüsselinformationen und das Aktivierungs-Flag wurde auf „Falsch“ geändert. Dies wird sich auch im Antwortobjekt List-Keys widerspiegeln.

```
$ 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"
  }
}
```

Replikation AWS Schlüssel zur Verschlüsselung von Zahlungen

AWS Payment Cryptography unterstützt die Multi-Region Schlüsselreplikation, sodass Sie Schlüsselmaterial und Metadaten von einem beliebigen AWS Payment Cryptography Key sicher an eine oder mehrere AWS-Regionen innerhalb derselben AWS Partition und desselben Kontos verteilen können.

Der Quellschlüssel wird als Primary Region Key (PRK) bezeichnet und bleibt die maßgebliche Quelle für alle Schlüsselverwaltungsaktivitäten, während sowohl der PRK- als auch der Replica-Regionsschlüssel (RRK) jeweils für kryptografische Operationen verwendet werden können. AWS-Regionen

Vorteile der Schlüsselreplikation Multi-Region

Im Folgenden werden einige Vorteile der Multi-Region Schlüsselreplikation beschrieben.

- Einfachere Einrichtung für Anwendungen mit hoher Verfügbarkeit — AWS Payment Cryptography übernimmt die Schlüsselverteilung für Sie, sodass Sie einen Schlüssel mehrfach verwenden können, AWS-Regionen ohne entkoppelte Kopien eines bestimmten Schlüssels erstellen zu müssen.
- Schlüssel mit hoher Verfügbarkeit und niedriger Latenz — Mit der Multi-Region Schlüsselreplikation können Sie mehrfach auf Ihre Schlüssel zugreifen, AWS-Regionen sodass sie hochverfügbar sind, was zu einer geringeren Latenz führt.
- Haltbarkeit des Schlüsselmaterials — Replikatregionsschlüssel sind vollständige Schlüsselrepliken und können unabhängig von ihrem Primärregionsschlüssel für kryptografische Operationen verwendet werden. Ein RRK bietet ein robustes Replikat für den Fall eines katastrophalen Datenverlusts eines PRK.

Multi-Region So funktioniert die Schlüsselreplikation

Wenn die Multi-Region Schlüsselreplikation aktiviert ist, verwendet der AWS Payment Cryptography Service sichere Mechanismen zur Schlüsselverteilung, um Schlüsselmaterial und Metadaten in das von AWS-Regionen Ihnen angegebene Replikat zu kopieren. Änderungen an Schlüsselmetadaten für die primäre Region, wie Schlüsselattribute, Status und Aktivierung, werden automatisch auf die Replikatregionsschlüssel repliziert.

Einschränkungen und Überlegungen

Im Folgenden sind einige Multi-Region wichtige Einschränkungen und Überlegungen zur Replikation aufgeführt.

- Die Schlüssel für die Zahlungskryptografie müssen auf exportierbar gesetzt sein, um die Multi-Region Schlüsselreplikation verwenden zu können.
- Sie müssen diese Funktion entweder für einen AWS-Region oder bestimmte Payment Cryptography Keys aktivieren.
  - Wenn diese Funktion für einen aktiviert ist AWS-Region, werden alle AWS Payment Cryptography Schlüssel, die nach der Aktivierung erstellt wurden, auf den angegebenen Wert repliziert. AWS-Region In dieser Region erstellte Schlüssel werden zu Primärregionsschlüsseln. Bestehende Schlüssel in dieser Region werden nicht automatisch repliziert. Sie können die Multi-Region Schlüsselreplikation für vorhandene Schlüssel innerhalb und AWS-Region auf Schlüsselebene aktivieren.
  - Jeder AWS-Region kann eigene Einstellungen für die Multi-Region Schlüsselreplikation haben.
  - Die Multi-Region Replikationseinstellungen eines Schlüssels haben Vorrang vor den Einstellungen für die AWS-Region Multi-Region Schlüsselreplikation.
- Ein Replikatregionsschlüssel kann nicht so konfiguriert werden, dass er auf einen anderen repliziert wird. AWS-Regionen
- Multi-Region Die Schlüsselreplikation ist für symmetrische Zahlungskryptografie-Schlüssel wie Triple Data Encryption Standard (3DES), Advanced Encryption Standard (AES) und Hash-based Message Authentication Code (HMAC) sowie für asymmetrische Zahlungskryptografieschlüssel wie RSA und Elliptic Curve Cryptography (ECC) verfügbar.
- Replikat-Regionsschlüssel sind schreibgeschützte Schlüssel. Alle Änderungen am Schlüssel für die primäre Region werden auf die Schlüssel der Replikatregion angewendet.
- Änderungen am Schlüssel für die primäre Region stimmen letztendlich mit den Regionsschlüsseln der Replikate überein.
- Schlüssel zur Zahlungskryptografie können nur mit derselben AWS Partition und demselben Konto repliziert werden.
- Die Anzahl der Schlüssel für die Replikatregion wird auf Ihr Limit für die AWS-Konto AWS Zahlungskryptografie angerechnet.
- Der Schlüssel für die primäre Region und der Schlüssel für die Replikatregion verwenden dieselbe Schlüssel-ID, sodass Sie in IAM-Richtlinien auf beide Schlüssel mit demselben ARN verweisen können.

- Sie müssen über `CreateKey` Berechtigungen für das Replikat verfügen, damit die Replikation AWS-Region erfolgreich ist.

Multi-Region Schlüsselreplikation aktivieren

Es gibt zwei Möglichkeiten, die Multi-Region Schlüsselreplikation für AWS Payment Cryptography-Schlüssel zu aktivieren.

1. AWS-Region: Die Multi-Region Schlüsselreplikation wird AWS-Region bei Aktivierung auf alle darin neu erstellten Schlüssel angewendet. Diese Methode ermöglicht eine konsistente Replikation für alle Schlüssel.
2. Spezifische Schlüssel für die AWS Zahlungskryptografie: Sie können die Multi-Region Schlüsselreplikation für einzelne Schlüssel verwalten, um eine detailliertere Kontrolle zu ermöglichen.

Sobald die Multi-Region Schlüsselreplikation aktiviert ist, werden Ihre Payment Cryptography-Schlüssel nach den von Ihnen angegebenen Werten repliziert. AWS-Regionen

> **Important**
>
> Multi-Region Die Schlüsselreplikation kann nicht unterbrochen werden. Ihre Schlüssel werden automatisch auf den von AWS-Regionen Ihnen angegebenen Wert repliziert, sobald die Replikation aktiviert ist. Multi-Region Die Schlüsselreplikation kann für bestimmte Schlüssel AWS-Region oder für Payment Cryptography deaktiviert werden. Sie müssen den Wert AWS-Region als Replikationsregion aus dem Schlüssel für die primäre Region entfernen, um den Schlüssel für die Replikatregion zu löschen.
> Alternativ können Sie den `StopKeyUsage` API- oder stop-key-usage CLI-Befehl auf Ihrem PRK aufrufen, um die Verwendung sowohl des PRK als auch aller zugehörigen RRKs zu beenden. Sie werden diese Schlüssel nicht für kryptografische Operationen verwenden können. Durch die Verwendung von `StopKeyUsage` API- oder stop-key-usage CLI-Befehlen wird die laufende Multi-Region Schlüsselreplikation, die für Ihr PRK aktiviert ist, nicht gestoppt.

Sie können die Einstellungen für die Multi-Region Schlüsselreplikation für AWS Payment Cryptography-Schlüssel in einem bestimmten Bereich überprüfen, AWS-Region indem Sie den `GetDefaultKeyReplicationRegions` API- oder get-default-key-replication-regions CLI-Befehl

aufrufen. Schlüssel in der AWS-Region Stelle, an der Sie diese API-Aktion oder diesen Befehl aufrufen, werden zu Ihrem ??? PRK.

Verwenden Sie die folgenden Verfahren, um die Multi-Region Schlüsselreplikation zu aktivieren.

For AWS-Region

- Verwenden Sie den folgenden Befehl, um die Multi-Region Schlüsselreplikation für einen von AWS-Region Ihnen angegebenen zu aktivieren. In diesem Beispiel ist die Multi-Region Schlüsselreplikation in USA Ost (Ohio) und USA West (Oregon) aktiviert. Um diesen Befehl zu verwenden, ersetzen Sie den Befehl *italicized placeholder text* im Beispiel durch Ihre eigenen Informationen.

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

> **Note**
>
> Wenn Sie die Multi-Region Schlüsselreplikation für an aktivieren, AWS-Region wird die Replikationskonfiguration vorhandener AWS Payment Cryptography Schlüssel nicht geändert. Sie können diese Funktion für vorhandene Schlüssel auf Schlüsselebene aktivieren. Nur Schlüssel, die nach der Aktivierung der Multi-Region Schlüsselreplikation für an erstellt wurden, verwenden AWS-Region die Einstellungen für die Regionsreplikation.

For specific AWS Payment Cryptography keys

- Verwenden Sie den folgenden Befehl, um die Multi-Region Schlüsselreplikation für bestimmte Payment Cryptography-Schlüssel zu aktivieren. In diesem Beispiel ist die Multi-Region Schlüsselreplikation in US East (Ohio) aktiviert. Um diesen Befehl zu verwenden, ersetzen Sie den Befehl *italicized placeholder text* im Beispiel durch Ihre eigenen Informationen.

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

Alternativ können Sie einen neuen Payment Cryptography Key erstellen, bei dem diese Funktion aktiviert ist, indem Sie die Replikation AWS-Regionen in Ihre Anfrage zur Schlüsselerstellung aufnehmen.

> **Note**
>
> Die Einstellungen für die Schlüsselreplikation haben Vorrang vor den AWS-Region Replikationseinstellungen.

Deaktivieren der Schlüsselreplikation Multi-Region

Wenn Sie die Multi-Region Schlüsselreplikation deaktivieren möchten, können Sie entweder den disable-default-key-replication oder den remove-key-replication-regions CLI-Befehl aufrufen, je nachdem, wie die Multi-Region Schlüsselreplikation aktiviert ist. Sie müssen den ARN des Schlüssels und den angeben, AWS-Region um die Multi-Region Schlüsselreplikation zu deaktivieren.

Überlegungen

Das Löschen von Schlüsseln für die Replikationsregion ist letztendlich konsistent.

Sie können die Einstellungen für die Multi-Region Schlüsselreplikation für AWS Payment Cryptography-Schlüssel in einem bestimmten Bereich überprüfen, AWS-Region indem Sie den `GetDefaultKeyReplicationRegions` API- oder get-default-key-replication-regions CLI-Befehl aufrufen.

Verwenden Sie die folgenden Verfahren, um die Multi-Region Schlüsselreplikation zu deaktivieren.

For AWS-Region

- Verwenden Sie den folgenden Befehl, um die Multi-Region Schlüsselreplikation für einen von AWS-Region Ihnen angegebenen zu deaktivieren. In diesem Beispiel ist die Multi-Region Schlüsselreplikation in US East (Ohio) deaktiviert. Um diesen Befehl zu verwenden, ersetzen Sie den Befehl *italicized placeholder text* im Beispiel durch Ihre eigenen Informationen.

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

For specific AWS Payment Cryptography keys

- Verwenden Sie den folgenden Befehl, um die Multi-Region Schlüsselreplikation für einen bestimmten Payment Cryptography-Schlüssel zu deaktivieren. In diesem Beispiel wird die Multi-Region Schlüsselreplikation in US East (Ohio) deaktiviert. Um diesen Befehl zu verwenden, ersetzen Sie den Befehl *italicized placeholder text* im Beispiel durch Ihre eigenen Informationen.

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

Sicherheitsüberlegungen

Im Folgenden finden Sie Sicherheitsüberlegungen, wenn Sie die Multi-Region Schlüsselreplikation für Ihre Payment Cryptography Schlüssel verwenden. Weitere Informationen finden Sie unter Bewährte Sicherheitsmethoden für AWS Kryptografie im Zahlungsverkehr.

- Beschränken Sie die gemeinsame Nutzung wichtiger Materialien.
- Halten Sie sich bei der Erstellung von IAM-Richtlinien an das Prinzip der geringsten Zugriffsrechte.
- Sie können keine Änderungen am Replikatregionsschlüssel vornehmen, da es sich um einen schreibgeschützten Schlüssel handelt.

Bewährte Methoden

Im Folgenden finden Sie einige bewährte Methoden, wenn Sie die Multi-Region Schlüsselreplikation mit AWS Payment Cryptography-Schlüsseln verwenden.

- Stellen Sie sicher, dass Ihre Anwendung auch dann funktioniert, wenn die Multi-Region Schlüsselreplikation auf die angegebenen Schlüssel nicht sofort AWS-Region erfolgt. Wenn Sie wissen möchten, wann die Multi-Region Schlüsselreplikation abgeschlossen ist, können Sie dies mithilfe der GetKey API-Aktion überwachen. Sie können wichtige Replikationsereignisse mit überwachen AWS CloudTrail.
- Testen und implementieren Sie automatisierte Bereitstellungsprozesse für den Fall eines Failovers von einer AWS-Region Region in eine andere.

Preisgestaltung

Ihnen werden Replikatregionsschlüssel in Rechnung gestellt, die Sie mit AWS Payment Cryptography erstellen. Diese Schlüssel werden pro Schlüssel berechnet. AWS-Region Die neuesten Preisinformationen für Payment Cryptography finden Sie auf der Preisseite für AWS Payment Cryptography.

Löschen von -Schlüsseln

Durch das Löschen eines AWS Payment Cryptography-Schlüssels werden das Schlüsselmaterial und alle mit dem Schlüssel verknüpften Metadaten gelöscht. Dies ist irreversibel, sofern keine Kopie des Schlüssels außerhalb von Payment Cryptography verfügbar ist. AWS Nachdem ein Schlüssel gelöscht wurde, können Sie die Daten, die mit diesem Schlüssel verschlüsselt wurden, nicht mehr entschlüsseln, was bedeutet, dass Daten möglicherweise nicht mehr wiederhergestellt werden können. Sie sollten einen Schlüssel nur löschen, wenn Sie sicher sind, dass Sie ihn nicht mehr benötigen und keine anderen Parteien diesen Schlüssel verwenden. Wenn Sie sich nicht sicher sind, sollten Sie erwägen, die Verwendung des Schlüssels zu beenden, anstatt ihn zu löschen. Sie können einen deaktivierten Schlüssel erneut aktivieren, falls Sie ihn später erneut verwenden müssen. Sie können einen gelöschten AWS Payment Cryptography Key jedoch nur wiederherstellen, wenn Sie ihn aus einer anderen Quelle erneut importieren können.

Bevor Sie einen Schlüssel löschen, sollten Sie sicherstellen, dass Sie den Schlüssel nicht mehr benötigen. AWS Payment Cryptography speichert die Ergebnisse kryptografischer Operationen wie CVV2 nicht und kann nicht feststellen, ob ein Schlüssel für persistentes kryptografisches Material benötigt wird.

AWS Payment Cryptography löscht niemals Schlüssel, die zu aktiven AWS Konten gehören, es sei denn, Sie planen ausdrücklich deren Löschung ein und die vorgeschriebene Wartezeit läuft ab.

Sie können sich jedoch aus einem oder mehreren der folgenden Gründe dafür entscheiden, einen AWS Payment Cryptography-Schlüssel zu löschen:

- Um den Schlüssellebenszyklus für einen Schlüssel abzuschließen, den Sie nicht mehr benötigen
- Um den Verwaltungsaufwand zu vermeiden, der mit der Verwaltung ungenutzter AWS Payment Cryptography-Schlüssel verbunden ist

> **Note**
>
> Wenn Sie Ihre schließen oder löschen AWS-Konto, ist Ihr AWS Payment Cryptography-Schlüssel nicht mehr zugänglich. Sie müssen die Löschung Ihres AWS Payment Cryptography-Schlüssels nicht getrennt von der Schließung des Kontos planen.

AWS Payment Cryptography zeichnet einen Eintrag in Ihrem AWS CloudTrail Protokoll auf, wenn Sie die Löschung des AWS Payment Cryptography-Schlüssels planen und wenn der AWS Payment Cryptography Schlüssel tatsächlich gelöscht wird.

Wenn Sie die Multi-Region Schlüsselreplikation verwenden und einen Payment Cryptography-Schlüssel löschen, bei dem es sich um einen Primärregionsschlüssel (PRK) handelt, werden die Replikatregionsschlüssel (RRK) ebenfalls automatisch gelöscht. Ein RRK kann nicht wie ein PRK gelöscht werden. Wenn Sie ein RRK löschen möchten, müssen Sie die Replikationsregionen für Ihr PRK ändern.

Über die Wartezeit

Da das Löschen eines Schlüssels irreversibel ist, müssen Sie bei AWS Payment Cryptography eine Wartezeit zwischen 3 und 180 Tagen festlegen. Die Standardwartezeit beträgt sieben Tage.

Die tatsächliche Wartezeit kann jedoch bis zu 24 Stunden länger sein als die, die Sie geplant haben. Verwenden Sie die GetKey Operationen, um das tatsächliche Datum und die Uhrzeit abzurufen, zu denen der AWS Payment Cryptography-Schlüssel gelöscht wird. Achten Sie darauf, die Zeitzone zu notieren.

Während der Wartezeit lauten der Status und Status des Schlüssels von AWS Payment Cryptography auf Ausstehende Löschung.

> **Note**
>
> Ein AWS Payment Cryptography-Schlüssel, dessen Löschung noch aussteht, kann für keine kryptografischen Operationen verwendet werden.

Nach Ablauf der Wartezeit löscht AWS Payment Cryptography den AWS Payment Cryptography Key, seine Aliase und alle zugehörigen Payment Cryptography Metadaten. AWS

Nutzen Sie die Wartezeit, um sicherzustellen, dass Sie den AWS Payment Cryptography-Schlüssel jetzt oder in Zukunft nicht benötigen. Wenn Sie feststellen, dass Sie den Schlüssel während der Wartezeit benötigen, können Sie das Löschen des Schlüssels vor Ablauf der Wartezeit stornieren. Nach Ablauf der Wartezeit können Sie das Löschen des Schlüssels nicht mehr stornieren, und der Dienst löscht den Schlüssel.

Example

In diesem Beispiel wird das Löschen eines Schlüssels angefordert. Neben den grundlegenden Schlüsselinformationen sind zwei wichtige Felder relevant: Der Schlüsselstatus wurde in DELETE_PENDING geändert, und delete gibt PendingTimestamp an, wann der Schlüssel derzeit gelöscht werden soll.

```
$ 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

In diesem Beispiel wird ein ausstehender Löschvorgang storniert. Nach erfolgreichem Abschluss wird ein Schlüssel nicht mehr gemäß dem vorherigen Zeitplan gelöscht. Die Antwort enthält die grundlegenden Schlüsselinformationen. Zusätzlich haben sich zwei relevante Felder geändert: `KeyState` und`deletePendingTimestamp`. `KeyState`wird auf den Wert CREATE_COMPLETE zurückgegeben, während entfernt `DeletePendingTimestamp` wird.

```
$ 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"
    }
}
```

Schlüssel importieren und exportieren

Sie können AWS Payment Cryptography-Schlüssel aus anderen Lösungen importieren und sie in andere Lösungen wie HSMs exportieren. Viele Kunden tauschen mithilfe der Import- und Exportfunktionen Schlüssel mit Dienstanbietern aus. Wir haben AWS Payment Cryptography so konzipiert, dass ein moderner, elektronischer Ansatz für die Schlüsselverwaltung verwendet wird, der Sie bei der Einhaltung von Vorschriften und Kontrollen unterstützt. Wir empfehlen, den standardbasierten elektronischen Schlüsselaustausch anstelle von papierbasierten Schlüsselkomponenten zu verwenden. Wenn Sie die Verarbeitung von Schlüsselkomponenten in Papierform so lange fortsetzen müssen, bis alle Partner den elektronischen Schlüsselaustausch unterstützen, können Sie Folgendes verwenden. Physischer Schlüsselaustausch

Mindeststärken der Schlüssel und deren Auswirkung auf Import- und Exportfunktionen

PCI erfordert bestimmte Mindestschlüsselstärken für kryptografische Operationen, Schlüsselspeicherung und Schlüsselübertragung. Diese Anforderungen können sich ändern, wenn die PCI-Standards überarbeitet werden. Die Regeln legen fest, dass Wrapping-Schlüssel, die für die Speicherung oder den Transport verwendet werden, mindestens so stark sein müssen wie der zu schützende Schlüssel. Wir setzen diese Anforderung beim Export automatisch durch und verhindern, dass Schlüssel durch schwächere Schlüssel geschützt werden, wie in der folgenden Tabelle dargestellt.

Die folgende Tabelle zeigt die unterstützten Kombinationen aus Schlüsselumbruch, zu schützenden Schlüsseln und Schutzmethoden.

	Schlüssel umschließen											
Schlüssel zum Schutz	TDE	TDE	AES_	AES_	AES_	RSA_	RSA_	RSA_	ECC_	ECC_	ECC_	Hinweise
TDES_2-SC HLÜSSEL	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3 RSA	TR-3 RSA	TR-3 RSA	ECDI	ECDI	ECDI	
TDES_3-SC HLÜSSEL	✗ Wird nicht	TR-3	TR-3	TR-3	TR-3	TR-3 RSA	TR-3 RSA	TR-3 RSA	ECDI	ECDI	ECDI	

	Schlüssel umschließen											
Schlüssel zum Schutz	TDES unterzt	TDES	AES_	AES_	AES_	RSA_	RSA_	RSA_	ECC_	ECC_	ECC_	Hinweise
AES-128	✗ Nicht unterzt	✗ Nicht unterzt	TR-3	TR-3	TR-3	✗ Nicht unterzt	TR-3 RSA	TR-3 RSA	ECDH	ECDH	ECDH	
AES_192	✗ Nicht unterzt	✗ Nicht unterzt	✗ Nicht unterzt	TR-3	TR-3	✗ Nicht unterzt	✗ Nicht unterzt	✗ Nicht unterzt	✗ Nicht unterzt	ECDH	ECDH	
AES_256	✗ Nicht unterzt	✗ Nicht unterzt	✗ Nicht unterzt	✗ Nicht unterzt	TR-3	✗ Nicht unterzt	✗ Nicht unterzt	✗ Nicht unterzt	✗ Nicht unterzt	✗ Nicht unterzt	ECDH	

Weitere Informationen finden Sie in Anhang D — Minimale und äquivalente Schlüsselgrößen und -stärken für zugelassene Algorithmen in den PCI-HSM-Standards.

Austausch von Schlüsselverschlüsselungsschlüsseln (KEK)

Wir empfehlen die Verwendung des X9.24 TR-34 ANSI-Standards. Dieser anfängliche Schlüsseltyp kann als Key Encryption Key (KEK), Zone Master Key (ZMK) oder Zone Control Master Key (ZCMK) bezeichnet werden. Wenn Ihre Systeme oder Partner TR-34 noch keinen Support bieten, können Sie RSA verwenden. Wrap/Unwrap Wenn Ihre Anforderungen den Austausch von AES-256 Schlüsseln beinhalten, können Sie ??? ECDH verwenden.

> **Note**
>
> Um Ihre eigenen Testschlüssel zu importieren oder Schlüssel mit Ihren vorhandenen HSMs zu synchronisieren, lesen Sie bitte den AWS Payment Cryptography Beispielcode unter. GitHub

Austausch von Arbeitsschlüsseln (WK)

Wir verwenden Industriestandards (ANSI X9.24 TR 31-2018 und X9.143) für den Austausch funktionierender Schlüssel. Dies setzt voraus, dass Sie bereits einen KEK mithilfe von RSA Wrap TR-34, ECDH oder ähnlichen Schemata ausgetauscht haben. Dieser Ansatz erfüllt die PCI-PIN-Anforderung, Schlüsselmaterial jederzeit kryptografisch an seinen Typ und seine Verwendung zu binden. Zu den Arbeitsschlüsseln gehören die Arbeitsschlüssel des Acquirers, die Arbeitsschlüssel des Ausstellers, BDK und IPEK.

Themen

- Schlüssel importieren
- Schlüssel exportieren
- Fortgeschrittene Themen

Schlüssel importieren

> **Important**
>
> Für Beispiele ist die neueste Version von AWS CLI V2 erforderlich. Bevor Sie beginnen, stellen Sie sicher, dass Sie auf die neueste Version aktualisiert haben.

Inhalt

- Einführung in das Importieren von Schlüsseln
- Symmetrische Schlüssel importieren
  - Schlüssel mithilfe asymmetrischer Techniken importieren () TR-34
  - Schlüssel mithilfe asymmetrischer Techniken (ECDH) importieren
  - Importieren Sie Schlüssel mithilfe asymmetrischer Techniken (RSA Unwrap)

- Importieren Sie symmetrische Schlüssel mithilfe eines vorab festgelegten Schlüsselaustauschschlüssels () TR-31
- Import asymmetrischer (RSA, ECC) öffentlicher Schlüssel
  - Öffentliche RSA-Schlüssel importieren
  - Öffentliche ECC-Schlüssel importieren

Einführung in das Importieren von Schlüsseln

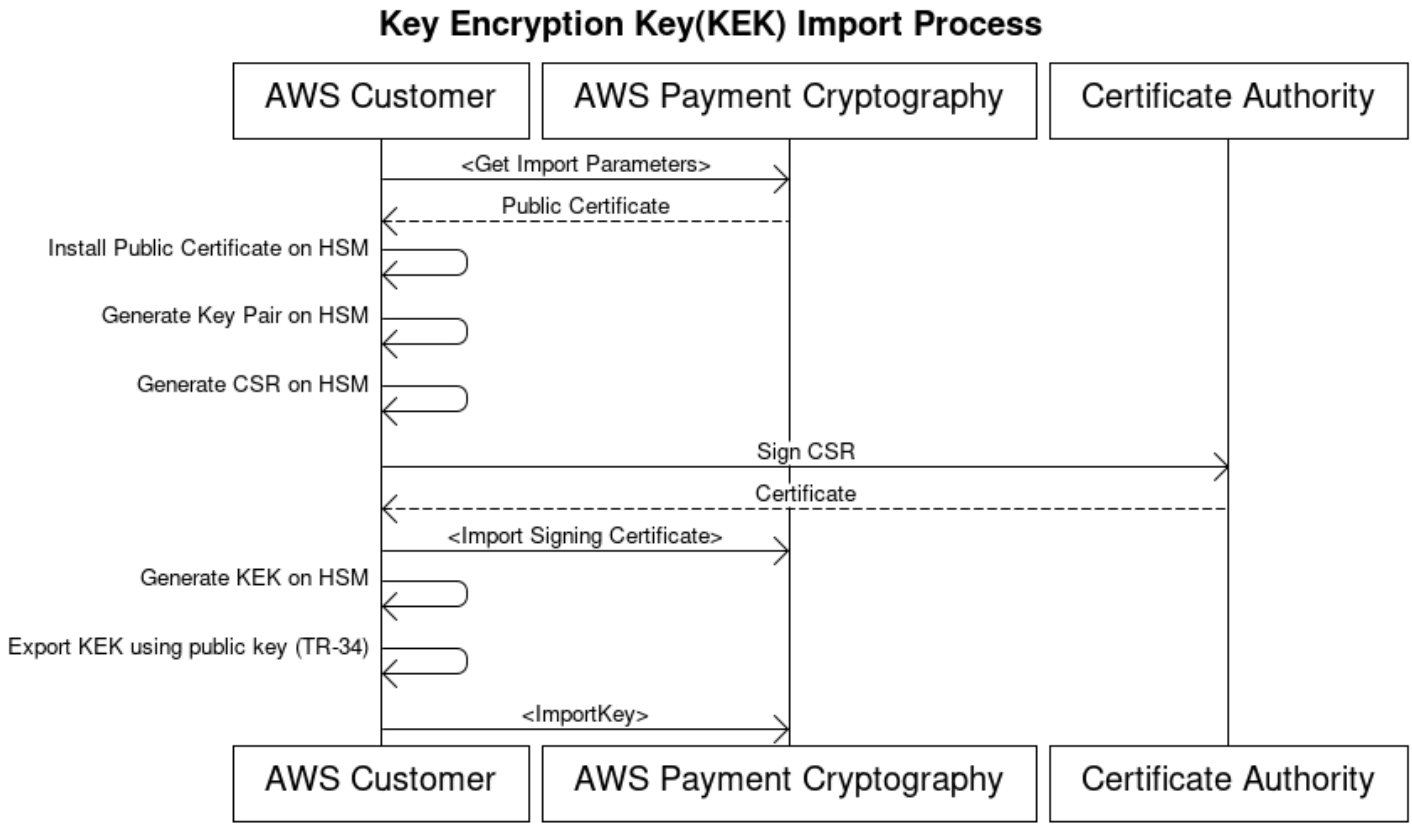
> **Note**
>
> Beim Import von Schlüsseln mithilfe X9.143 von TR-34 Schlüsselblöcken behält AWS Payment Cryptography in der Regel optionale Header bei (verwendet sie jedoch nicht). TR-31 Der HM-Header (HMAC-Hash-Typ) wird bei kryptografischen Vorgängen verwendet. Der KP-Header (KCV des Wrapping-Schlüssels) ist spezifisch für den Importvorgang und wird nicht beibehalten.

Beim Austausch von Schlüsseln mit einer Gegenpartei wird in der Regel zuerst ein Key Exchange Key (KEK) ausgetauscht. Dieser Schlüssel wird dann zum Schutz nachfolgender Schlüssel verwendet. Unter Verwendung elektronischer Formate kann der KEK mithilfe asymmetrischer Techniken wie ECDH oder TR-34 RSA Wrap ausgetauscht werden. Nachfolgende Schlüssel werden unter Verwendung eines symmetrischen Schlüsselaustauschs ausgetauscht, z. B. TR-31 Dieser KEK wird eine lange Lebensdauer haben und möglicherweise nur alle paar Jahre aktualisiert werden, je nach Richtlinie und definierter Kryptoperiode.

Wenn nur ein oder zwei Schlüssel ausgetauscht werden, können Sie auch asymmetrische Techniken verwenden, um diesen Schlüssel direkt auszutauschen, z. B. einen BDK. AWS Payment Cryptography unterstützt beide Methoden des Schlüsselaustauschs.

Symmetrische Schlüssel importieren

Schlüssel mithilfe asymmetrischer Techniken importieren () TR-34



TR-34 verwendet asymmetrische RSA-Kryptografie, um symmetrische Schlüssel für den Austausch zu verschlüsseln und zu signieren. Dadurch wird sowohl die Vertraulichkeit (Verschlüsselung) als auch die Integrität (Signatur) des verpackten Schlüssels gewährleistet.

Um Ihre eigenen Schlüssel zu importieren, schauen Sie sich das Beispielprojekt AWS Payment Cryptography unter an GitHub. Anweisungen zum Verwenden von import/export Schlüsseln von anderen Plattformen finden Sie im Beispielcode auf GitHub oder lesen Sie im Benutzerhandbuch für diese Plattformen nach.

1. Rufen Sie den Befehl „Import initialisieren“ auf

   Rufen Sie `get-parameters-for-import` auf, um den Importvorgang zu initialisieren. Diese API generiert ein Schlüsselpaar für Schlüsselimporte, signiert den Schlüssel und gibt das Zertifikat und den Zertifikatsstamm zurück. Verschlüsseln Sie den Schlüssel, der exportiert werden soll, mit diesem Schlüssel. In der TR-34 Terminologie wird dies als KRD-Zertifikat

bezeichnet. Diese Zertifikate sind Base64-kodiert, kurzlebig und nur für diesen Zweck bestimmt. Speichern Sie den Wert. `ImportToken`

```
$ 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. Installieren Sie das öffentliche Zertifikat auf dem Schlüsselquellsystem

   Bei den meisten HSMs müssen Sie das in Schritt 1 generierte öffentliche Zertifikat installieren, laden oder als vertrauenswürdig einstufen, um Schlüssel damit zu exportieren. Dies kann je nach HSM die gesamte Zertifikatskette oder nur das Stammzertifikat aus Schritt 1 umfassen.

3. Generieren Sie das Schlüsselpaar auf dem Quellsystem und stellen Sie die Zertifikatskette bereit für AWS Kryptografie im Zahlungsverkehr

   Um die Integrität der übertragenen Nutzdaten zu gewährleisten, signiert die sendende Partei (Key Distribution Host oder KDH) sie. Generieren Sie zu diesem Zweck einen öffentlichen Schlüssel und erstellen Sie ein Zertifikat für öffentliche Schlüssel (X509), um Payment Cryptography zur Verfügung zu AWS stellen.

   Wenn Sie Schlüssel von einem HSM übertragen, erstellen Sie ein Schlüsselpaar auf diesem HSM. Das HSM, ein Drittanbieter oder ein Dienst wie dieser AWS Private CA kann das Zertifikat generieren.

   Laden Sie das Stammzertifikat mit den AWS `importKey` Befehlen KeyMaterialType of `RootCertificatePublicKey` und KeyUsageType of in Payment Cryptography. `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`

   Verwenden Sie für Zwischenzertifikate den `importKey` Befehl mit KeyMaterialType of `TrustedCertificatePublicKey` und KeyUsageType of`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`. Wiederholen Sie diesen Vorgang

für mehrere Zwischenzertifikate. Verwenden Sie das Zertifikat `KeyArn` des letzten importierten Zertifikats in der Kette als Eingabe für nachfolgende Importbefehle.

> **Note**
>
> Importieren Sie das Leaf-Zertifikat nicht. Geben Sie es direkt während des Importbefehls an.

4. Schlüssel aus dem Quellsystem exportieren

   Viele HSMs und verwandte Systeme unterstützen den Export von Schlüsseln unter Verwendung der TR-34 Norm. Geben Sie den öffentlichen Schlüssel aus Schritt 1 als KRD-Zertifikat (Verschlüsselung) und den Schlüssel aus Schritt 3 als KDH-Zertifikat (Signierung) an. Für den Import in AWS Payment Cryptography geben Sie das Format als TR-34.2012 Nicht-CMS-Format mit zwei Durchgängen an, das auch als TR-34 Diebold-Format bezeichnet werden kann.

5. Rufen Sie Import Key auf

   Rufen Sie die ImportKey-API mit einem KeyMaterialType von auf auf. `TR34_KEY_BLOCK` Verwenden Sie den KeyArn der zuletzt in Schritt 3 importierten Zertifizierungsstelle für`certificate-authority-public-key-identifier`, das verpackte Schlüsselmaterial aus Schritt 4 für und das `key-material` Blattzertifikat aus Schritt 3 für. `signing-key-certificate` Fügen Sie das Import-Token aus Schritt 1 hinzu.

```
$ 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.
 \
    }'
```

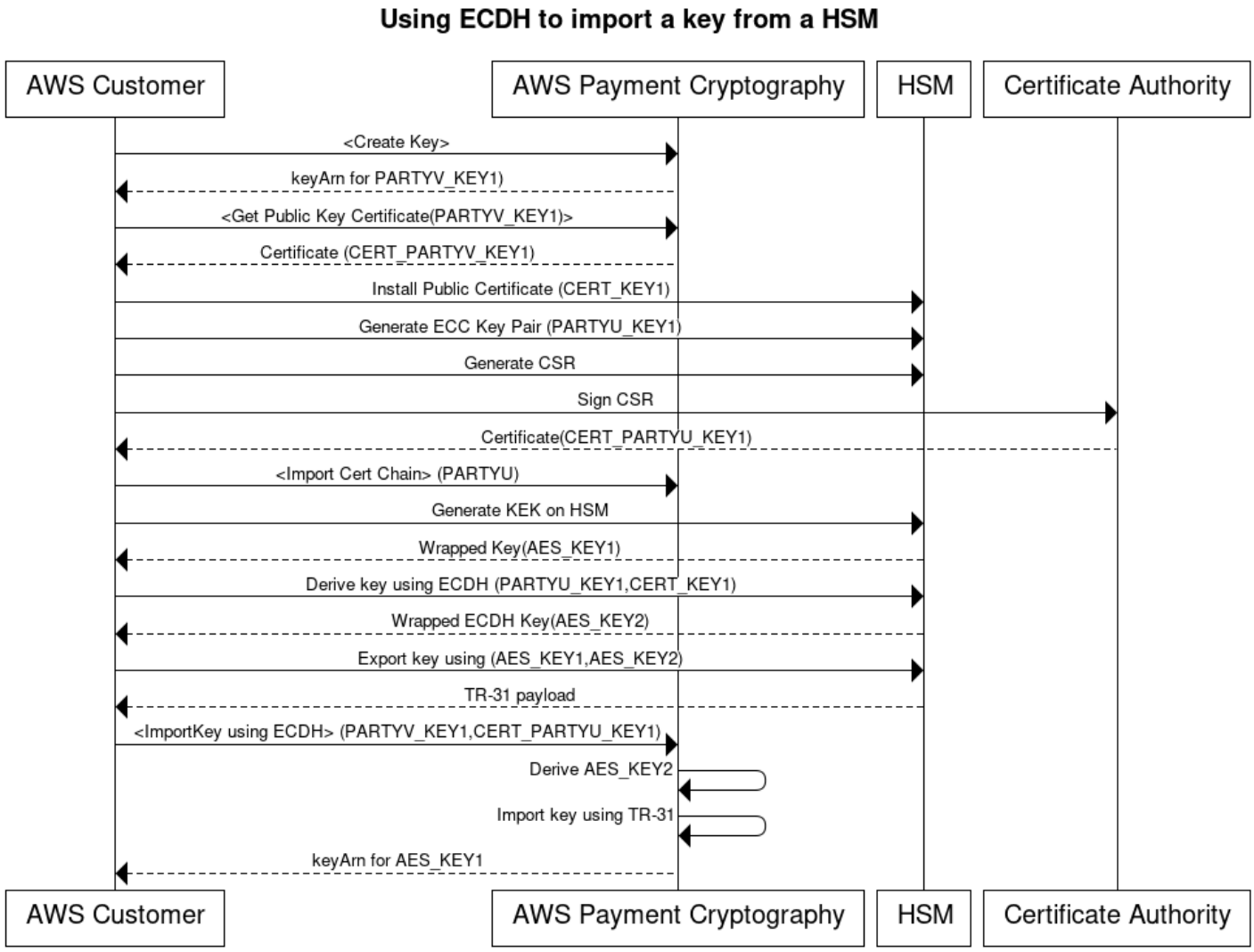
```
{
  "Key": {
    "CreateTimestamp": "2023-06-13T16:52:52.859000-04:00",
    "Enabled": true,
```

```
    "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. Verwenden Sie den importierten Schlüssel für kryptografische Operationen oder den nachfolgenden Import

   Wenn der importierte Schlüssel TR31_K0_KEY_ENCRYPTION_KEY KeyUsage war, können Sie diesen Schlüssel für nachfolgende Schlüsselimporte verwenden. TR-31 Für andere Schlüsseltypen (wie TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY) können Sie den Schlüssel direkt für kryptografische Operationen verwenden.

Schlüssel mithilfe asymmetrischer Techniken (ECDH) importieren



Elliptic Curve Diffie-Hellman (ECDH) verwendet die asymmetrische ECC-Kryptografie, um einen gemeinsamen Schlüssel zwischen zwei Parteien zu erstellen, ohne dass zuvor Schlüssel ausgetauscht werden müssen. ECDH-Schlüssel sind kurzlebig, weshalb Payment Cryptography sie nicht speichert. AWS Bei diesem Prozess wird mithilfe von ECDH ein einmaliges Passwort KBPK/KEK abgeleitet. Dieser abgeleitete Schlüssel wird sofort zum Umschließen des eigentlichen Schlüssels verwendet, den Sie übertragen möchten. Dabei kann es sich um einen anderen KBPK-Schlüssel, einen IPEK-Schlüssel oder andere Schlüsseltypen handeln.

Beim Import wird das sendende System allgemein als Partei U (Initiator) und AWS Payment Cryptography als Party V (Responder) bezeichnet.

> **Note**
>
> ECDH kann zwar für den Austausch aller symmetrischen Schlüsseltypen verwendet werden, es ist jedoch die einzige Methode, mit der Schlüssel sicher übertragen werden können. AES-256

1. Generieren Sie ein ECC-Schlüsselpaar

   Rufen Sie `create-key` auf, um ein ECC-Schlüsselpaar für diesen Prozess zu erstellen. Diese API generiert ein Schlüsselpaar für Schlüsselimporte oder -exporte. Geben Sie bei der Erstellung an, welche Art von Schlüsseln mit diesem ECC-Schlüssel abgeleitet werden können. Wenn Sie ECDH zum Austauschen (Umschließen) anderer Schlüssel verwenden, verwenden Sie den Wert. `TR31_K1_KEY_BLOCK_PROTECTION_KEY`

   > **Note**
   >
   > Obwohl Low-Level-ECDH einen abgeleiteten Schlüssel generiert, der für jeden Zweck verwendet werden kann, begrenzt AWS Payment Cryptography die versehentliche Wiederverwendung eines Schlüssels für mehrere Zwecke, indem ein Schlüssel nur für einen einzigen abgeleiteten Schlüsseltyp verwendet werden kann.

```
$ 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. Holen Sie sich das Zertifikat für einen öffentlichen Schlüssel

   Rufen Sie an`get-public-key-certificate`, um den öffentlichen Schlüssel als X.509 Zertifikat zu erhalten, das von der CA Ihres Kontos signiert wurde und spezifisch für AWS Zahlungskryptografie in einer bestimmten Region ist.

   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. Installieren Sie das öffentliche Zertifikat auf dem System der Gegenpartei (Partei U)

   Bei vielen HSMs müssen Sie das in Schritt 1 generierte öffentliche Zertifikat installieren, laden oder als vertrauenswürdig einstufen, um Schlüssel damit zu exportieren. Dies kann je nach

HSM die gesamte Zertifikatskette oder nur das Stammzertifikat aus Schritt 1 umfassen. Weitere Informationen finden Sie in Ihrer HSM-Dokumentation.

4. Generieren Sie das ECC-Schlüsselpaar auf dem Quellsystem und stellen Sie die Zertifikatskette bereit für AWS Kryptografie im Zahlungsverkehr

   Bei ECDH generiert jede Partei ein Schlüsselpaar und einigt sich auf einen gemeinsamen Schlüssel. Damit AWS Payment Cryptography den Schlüssel ableiten kann, benötigt es den öffentlichen Schlüssel der Gegenpartei im X.509 öffentlichen Schlüsselformat.

   Wenn Sie Schlüssel von einem HSM übertragen, erstellen Sie ein Schlüsselpaar auf diesem HSM. Bei HSMs, die Schlüsselblöcke unterstützen, sieht der Schlüsselheader ähnlich aus wie. `D0144K3EX00E0000` Beim Erstellen des Zertifikats generieren Sie in der Regel eine CSR auf dem HSM, und dann AWS Private CA kann das HSM, ein Drittanbieter oder ein Dienst wie dieser das Zertifikat generieren.

   Laden Sie das Stammzertifikat mithilfe des `importKey` Befehls mit KeyMaterialType of und of in AWS Payment Cryptography. `RootCertificatePublicKey` KeyUsageType `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`

   Verwenden Sie für Zwischenzertifikate den `importKey` Befehl mit KeyMaterialType of `TrustedCertificatePublicKey` und KeyUsageType of`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`. Wiederholen Sie diesen Vorgang für mehrere Zwischenzertifikate. Verwenden Sie das Zertifikat `KeyArn` des letzten importierten Zertifikats in der Kette als Eingabe für nachfolgende Importbefehle.

   > **Note**
   >
   > Importieren Sie das Leaf-Zertifikat nicht. Geben Sie es direkt während des Importbefehls an.

5. Leiten Sie mithilfe von ECDH auf Party U HSM einen Einmalschlüssel ab

   Viele HSMs und verwandte Systeme unterstützen die Erstellung von Schlüsseln mithilfe von ECDH. Geben Sie den öffentlichen Schlüssel aus Schritt 1 als öffentlichen Schlüssel und den Schlüssel aus Schritt 3 als privaten Schlüssel an. Zulässige Optionen, z. B. Ableitungsmethoden, finden Sie im API-Leitfaden.

> **Note**
>
> Die Ableitungsparameter wie der Hash-Typ müssen auf beiden Seiten exakt übereinstimmen. Andernfalls generieren Sie einen anderen Schlüssel.

6. Schlüssel aus dem Quellsystem exportieren

   Exportieren Sie abschließend den Schlüssel, den Sie mithilfe von TR-31 Standardbefehlen an AWS Payment Cryptography übertragen möchten. Geben Sie den von ECDH abgeleiteten Schlüssel als KBPK an. Bei dem zu exportierenden Schlüssel kann es sich um einen beliebigen TDES- oder AES-Schlüssel handeln, sofern TR-31 gültige Kombinationen gelten, sofern der umschließende Schlüssel mindestens so stark ist wie der zu exportierende Schlüssel.

7. Rufen Sie Import Key auf

   Rufen Sie die `import-key` API mit einem KeyMaterialType von auf auf`DiffieHellmanTr31KeyBlock`. Verwenden Sie den KeyArn der zuletzt in Schritt 3 importierten CA für`certificate-authority-public-key-identifier`, das verpackte Schlüsselmaterial aus Schritt 4 für `key-material` und das Leaf-Zertifikat aus Schritt 3 für`public-key-certificate`. Schließen Sie den ARN des privaten Schlüssels aus Schritt 1 ein.

```
$ 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. Verwenden Sie den importierten Schlüssel für kryptografische Operationen oder den nachfolgenden Import

   Wenn der importierte Schlüssel TR31_K0_KEY_ENCRYPTION_KEY KeyUsage war, können Sie diesen Schlüssel für nachfolgende Schlüsselimporte verwenden. TR-31 Für andere Schlüsseltypen (wie TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY) können Sie den Schlüssel direkt für kryptografische Operationen verwenden.

Importieren Sie Schlüssel mithilfe asymmetrischer Techniken (RSA Unwrap)

Überblick: AWS Payment Cryptography unterstützt RSA wrap/unwrap für den Schlüsselaustausch, wenn dies nicht möglich ist. TR-34 Ebenso verwendet diese Technik asymmetrische RSA-Kryptografie TR-34, um symmetrische Schlüssel für den Austausch zu verschlüsseln. Im Gegensatz zu TR-34 dieser Methode muss der Absender die Nutzlast jedoch nicht signieren. Außerdem gewährleistet diese RSA-Wrap-Technik nicht die Integrität der Schlüsselmetadaten während der Übertragung, da sie keine Schlüsselblöcke enthält.

> **Note**
>
> Sie können RSA-Wrap verwenden, um TDES und Schlüssel zu importieren oder zu exportieren. AES-128

1. Rufen Sie den Befehl Initialize Import auf

   Rufen Sie get-parameters-for-import auf, um den Importvorgang mit einem `KeyMaterialType` von zu initialisieren. `KEY_CRYPTOGRAM` `WrappingKeyAlgorithm`Wird beim Austausch von TDES-Schlüsseln verwendet`RSA_2048`. Verwenden Sie `RSA_3072` oder `RSA_4096` beim Austausch von TDES oder AES-128 Schlüsseln. Diese API generiert ein Schlüsselpaar für Schlüsselimporte, signiert den Schlüssel mithilfe eines Zertifikatsstamms und gibt sowohl das Zertifikat als auch den Zertifikatsstamm zurück. Verschlüsseln Sie den zu exportierenden Schlüssel mit diesem Schlüssel. Diese Zertifikate sind kurzlebig und nur für diesen Zweck bestimmt.

   ```
   $ 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. Installieren Sie das öffentliche Zertifikat auf dem Schlüsselquellsystem

Bei vielen HSMs müssen Sie das in Schritt 1 generierte öffentliche Zertifikat (and/orsein Stammzertifikat) installieren, laden oder als vertrauenswürdig einstufen, um Schlüssel damit zu exportieren.

3. Schlüssel aus dem Quellsystem exportieren

   Viele HSMs und verwandte Systeme unterstützen den Export von Schlüsseln mithilfe von RSA Wrap. Geben Sie den öffentlichen Schlüssel aus Schritt 1 als Verschlüsselungszertifikat () an. `WrappingKeyCertificate` Wenn Sie die Vertrauenskette benötigen, verwenden Sie die `WrappingKeyCertificateChain` aus Schritt 1. Wenn Sie den Schlüssel aus Ihrem HSM exportieren, geben Sie das Format als RSA mit Padding Mode = PKCS #1 v2.2 OAEP (mit SHA 256 oder SHA 512) an.

4. Rufen Sie Schlüssel importieren

   Rufen Sie die import-key API mit einem `KeyMaterialType` von auf. `KeyMaterial` Sie benötigen das `ImportToken` aus Schritt 1 und das `key-material` (verpackte Schlüsselmaterial) aus Schritt 3. Geben Sie die Schlüsselparameter (z. B. Key Usage) an, da RSA Wrap keine Schlüsselblöcke verwendet.

```
$ 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"
    }
  }
}
```

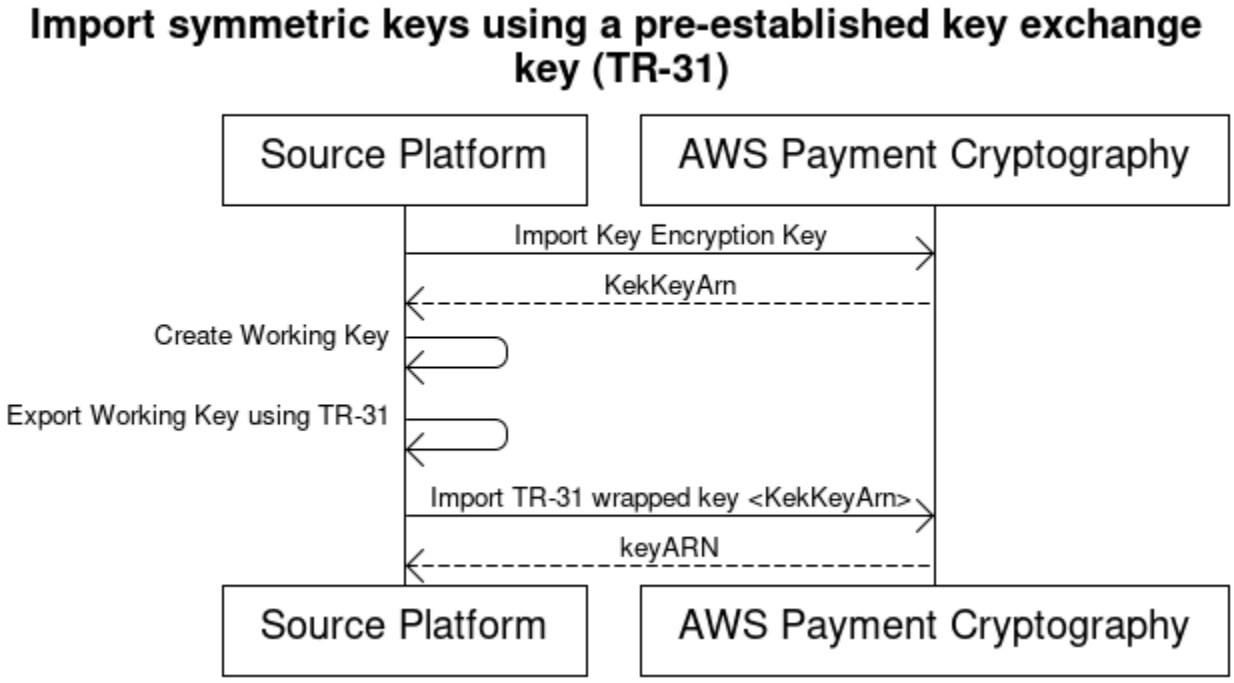
```
$ aws payment-cryptography import-key --cli-input-json file://import-key-
cryptogram.json
```

```
{
  "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. Verwenden Sie den importierten Schlüssel für kryptografische Operationen oder nachfolgende Importe

   Wenn der importierte Schlüssel `TR31_K0_KEY_ENCRYPTION_KEY` oder `KeyUsage` war`TR31_K1_KEY_BLOCK_PROTECTION_KEY`, können Sie diesen Schlüssel für nachfolgende Schlüsselimporte verwenden TR-31. Wenn der Schlüsseltyp ein anderer Typ war (z. B.`TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY`), können Sie den Schlüssel direkt für kryptografische Operationen verwenden.

Importieren Sie symmetrische Schlüssel mithilfe eines vorab festgelegten Schlüsselaustauschschlüssels () TR-31



Wenn mehrere Schlüssel ausgetauscht werden oder die Schlüsselrotation unterstützt wird, tauschen die Partner in der Regel zuerst einen Initial Key Encryption Key (KEK) aus. Sie können KEK gegen AWS Payment Cryptography austauschen, indem Sie Techniken wie TR-34 oder Physischer Schlüsselaustausch verwenden.

Nachdem Sie einen KEK eingerichtet haben, können Sie ihn für den Transport nachfolgender Schlüssel (einschließlich anderer KEKs) verwenden. AWS Payment Cryptography unterstützt diesen Schlüsselaustausch mithilfe von ANSI TR-31, das von HSM-Anbietern häufig verwendet und unterstützt wird.

1. Schlüsselverschlüsselungsschlüssel (KEK) importieren

   Stellen Sie sicher, dass Sie Ihren KEK bereits importiert haben und KeyArn (oder KeyAlias) verfügbar haben.

2. Erstellen Sie den Schlüssel auf der Quellplattform

   Wenn der Schlüssel nicht existiert, erstellen Sie ihn auf der Quellplattform. Alternativ können Sie den Schlüssel auf AWS Payment Cryptography erstellen und den export Befehl verwenden.

3. Exportieren Sie den Schlüssel von der Quellplattform

   Geben Sie beim Exportieren das Exportformat als an TR-31. Die Quellplattform fragt nach dem zu exportierenden Schlüssel und dem zu verwendenden Schlüsselverschlüsselungsschlüssel.

4. Importiere in AWS Kryptografie im Zahlungsverkehr

   Verwenden Sie beim Aufrufen des import-key Befehls den KeyArn (oder Alias) Ihres Schlüsselverschlüsselungsschlüssels für`WrappingKeyIdentifier`. Verwenden Sie die Ausgabe der Quellplattform für`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"
  }
}
```

Import asymmetrischer (RSA, ECC) öffentlicher Schlüssel

Alle importierten Zertifikate müssen mindestens so stark sein wie das ausstellende (Vorgänger-) Zertifikat in der Kette. Das bedeutet, dass eine RSA_2048-CA nur zum Schutz eines RSA_2048-Blattzertifikats verwendet werden kann und ein ECC-Zertifikat durch ein anderes ECC-Zertifikat gleicher Stärke geschützt werden muss. Ein ECC P384-Zertifikat kann nur von einer P384- oder P521-CA ausgestellt werden. Alle Zertifikate müssen zum Zeitpunkt des Imports noch nicht abgelaufen sein.

Öffentliche RSA-Schlüssel importieren

AWS Payment Cryptography unterstützt den Import öffentlicher RSA-Schlüssel als Zertifikate. X.509 Um ein Zertifikat zu importieren, importieren Sie zuerst das Stammzertifikat. Alle Zertifikate müssen zum Zeitpunkt des Imports noch nicht abgelaufen sein. Das Zertifikat sollte im PEM-Format vorliegen und Base64-kodiert sein.

1. Importieren Sie das Stammzertifikat in AWS Kryptografie im Zahlungsverkehr

   Verwenden Sie den folgenden Befehl, um das Stammzertifikat zu importieren:

Example

2. Importieren Sie das Public-Key-Zertifikat in AWS Kryptografie im Zahlungsverkehr

   Sie können jetzt einen öffentlichen Schlüssel importieren. Da TR-34 ECDH darauf angewiesen sind, das Leaf-Zertifikat zur Laufzeit zu übergeben, wird diese Option nur verwendet, wenn Daten mit einem öffentlichen Schlüssel aus einem anderen System verschlüsselt werden. KeyUsage wird auf TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION gesetzt.

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"
  }
}
```

Öffentliche ECC-Schlüssel importieren

AWS Payment Cryptography unterstützt den Import von öffentlichen ECC-Schlüsseln als Zertifikate. X.509 Um ein Zertifikat zu importieren, importieren Sie zuerst das Root-CA-Zertifikat und alle Zwischenzertifikate. Alle Zertifikate müssen zum Zeitpunkt des Imports noch nicht abgelaufen sein. Das Zertifikat sollte im PEM-Format vorliegen und Base64-kodiert sein.

1. Importieren Sie das ECC-Root-Zertifikat in AWS Kryptografie im Zahlungsverkehr

   Verwenden Sie den folgenden Befehl, um das Stammzertifikat zu importieren:

Example

2. Importieren Sie das Zwischenzertifikat AWS Kryptografie im Zahlungsverkehr

   Verwenden Sie den folgenden Befehl, um ein Zwischenzertifikat zu importieren:

Example

3. Importieren Sie das Public-Key-Zertifikat (Leaf) in AWS Kryptografie im Zahlungsverkehr

   Obwohl Sie ein ECC-Leaf-Zertifikat importieren können, gibt es in AWS Payment Cryptography derzeit außer der Speicherung keine definierten Funktionen dafür. Das liegt daran, dass bei der Verwendung von ECDH-Funktionen das Leaf-Zertifikat zur Laufzeit übergeben wird.

Schlüssel exportieren

Inhalt

- Exportiert symmetrische Schlüssel
  - Exportieren Sie Schlüssel mithilfe asymmetrischer Techniken () TR-34
  - Exportieren Sie Schlüssel mit asymmetrischen Techniken (ECDH)
  - Exportieren Sie Schlüssel mithilfe asymmetrischer Techniken (RSA Wrap)
  - Exportieren Sie symmetrische Schlüssel mithilfe eines vorab festgelegten Schlüsselaustauschschlüssels () TR-31
- Exportieren Sie die DUPT-Anfangsschlüssel () IPEK/IK
- Geben Sie Schlüsselblock-Header für den Export an
  - Allgemeine Header
- Exportiert asymmetrische (RSA) Schlüssel

Exportiert symmetrische Schlüssel

> **Important**
>
> Stellen Sie sicher, dass Sie die neueste Version von haben, AWS CLI bevor Sie beginnen. Informationen zum Upgrade finden Sie unter Installation von AWS CLI.

Exportieren Sie Schlüssel mithilfe asymmetrischer Techniken () TR-34

TR-34 verwendet asymmetrische RSA-Kryptografie, um symmetrische Schlüssel für den Austausch zu verschlüsseln und zu signieren. Die Verschlüsselung schützt die Vertraulichkeit, während die Signatur die Integrität gewährleistet. Wenn Sie Schlüssel exportieren, fungiert AWS Payment Cryptography als Key Distribution Host (KDH), und Ihr Zielsystem wird zum Schlüsselempfangsgerät (KRD).

> **Note**
>
> Wenn Ihr HSM zwar den TR-34 Export, aber keinen TR-34 Import unterstützt, empfehlen wir Ihnen, zunächst einen gemeinsamen KEK zwischen Ihrem HSM und Payment Cryptography einzurichten. AWS TR-34 Sie können ihn dann verwenden TR-31, um Ihre verbleibenden Schlüssel zu übertragen.

1. Initialisieren Sie den Exportvorgang

   Ausführenget-parameters-for-export, um ein Schlüsselpaar für Schlüsselexporte zu generieren. Wir verwenden dieses Schlüsselpaar, um die TR-34 Payload zu signieren. In der TR-34 Terminologie ist dies das KDH-Signaturzertifikat. Die Zertifikate sind kurzlebig und nur für die unter angegebene Dauer gültig. `ParametersValidUntilTimestamp`

   > **Note**
   >
   > Alle Zertifikate sind in Base64-Kodierung.

   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. Importiere die AWS Kryptografie-Zertifikat für Zahlungen in Ihr Empfangssystem

   Importieren Sie die Zertifikatskette aus Schritt 1 in Ihr Empfangssystem.

3. Richten Sie die Zertifikate Ihres Empfangssystems ein

Um die übertragene Nutzlast zu schützen, verschlüsselt die sendende Partei (KDH) sie. Ihr Empfangssystem (in der Regel Ihr HSM oder das HSM Ihres Partners) muss einen öffentlichen Schlüssel generieren und ein Zertifikat für öffentliche Schlüssel erstellen. X.509 Sie können AWS Private CA es verwenden, um Zertifikate zu generieren, aber Sie können jede beliebige Zertifizierungsstelle verwenden.

Nachdem Sie das Zertifikat erhalten haben, importieren Sie das Stammzertifikat mithilfe des ImportKey Befehls in AWS Payment Cryptography. Setzen Sie `KeyMaterialType` auf `RootCertificatePublicKey` und `KeyUsageType` auf `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`.

Wir verwenden `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE` als, `KeyUsageType` weil dies der Stammschlüssel ist, der das Leaf-Zertifikat signiert. Sie müssen keine Leaf-Zertifikate in AWS Payment Cryptography importieren — Sie können sie direkt weitergeben.

> **Note**
>
> Wenn Sie das Stammzertifikat zuvor importiert haben, überspringen Sie diesen Schritt. Verwenden Sie für ZwischenzertifikateTrustedCertificatePublicKey.

4. Exportieren Sie Ihren Schlüssel

Rufen Sie die ExportKey API mit `KeyMaterialType` set to auf auf auf aufTR34_KEY_BLOCK. Sie müssen Folgendes angeben:

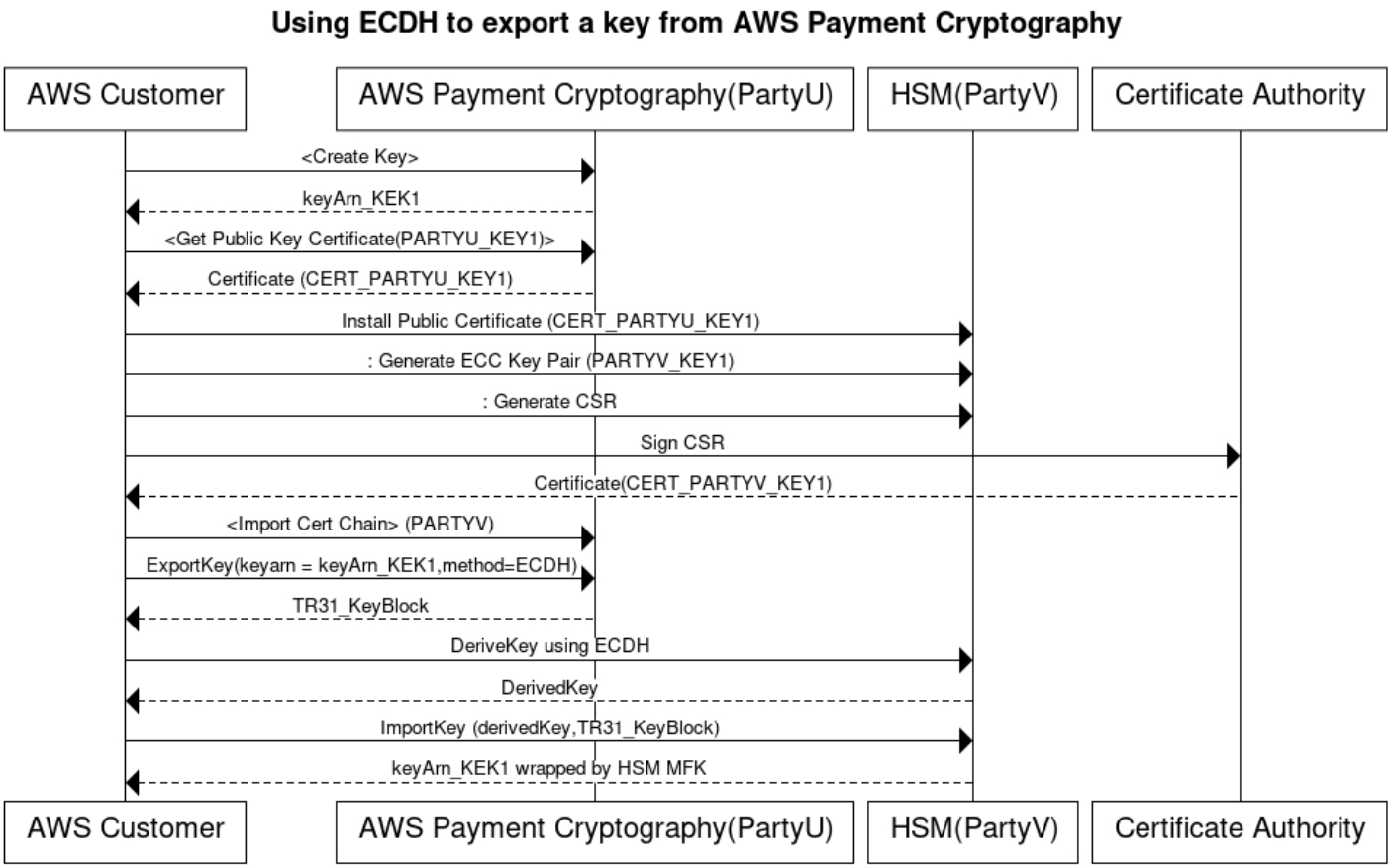
- Der KeyArn der Root-CA aus Schritt 3 als `CertificateAuthorityPublicKeyIdentifier`
- Das Blattzertifikat aus Schritt 3 als `WrappingKeyCertificate`
- Der KeyArn (oder Alias) des Schlüssels, den Sie exportieren möchten `--export-key-identifier`
- Das Export-Token aus Schritt 1

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"
  }
}
```

Exportieren Sie Schlüssel mit asymmetrischen Techniken (ECDH)



Elliptic Curve Diffie-Hellman (ECDH) verwendet die asymmetrische ECC-Kryptografie, um einen gemeinsamen Schlüssel zwischen zwei Parteien zu erstellen, ohne dass zuvor Schlüssel ausgetauscht werden müssen. ECDH-Schlüssel sind kurzlebig, weshalb Payment Cryptography sie nicht speichert. AWS Bei diesem Prozess wird mithilfe von ECDH ein einmaliges Passwort KBPK/KEK abgeleitet. Dieser abgeleitete Schlüssel wird sofort zum Umschließen des Schlüssels verwendet, den Sie übertragen möchten. Dabei kann es sich um einen anderen KBPK-, BDK-, IPEK-Schlüssel oder andere Schlüsseltypen handeln.

Beim Export wird AWS Payment Cryptography als Partei U (Initiator) und das Empfangssystem als Partei V (Responder) bezeichnet.

> **Note**
>
> ECDH kann verwendet werden, um jeden symmetrischen Schlüsseltyp auszutauschen, aber es ist der einzige Ansatz, der für die Übertragung von AES-256 Schlüsseln verwendet werden kann, wenn noch kein KEK eingerichtet ist.

1. Generieren Sie ein ECC-Schlüsselpaar

   Rufen Sie `create-key` auf, um ein ECC-Schlüsselpaar für diesen Prozess zu erstellen. Diese API generiert ein Schlüsselpaar für Schlüsselimporte oder -exporte. Geben Sie bei der Erstellung an, welche Art von Schlüsseln mit diesem ECC-Schlüssel abgeleitet werden können. Wenn Sie ECDH zum Austauschen (Umschließen) anderer Schlüssel verwenden, verwenden Sie den Wert. `TR31_K1_KEY_BLOCK_PROTECTION_KEY`

   > **Note**
   >
   > Obwohl Low-Level-ECDH einen abgeleiteten Schlüssel generiert, der für jeden Zweck verwendet werden kann, begrenzt AWS Payment Cryptography die versehentliche Wiederverwendung eines Schlüssels für mehrere Zwecke, indem ein Schlüssel nur für einen einzigen abgeleiteten Schlüsseltyp verwendet werden kann.

```
$ 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. Holen Sie sich das Zertifikat für einen öffentlichen Schlüssel

   Rufen Sie an`get-public-key-certificate`, um den öffentlichen Schlüssel als X.509 Zertifikat zu erhalten, das von der CA Ihres Kontos signiert wurde und spezifisch für AWS Zahlungskryptografie in einer bestimmten Region ist.

   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. Installieren Sie das öffentliche Zertifikat auf dem System der Gegenpartei (Partei V)

   Bei vielen HSMs müssen Sie das in Schritt 1 generierte öffentliche Zertifikat installieren, laden oder als vertrauenswürdig einstufen, um Schlüssel einzurichten. Dies kann je nach HSM die gesamte Zertifikatskette oder nur das Stammzertifikat umfassen. Spezifische Anweisungen finden Sie in Ihrer HSM-Dokumentation.

4. Generieren Sie das ECC-Schlüsselpaar auf dem Quellsystem und stellen Sie die Zertifikatskette bereit für AWS Kryptografie im Zahlungsverkehr

   Bei ECDH generiert jede Partei ein Schlüsselpaar und einigt sich auf einen gemeinsamen Schlüssel. Damit AWS Payment Cryptography den Schlüssel ableiten kann, benötigt es den öffentlichen Schlüssel der Gegenpartei im X.509 öffentlichen Schlüsselformat.

Wenn Sie Schlüssel von einem HSM übertragen, erstellen Sie ein Schlüsselpaar auf diesem HSM. Bei HSMs, die Schlüsselblöcke unterstützen, sieht der Schlüsselheader ähnlich aus wie. `D0144K3EX00E0000` Beim Erstellen des Zertifikats generieren Sie in der Regel eine CSR auf dem HSM, und dann AWS Private CA kann das HSM, ein Drittanbieter oder ein Dienst wie dieser das Zertifikat generieren.

Laden Sie das Stammzertifikat mithilfe des `importKey` Befehls mit KeyMaterialType of und of in AWS Payment Cryptography. `RootCertificatePublicKey` KeyUsageType `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`

Verwenden Sie für Zwischenzertifikate den `importKey` Befehl mit KeyMaterialType of `TrustedCertificatePublicKey` und KeyUsageType of`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`. Wiederholen Sie diesen Vorgang für mehrere Zwischenzertifikate. Verwenden Sie das Zertifikat `KeyArn` des letzten importierten Zertifikats in der Kette als Eingabe für nachfolgende Exportbefehle.

> **Note**
>
> Importieren Sie das Leaf-Zertifikat nicht. Geben Sie es direkt während des Exportbefehls an.

5. Schlüssel und Exportschlüssel ableiten von AWS Kryptografie im Zahlungsverkehr

Beim Exportieren leitet der Dienst mithilfe von ECDH einen Schlüssel ab und verwendet ihn dann sofort als KBPK, um den zu exportierenden Schlüssel zu umschließen. TR-31 Bei dem zu exportierenden Schlüssel kann es sich um einen beliebigen TDES- oder AES-Schlüssel handeln, sofern es sich um TR-31 gültige Kombinationen handelt, sofern der umschließende Schlüssel mindestens so stark ist wie der zu exportierende Schlüssel.

```
$ 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. Leiten Sie mithilfe von ECDH auf Party V-HSM einen einmaligen Schlüssel ab

   Viele HSMs und verwandte Systeme unterstützen die Erstellung von Schlüsseln mithilfe von ECDH. Geben Sie den öffentlichen Schlüssel aus Schritt 1 als öffentlichen Schlüssel und den Schlüssel aus Schritt 3 als privaten Schlüssel an. Zulässige Optionen, z. B. Ableitungsmethoden, finden Sie im API-Leitfaden.

   > **Note**
   >
   > Die Ableitungsparameter wie der Hash-Typ müssen auf beiden Seiten exakt übereinstimmen. Andernfalls generieren Sie einen anderen Schlüssel.

7. Schlüssel in das Zielsystem importieren

   Importieren Sie abschließend den Schlüssel aus AWS Payment Cryptography mithilfe von TR-31 Standardbefehlen. Geben Sie den von ECDH abgeleiteten Schlüssel als KBPK an und verwenden Sie den TR-31 Schlüsselblock, der zuvor aus AWS Payment Cryptography exportiert wurde.

Exportieren Sie Schlüssel mithilfe asymmetrischer Techniken (RSA Wrap)

Wenn RSA nicht TR-34 verfügbar ist, können Sie RSA wrap/unwrap für den Schlüsselaustausch verwenden. Ebenso TR-34 verwendet diese Methode asymmetrische RSA-Kryptografie, um symmetrische Schlüssel zu verschlüsseln. RSA Wrap beinhaltet jedoch nicht:

- Signierung der Nutzdaten durch den Absender
- Schlüsselblöcke, die die Integrität der wichtigsten Metadaten während des Transports aufrechterhalten

> **Note**
>
> Sie können RSA Wrap verwenden, um TDES und AES-128 Schlüssel zu exportieren.

1. Erstellen Sie einen RSA-Schlüssel und ein Zertifikat auf Ihrem Empfangssystem

   Erstellen oder identifizieren Sie einen RSA-Schlüssel für den Empfang des verpackten Schlüssels. Die Schlüssel müssen im X.509 Zertifikatsformat vorliegen. Stellen Sie sicher, dass das Zertifikat von einem Stammzertifikat signiert ist, das Sie in AWS Payment Cryptography importieren können.
2. Importieren Sie das öffentliche Stammzertifikat in AWS Kryptografie im Zahlungsverkehr

   Verwenden Sie es import-key mit der `--key-material` Option, das Zertifikat zu importieren

```
$ 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. Exportieren Sie Ihren Schlüssel

   Weisen Sie AWS Payment Cryptography an, Ihren Schlüssel mithilfe Ihres Leaf-Zertifikats zu exportieren. Sie müssen Folgendes angeben:

   - Der ARN für das Stammzertifikat, das Sie in Schritt 2 importiert haben
   - Das Blattzertifikat für den Export
   - Der symmetrische Schlüssel für den Export

   Die Ausgabe ist eine hexadezimale, binär verpackte (verschlüsselte) Version Ihres symmetrischen Schlüssels.

Example Beispiel — Exportieren eines Schlüssels

```
$ 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. Importieren Sie den Schlüssel in Ihr Empfangssystem

   Viele HSMs und verwandte Systeme unterstützen den Import von Schlüsseln mithilfe von RSA Unwrap (einschließlich AWS Payment Cryptography). Geben Sie beim Import Folgendes an:

   - Der öffentliche Schlüssel aus Schritt 1 als Verschlüsselungszertifikat
   - Das Format als RSA
   - Padding-Modus als PKCS #1 v2.2 OAEP (mit SHA 256)

> **Note**
>
> Wir geben den verpackten Schlüssel im HexBinary-Format aus. Möglicherweise müssen Sie das Format konvertieren, wenn Ihr System eine andere Binärdarstellung benötigt, z. B. base64.

Exportieren Sie symmetrische Schlüssel mithilfe eines vorab festgelegten Schlüsselaustauschschlüssels () TR-31

Beim Austausch mehrerer Schlüssel oder bei der Unterstützung der Schlüsselrotation tauschen die Partner in der Regel zuerst einen anfänglichen Schlüsselverschlüsselungsschlüssel (KEK) aus. Sie können KEK gegen AWS Payment Cryptography austauschen, indem Sie Techniken wie TR-34 oder Physischer Schlüsselaustausch verwenden. Nachdem Sie einen KEK eingerichtet haben, können Sie ihn verwenden, um nachfolgende Schlüssel, einschließlich anderer KEKs, zu transportieren. Wir unterstützen diesen Schlüsselaustausch mithilfe von ANSI TR-31, das von HSM-Anbietern weithin unterstützt wird.

1. Richten Sie Ihren Key Encryption Key (KEK) ein

   Stellen Sie sicher, dass Sie Ihren KEK bereits ausgetauscht haben und den KeyArn (oder KeyAlias) verfügbar haben.

2. Erstellen Sie Ihren Schlüssel auf AWS Kryptografie im Zahlungsverkehr

   Erstellen Sie Ihren Schlüssel, falls er noch nicht existiert. Alternativ können Sie den Schlüssel auf Ihrem anderen System erstellen und den ??? Import-Befehl verwenden.

3. Exportieren Sie Ihren Schlüssel von AWS Kryptografie im Zahlungsverkehr

   Geben Sie beim Exportieren im TR-31 Format den Schlüssel an, den Sie exportieren möchten, und den zu verwendenden Wrapping-Schlüssel.

Example Beispiel — Exportieren eines Schlüssels mit dem TR31-Schlüsselblock

```
$ 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. Importieren Sie den Schlüssel in Ihr System

   Verwenden Sie die Importschlüsselimplementierung Ihres Systems, um den Schlüssel zu importieren.

Exportieren Sie die DUPT-Anfangsschlüssel () IPEK/IK

Wenn Sie DUKPT verwenden, können Sie einen einzelnen Basisableitungsschlüssel (BDK) für eine Flotte von Terminals generieren. Die Terminals haben keinen direkten Zugang zum BDK. Stattdessen erhält jedes Terminal einen eindeutigen ersten Terminalschlüssel, der als IPEK oder Initial Key (IK) bezeichnet wird. Jedes IPEK wird mithilfe einer eindeutigen Schlüsselseriennummer (KSN) vom BDK abgeleitet.

Die KSN-Struktur variiert je nach Verschlüsselungstyp:

- Für TDES: Das 10-Byte-KSN beinhaltet:
  - 24 Bit für die Schlüsselsatz-ID
  - 19 Bit für die Terminal-ID
  - 21 Bit für den Transaktionszähler

- Für AES: Das 12-Byte-KSN beinhaltet:
  - 32 Bit für die BDK-ID
  - 32 Bit für den Ableitungsbezeichner (ID)
  - 32 Bit für den Transaktionszähler

Wir bieten einen Mechanismus zum Generieren und Exportieren dieser Anfangsschlüssel. Sie können die generierten Schlüssel mithilfe der Methoden TR-31, TR-34, oder RSA Wrap exportieren. Beachten Sie, dass IPEK-Schlüssel nicht persistent sind und nicht für nachfolgende Operationen mit Payment Cryptography verwendet werden können. AWS

Wir erzwingen die Aufteilung zwischen den ersten beiden Teilen des KSN nicht. Wenn Sie die Ableitungs-ID zusammen mit dem BDK speichern möchten, können Sie Tags verwenden. AWS

> **Note**
>
> Der Zählerteil des KSN (32 Bit für AES DUKPT) wird nicht für die Ableitung verwendet. IPEK/IK Beispielsweise erzeugen Eingaben von 12345678901234560001 und 12345678901234569999 dasselbe IPEK.

```
$ 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"
}
}
```

Geben Sie Schlüsselblock-Header für den Export an

Sie können Schlüsselblockinformationen ändern oder anhängen, wenn Sie in ASC TR-31 oder Formaten exportieren. TR-34 In der folgenden Tabelle werden das TR-31 Schlüsselblockformat und die Elemente beschrieben, die Sie beim Export ändern können.

Schlüsselblock-Attribut	Zweck	Können Sie während des Exports Änderungen vornehmen?	Hinweise
Versions-ID	Definiert die Methode, die zum Schutz des Schlüsselmaterials verwendet wird. Der Standard beinhaltet: • Version A und C (wichtige Variante — veraltet) • Version B (Ableitung mit TDES) • Version D (Schlüsselableitung mit AES)	Nein	Wir verwenden Version B für TDES-Wrapping Keys und Version D für AES-Wrapping Keys. Wir unterstützen die Versionen A und C nur für Importvorgänge.
Länge des Schlüsselblocks	Gibt die Länge der verbleibenden Nachricht an	Nein	Wir berechnen diesen Wert automatisch. Die Länge erscheint möglicherweise falsch, bevor die Nutzlast entschlüsselt wird, da wir möglicherweise das in der Spezifikation vorgeschriebene

Schlüsselblock-Attribut	Zweck	Können Sie während des Exports Änderungen vornehmen?	Hinweise
			Schlüsselpadding hinzufügen.
Schlüsselverwendung	Definiert die zulässigen Zwecke für den Schlüssel, z. B.: • C0 (Kartenverifizierung) • B0 (Basisableitungsschlüssel)	Nein	
Algorithmus	Gibt den Algorithmus des zugrunde liegenden Schlüssels an. Wir unterstützen: • (TDES) • (HMAC) • (AES)	Nein	Wir exportieren diesen Wert unverändert.
Schlüsselverwendung	Definiert zulässige Operationen, wie zum Beispiel: • Generieren und verifizieren (C) • Encrypt/Decrypt/Wrap/Unwrap (B)	Ja*	

Schlüsselblock-Attribut	Zweck	Können Sie während des Exports Änderungen vornehmen?	Hinweise
Schlüsselversion	Gibt die Versionsnummer für den Schlüssel an replacement/rotation. Der Standardwert ist 00, wenn nicht angegeben.	Ja — Kann anhängen	
Wichtige Exportfähigkeit	Steuert, ob der Schlüssel exportiert werden kann: • N — Keine Exportfähigkeit • E - Export gemäß X9.24 (Schlüsselblöcke) • S — Export im Schlüsselblock- oder Nichtschlüsselblockformat	Ja*	

Schlüsselblock-Attribut	Zweck	Können Sie während des Exports Änderungen vornehmen?	Hinweise
Optionale Tastenblöcke	Ja — Kann anhängen	Optionale Schlüsselblöcke sind name/value Paare, die kryptografisch an den Schlüssel gebunden sind. Zum Beispiel KeySet ID für DUKPT-Schlüssel. Wir berechnen automatisch die Anzahl der Blöcke, die Länge jedes Blocks und den Füllblock (PB) auf der Grundlage Ihrer name/value Paareingabe.	

*Wenn Sie Werte ändern, muss Ihr neuer Wert restriktiver sein als der aktuelle Wert in AWS Payment Cryptography. Beispiel:

- Wenn der aktuelle Tastenverwendungsmodus Generate=True, Verify=True ist, können Sie ihn in Generate=True, Verify=False ändern
- Wenn der Schlüssel bereits auf „Nicht exportierbar“ gesetzt ist, können Sie ihn nicht in „exportierbar“ ändern

Wenn Sie Schlüssel exportieren, wenden wir automatisch die aktuellen Werte des exportierten Schlüssels an. Möglicherweise möchten Sie diese Werte jedoch ändern oder anhängen, bevor Sie sie an das empfangende System senden. Hier sind einige gängige Szenarien:

- Wenn Sie einen Schlüssel in ein Zahlungsterminal exportieren, stellen Sie die Exportfähigkeit auf ein, `Not Exportable` da Terminals normalerweise nur Schlüssel importieren und diese nicht exportieren sollten.
- Wenn Sie zugehörige Schlüsselmetadaten an das Empfangssystem weitergeben müssen, verwenden Sie TR-31 optionale Header, um die Metadaten kryptografisch an den Schlüssel zu binden, anstatt eine benutzerdefinierte Nutzlast zu erstellen.
- Stellen Sie die Schlüsselversion mithilfe des `KeyVersion` Felds ein, um die Schlüsselrotation zu verfolgen.

TR-31/X9.143 definiert allgemeine Header, Sie können jedoch auch andere Header verwenden, sofern sie den AWS Payment Cryptography-Parametern entsprechen und Ihr Empfangssystem sie akzeptiert. Weitere Informationen zu Schlüsselblock-Headern beim Export finden Sie im API-Handbuch unter Schlüsselblock-Header.

Hier ist ein Beispiel für den Export eines BDK-Schlüssels (z. B. in ein KIF) mit diesen Spezifikationen:

- Schlüsselversion: 02
- KeyExportability: NICHT EXPORTIERBAR
- KeySetID: 00ABCDEFAB (00 gibt den TDES-Schlüssel an, ABCDEFABCD ist der Anfangsschlüssel)

Da wir keine Hauptverwendungsmodi angeben, erbt dieser Schlüssel den Verwendungsmodus von arn:aws:payment-cryptography:us-east- 2:111122223333: (= true). key/5rplquuwozodpwsp DeriveKey

> **Note**
>
> Selbst wenn Sie in diesem Beispiel die Exportfähigkeit auf Nicht exportierbar setzen, kann das KIF immer noch: ???
>
> - Schlüssel ableiten, wie sie beispielsweise in DUKPT verwendet werden IPEK/IK
> - Exportieren Sie diese abgeleiteten Schlüssel, um sie auf Geräten zu installieren
>
> Dies ist in den Standards ausdrücklich zulässig.

```
$ 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"
    }
}
```

Allgemeine Header

X9.143 definiert bestimmte Header für häufige Anwendungsfälle. Mit Ausnahme des HM-Headers (HMAC Hash) analysiert oder verwendet AWS Payment Cryptography diese Header nicht.

Header-Name	Zweck	Typische Validierung	Hinweise
BI	Schlüssel-ID für die Basisableitung für DUKPT	2 Hex-Zeichen (00 für TDES, 11 für AES), dann 10 Hex-Zeichen für TDES KSI oder 8 Hex-Zeichen für BDK-ID (AES DUKPT).	Enthält die (BDK-ID, für AES DUKPT) oder den Key Set Identifier (KSI, für TDES DUKPT). Kann beim Austausch der BDK-ID oder des KSI verwendet werden, muss aber nicht die

Header-Name	Zweck	Typische Validierung	Hinweise
			anderen in den IK- und KS-Blöcken enthaltenen Daten austauschen. In der Regel wird BI für die Übertragung an ein KIF verwendet, während IK oder KS für die Übertragung in das Terminal selbst verwendet werden.
HM	Gibt den Hashtyp für HMAC-Operationen an	• 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 — SCHÜTTELN (128) • 41 — SCHÜTTELN 256	Der Dienst füllt dieses Feld beim Export automatisch aus und analysiert es beim Import. Hashtypen, die vom Dienst nicht unterstützt werden, wie SHAKE128, können importiert werden, sind aber möglicherweise nicht für kryptografische Funktionen nutzbar.

Header-Name	Zweck	Typische Validierung	Hinweise
IK	Seriennummer des Anfangsschlüssels für AES DUKPT	16 Hex-Zeichen	Dieser Wert wird verwendet, um die Verwendung des Initial-DUKPT-Schl üssels auf dem Empfangsgerät zu instanziieren, und er identifiziert den von einem BDK abgeleite ten Initialschlüssel. Dieses Feld enthält normalerweise die Ableitungsdaten, aber keinen Zähler. Verwenden Sie KS für TDES DUKPT.
IST	Anfangsschlüssel-Seriennummer für TDES DUKPT	20 Hex-Zeichen	Dieser Wert wird verwendet, um die Verwendung des Initial DUKPT-Sch lüssels auf dem Empfangsgerät zu instanziieren, und er identifiziert den Initial Key, der von einem BDK abgeleite t wurde. Dieses Feld enthält in der Regel die Ableitungsdaten plus einen auf Null gesetzten Zählerwert. Verwenden Sie IK für AES DUKPT.

Header-Name	Zweck	Typische Validierung	Hinweise
KP	Schlüsselprüfwert des KBPK ???	2 Hex-Zeichen stehen für die KCV-Metho de (00 für die X9.24 Methode und 01 für die CMAC-Methode). Gefolgt vom KCV-Wert, der normalerw eise aus 6 Hex-Zeichen besteht. Zum Beispiel steht 010FA329 für den KCV von 0FA329, berechnet mit der 01 (CMAC) -Methode.	Berechnet gemäß ANSI X9.24-1-2017 Anhang A; der KCV wird für die Schlüssel validierung verwendet , nicht als Integritä tsmechanismus.
PB	Polsterblock	zufällige druckbare ASCII-Zeichen	Der Dienst füllt dieses Feld beim Export automatisch aus, um sicherzustellen, dass optionale Header ein Vielfaches der Verschlüsselungsbl ocklänge sind

Exportiert asymmetrische (RSA) Schlüssel

Verwenden Sie den Befehl, um einen öffentlichen Schlüssel in Zertifikatsform zu exportieren. get-public-key-certificate Dieser Befehl gibt Folgendes zurück:

- Das Zertifikat
- Das Stammzertifikat

Beide Zertifikate sind Base64-kodiert.

> **Note**
>
> Dieser Vorgang ist nicht impotent — nachfolgende Aufrufe können unterschiedliche Zertifikate generieren, selbst wenn derselbe zugrunde liegende Schlüssel verwendet wird.

Example

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

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

Fortgeschrittene Themen

In diesem Abschnitt werden erweiterte Szenarien und Konfigurationen für den Schlüsselaustausch behandelt.

Themen

- Bringen Sie Ihre eigene Zertifizierungsstelle (BYOCA) mit
- Physischer Schlüsselaustausch

Bringen Sie Ihre eigene Zertifizierungsstelle (BYOCA) mit

Wenn ein Public-Key-Zertifikat für asymmetrische (RSA, ECC) Schlüssel benötigt wird, die innerhalb des Dienstes erstellt wurden, werden diese Zertifikate standardmäßig von einer Zertifizierungsstelle (CA) für AWS Zahlungskryptografie und Kontoinformationen ausgestellt. Dies soll die Verwendung vereinfachen, X.509 ohne dass Sie eine CA identifizieren oder einrichten oder Certificate Signing Requests (CSR) verwalten müssen.

AWS Payment Cryptography bietet auch die Möglichkeit, Ihre eigene CA zu verwenden, wenn dies aus politischen oder Compliance-Gründen erforderlich ist.

Themen

- -Übersicht
- BYOCA Export-Arbeitsablauf
- BYOCA Import-Arbeitsablauf
- Weitere Hinweise

-Übersicht

Mit BYOCA können Sie Ihre eigene CA für RSA Unwrap TR-34 import/export - und ECDH-Schlüsselübertragungen verwenden. Verwenden Sie BYOCA, wenn Sie eine konsistente Zertifikatskette in Ihrem Unternehmen benötigen oder wenn Partner bestimmte CA-Zertifikate benötigen. Die folgenden Beispiele zeigen den BYOCA-Workflow für TR-34 Schlüsselexport und Schlüsselimport. TR-34

Es gibt drei wesentliche Unterschiede zum TR-34 Standardablauf:

1. Sie erstellen den RSA-Schlüssel mit CreateKey. Vorher, `GetParametersForExport` oder `GetParametersForImport` hat ihn für Sie erstellt.
2. Die GetCertificateSigningRequest API erstellt eine CSR. Ihre externe CA kann es dann signieren.
3. Die ExportKey und ImportKey APIs akzeptieren ein Zertifikat zum Zeitpunkt des Anrufs. Das Token ist jetzt optional.

⚠ Wichtige Überlegungen

- In diesen Beispielen werden RSA-2048 Schlüssel verwendet und ein TDES-2KEY Schlüssel wird umschlossen. Stellen Sie beim Exportieren sicher AES-128, dass alle Schlüssel RSA-3072 oder sind RSA-4096.
- Der häufigste Fehler ist, dass die durch `SigningKeyIdentifier` und dargestellten Schlüssel `SigningKeyCertificate` nicht übereinstimmen.

BYOCA Export-Arbeitsablauf

Die folgenden Schritte veranschaulichen den vollständigen BYOCA-Workflow für den Export. TR-34

Schritte

- Schritt 1: RSA-Schlüssel erstellen
- Schritt 2: Generieren Sie eine Anfrage zur Zertifikatsignierung
- Schritt 3: CSR überprüfen (optional)
- Schritt 4: Signieren Sie den CSR bei einer Zertifizierungsstelle
- Schritt 5: CA-Zertifikat importieren
- Schritt 6: Holen Sie sich das KRD-Verschlüsselungszertifikat
- Schritt 7: Schlüssel mit BYOCA exportieren

Schritt 1: RSA-Schlüssel erstellen

Erstellen Sie zunächst ein RSA-Schlüsselpaar, das letztendlich das KDH-Signaturzertifikat sein wird. Sie können Tags hinzufügen, um den Zweck des Schlüssels zu identifizieren.

Example Erstellen Sie einen RSA-Schlüssel zum Signieren

```
$ 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"
    }
}
```

Notieren Sie sich den, `KeyArn` da Sie ihn im nächsten Schritt benötigen.

Schritt 2: Generieren Sie eine Anfrage zur Zertifikatsignierung

Generieren Sie mithilfe der GetCertificateSigningRequest API eine Zertifikatsignieranforderung (CSR), die von Ihrer externen CA signiert werden soll. Die Ausgabe ist eine Base64-kodierte PEM-Datei. Wenn Sie den Inhalt mit Base64 dekodieren und speichern, haben Sie eine gültige CSR im PEM-Format.

Example Generieren Sie CSR

```
$ aws payment-cryptography 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..."
}
```

Das `CertificateSigningRequest` Feld enthält die gesamte PEM-Datei, base64-kodiert. Die PEM-Datei enthält die Markierungen und. `-----BEGIN CERTIFICATE REQUEST-----` `-----END CERTIFICATE REQUEST-----` Senden Sie diesen Wert zur Unterzeichnung an Ihre CA.

Schritt 3: CSR überprüfen (optional)

Sie können optional OpenSSL verwenden, um den CSR-Inhalt zu überprüfen und sicherzustellen, dass er gültig ist und den Erwartungen entspricht.

ExampleÜberprüfen Sie CSR mit OpenSSL

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

Schritt 4: Signieren Sie den CSR bei einer Zertifizierungsstelle

Nachdem Sie den CSR generiert haben, müssen Sie ihn von einer Zertifizierungsstelle (CA) signieren lassen. In Produktionsumgebungen verwenden AWS Private CA Sie in der Regel die etablierte CA-Infrastruktur Ihres Unternehmens. Zu Testzwecken können Sie OpenSSL verwenden, um ein selbstsigniertes Zertifikat zu erstellen.

Verwenden AWS Private CA

Um den CSR mit zu signieren AWS Private CA, dekodieren Sie zuerst den Base64-kodierten CSR und speichern Sie ihn in einer Datei. Verwenden Sie dann die API. IssueCertificate

Example Signieren Sie CSR mit 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"
}
```

Rufen Sie dann das signierte Zertifikat ab:

Example Signiertes Zertifikat abrufen

```
$ 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-----"
```

```
}
```

Speichern Sie den Zertifikatsinhalt zur Verwendung im Exportschritt. Sie müssen es Base64-kodieren, wenn Sie es der API zur Verfügung stellen. `ExportKey`

Verwenden von OpenSSL zum Testen

Zu Testzwecken können Sie OpenSSL verwenden, um eine selbstsignierte CA zu erstellen und die CSR zu signieren. Erstellen Sie zunächst einen privaten CA-Schlüssel und ein selbstsigniertes Zertifikat:

Example Erstellen Sie eine Test-CA mit OpenSSL

```
$ # 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"
```

Dekodieren Sie dann den CSR aus dem vorherigen Schritt und signieren Sie ihn mit Ihrer Test-CA:

Example Signieren Sie CSR mit OpenSSL

```
$ # 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
```

Das signierte Zertifikat ist jetzt da. `signed-cert.pem` Sie müssen dieses Zertifikat base64-kodieren, wenn Sie es der API zur Verfügung stellen: `ExportKey`

Example Base64 Codieren Sie das signierte Zertifikat

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

Schritt 5: CA-Zertifikat importieren

Jede verwendete CA muss zuerst als vertrauenswürdig eingestuft werden, um zu verhindern, dass willkürliche Zertifikate verwendet werden. Importieren Sie das Stammzertifikat Ihrer externen CA mithilfe der ImportKey API. Wenn Sie eine Zwischenzertifizierungsstelle verwenden, rufen Sie `import-key` erneut auf, geben Sie jedoch `TrustedPublicKey` anstelle von an `RootCertificatePublicKey` und geben Sie den ARN der Stammzertifizierungsstelle an.

Example Root-CA-Zertifikat importieren

```
$ 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"
    }
}
```

Notieren Sie sich die Zertifizierungsstellen, `KeyArn` die Sie im Exportschritt verwenden können.

Schritt 6: Holen Sie sich das KRD-Verschlüsselungszertifikat

In diesem Beispiel importieren wir zurück in AWS Payment Cryptography, also rufen wir den Dienst auf, um mithilfe der API ein KRD-Zertifikat für öffentliche Schlüssel zu erhalten. GetParametersForImport In einem realen Szenario würde dies vom anderen System bereitgestellt werden, z. B. von einem HSM, einem Geldautomaten, einem Zahlungsterminal oder einem Zahlungsterminal-Managementsystem.

Example Parameter für den Import abrufen

```
$ aws payment-cryptography 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"
}
```

Schritt 7: Schlüssel mit BYOCA exportieren

Exportieren Sie den Schlüssel abschließend TR-34 mit Ihrem eigenen CA-signed Zertifikat mithilfe der ExportKey API. Geben Sie das Signaturzertifikat an, das von Ihrer externen CA signiert wurde.

Example TR-34 Exportieren Sie mit BYOCA

```
$ aws payment-cryptography 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"
    }
}
```

Der exportierte Schlüsselblock kann jetzt vom Empfangssystem mithilfe des TR-34 Standard-Importprozesses importiert werden.

BYOCA Import-Arbeitsablauf

Die folgenden Schritte führen Sie durch den BYOCA-Workflow für den Schlüsselimport. TR-34 Verwenden Sie diesen Workflow, um einen Schlüssel für AWS Payment Cryptography mit Ihrem eigenen CA-signed KRD-Verschlüsselungszertifikat (Key Receiving Device) zu erhalten. Beim BYOCA-Import kontrollieren Sie die Vertrauenskette, indem Sie eine CA verwenden, die Sie bereits verwalten.

Schritte

- Schritt 1: RSA-Schlüssel erstellen
- Schritt 2: Generieren Sie eine Anfrage zur Zertifikatsignierung
- Schritt 3: CSR überprüfen (optional)
- Schritt 4: Signieren Sie den CSR bei einer Zertifizierungsstelle
- Schritt 5: CA-Zertifikat importieren
- Schritt 6: Holen Sie sich das KDH-Signaturzertifikat vom sendenden System
- Schritt 7: Schlüssel mit BYOCA importieren

Schritt 1: RSA-Schlüssel erstellen

Erstellen Sie ein RSA-Schlüsselpaar, das als KRD-Verschlüsselungsschlüssel verwendet werden soll. Dieser Schlüssel entpackt und entschlüsselt den eingehenden Schlüssel beim Import. TR-34 Der Schlüssel kann nicht exportiert werden, da er als Empfangsschlüssel im Dienst verbleibt.

Example Erstellen Sie den KRD-Verschlüsselungsschlüssel

```
$ aws payment-cryptography create-key \
    --key-attributes
 KeyAlgorithm=RSA_2048,KeyUsage=TR31_K2_TR34_ASYMMETRIC_KEY,KeyClass=ASYMMETRIC_KEY_PAIR,KeyMod
```

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
bm7t4qv8hzk24jce",
        "KeyAttributes": {
            "KeyUsage": "TR31_K2_TR34_ASYMMETRIC_KEY",
            "KeyClass": "ASYMMETRIC_KEY_PAIR",
            "KeyAlgorithm": "RSA_2048",
            "KeyModesOfUse": {
                "Unwrap": true,
                "Decrypt": true
            }
        },
        "KeyCheckValue": "7FA29C1E",
        "KeyCheckValueAlgorithm": "SHA_1",
        "Enabled": true,
        "Exportable": false,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY"
    }
}
```

Notieren Sie sich den `KeyArn`-Wert. Sie benötigen ihn im nächsten Schritt.

Schritt 2: Generieren Sie eine Anfrage zur Zertifikatsignierung

Generieren Sie mithilfe der API eine Zertifikatsignieranforderung (CSR) für den KRD-Verschlüsselungsschlüssel. GetCertificateSigningRequest Die Ausgabe ist eine Base64-kodierte PEM-Datei. Wenn Sie den Inhalt mit Base64 dekodieren und speichern, haben Sie eine gültige CSR im PEM-Format.

Example Generieren Sie CSR für den KRD-Schlüssel

```
$ aws payment-cryptography get-certificate-signing-request \
    --key-identifier arn:aws:payment-cryptography:us-east-1:111122223333:key/
bm7t4qv8hzk24jce \
```

```
    --signing-algorithm SHA512 \
    --certificate-subject '{
        "CommonName": "MyImportKRDCertAWSUSEAST",
        "Organization": "Amazon",
        "OrganizationUnit": "PaymentCryptography",
        "Country": "US",
        "StateOrProvince": "Virginia",
        "City": "Arlington"
    }'
```

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

Das `CertificateSigningRequest` Feld enthält die gesamte PEM-Datei, base64-kodiert. Die PEM-Datei enthält die Markierungen und. `-----BEGIN CERTIFICATE REQUEST-----` `-----END CERTIFICATE REQUEST-----` Senden Sie diesen Wert zur Unterzeichnung an Ihre CA.

Schritt 3: CSR überprüfen (optional)

Sie können optional OpenSSL verwenden, um den CSR-Inhalt zu überprüfen und sicherzustellen, dass er gültig ist und den Erwartungen entspricht.

ExampleÜberprüfen Sie CSR mit OpenSSL

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

Schritt 4: Signieren Sie den CSR bei einer Zertifizierungsstelle

Nachdem Sie die CSR generiert haben, lassen Sie sie von einer Zertifizierungsstelle (CA) signieren. In Produktionsumgebungen verwenden AWS Private CA Sie in der Regel die etablierte CA-Infrastruktur Ihres Unternehmens. Zu Testzwecken können Sie OpenSSL verwenden, um ein selbstsigniertes Zertifikat zu erstellen.

Verwenden AWS Private CA

Um den CSR mit zu signieren AWS Private CA, dekodieren Sie zuerst den Base64-kodierten CSR und speichern Sie ihn in einer Datei. Verwenden Sie dann die API. IssueCertificate

Example Signieren Sie CSR mit AWS Private CA

```
$ echo "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0..." | base64 -d > krd-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://krd-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/fedcba0987654321"
}
```

Rufen Sie dann das signierte Zertifikat ab:

Example Signiertes Zertifikat abrufen

```
$ 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/fedcba0987654321
```

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

Speichern Sie den Zertifikatsinhalt zur Verwendung im Importschritt. Base64-encode das Zertifikat, bevor Sie es der `ImportKey` API zur Verfügung stellen.

OpenSSL zum Testen verwenden

Zu Testzwecken können Sie OpenSSL verwenden, um eine selbstsignierte CA zu erstellen und die CSR zu signieren. Erstellen Sie zunächst einen privaten CA-Schlüssel und ein selbstsigniertes Zertifikat:

Example Erstellen Sie eine Test-CA mit OpenSSL

```
$ # 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"
```

Dekodieren Sie dann den CSR aus dem vorherigen Schritt und signieren Sie ihn mit Ihrer Test-CA:

Example CSR mit OpenSSL signieren

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

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

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

Das signierte Zertifikat ist jetzt da. `krd-signed-cert.pem` Base64-encode dieses Zertifikat, bevor Sie es der `ImportKey` API zur Verfügung stellen:

Example Base64 Codieren Sie das signierte Zertifikat

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

Schritt 5: CA-Zertifikat importieren

Bevor Sie eine CA verwenden, müssen Sie ihr zuerst vertrauen. Importieren Sie das Stammzertifikat Ihrer externen CA mithilfe der ImportKey API. Wenn Sie eine Zwischenzertifizierungsstelle verwenden, rufen Sie `import-key` erneut auf, geben Sie jedoch `TrustedPublicKey` anstelle von an `RootCertificatePublicKey` und geben Sie den ARN der Stammzertifizierungsstelle an.

Example Root-CA-Zertifikat importieren

```
$ 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/
nqr3c5v2wp6yl8az",
        "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"
    }
}
```

Notieren Sie sich die `KeyArn` Zertifizierungsstellen, die im Importschritt verwendet werden sollen.

Schritt 6: Holen Sie sich das KDH-Signaturzertifikat vom sendenden System

Bei einem TR-34 Schlüsselaustausch stellt die sendende Seite (Key Distribution Host oder KDH) ihr Public-Key-Zertifikat zum Signieren bereit. Dieses Zertifikat wird verwendet, um die Signatur auf dem verpackten Schlüsselblock während des Imports zu überprüfen und sicherzustellen, dass das Schlüsselmaterial von einem vertrauenswürdigen Absender stammt. Bei der Produktion wird dieses Zertifikat in der Regel vom sendenden System — wie einem HSM, einem Zahlungsterminal oder einem Schlüsselverwaltungssystem — über einen eigenen Prozess zur Zertifikatsverteilung bereitgestellt.

In diesem Beispiel verwenden Sie AWS Payment Cryptography sowohl als Absender als auch als Empfänger, um die Schritte kurz zu halten. Rufen Sie GetParametersForExport an, um das KDH-Signaturzertifikat zu erhalten, das normalerweise vom sendenden System bereitgestellt wird.

Example Parameter für den Export abrufen

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

```
{
    "ExportToken": "export-token-k8sfhe2w9rjy4bx6",
    "SigningKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...",
    "SigningKeyCertificateChain": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...",
    "SigningKeyAlgorithm": "RSA_2048",
    "ParametersValidUntilTimestamp": "2025-11-01T18:45:31.271000-07:00"
}
```

Beachten Sie das `ExportToken` und `SigningKeyCertificate` für die Verwendung im Importschritt.

Schritt 7: Schlüssel mit BYOCA importieren

Schließlich importieren Sie den Schlüssel TR-34 mithilfe Ihres eigenen CA-signed Zertifikats mithilfe der `ImportKey` API. Stellen Sie das KRD-Verschlüsselungszertifikat bereit, das von Ihrer externen CA signiert wurde.

Der `RandomNonce` Parameter ist erforderlich, wenn Sie TR-34 2-Pass verwenden. Der Service generiert keine Nonce für Sie — es ist ein zufälliger Hex-Wert, den Sie generieren, und er muss sowohl auf der Sende- als auch auf der Empfangsseite identisch sein. Wenn Sie 1-Pass verwenden TR-34 (das auf einem Zeitstempel statt einer Nonce basiert), darf der Zeitstempel nicht in der Zukunft liegen. Der Service validiert keine bestimmte Aktualität (z. B. innerhalb einer bestimmten Anzahl von Stunden), aber Sie können bei Bedarf Frischeprüfungen auf Ihrer Seite erzwingen.

Example TR-34 Import mit BYOCA

```
$ aws payment-cryptography import-key --key-material='{
    "Tr34KeyBlock": {
        "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-cryptography:us-
east-1:111122223333:key/nqr3c5v2wp6yl8az",
        "KeyBlockFormat": "X9_TR34_2012",
```

```
        "RandomNonce": "4997FBB4587D571F",
        "SigningKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...",
        "WrappingKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t...",
        "WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-
east-1:111122223333:key/bm7t4qv8hzk24jce",
        "WrappedKeyBlock": "D0112B0AX00E0000B82679114F470F540165EDFBF407..."
    }
}'
```

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
gwz7u3rfxd4o52sb",
        "KeyAttributes": {
            "KeyUsage": "TR31_K0_KEY_ENCRYPTION_KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyAlgorithm": "TDES_2KEY",
            "KeyModesOfUse": {
                "Encrypt": true,
                "Decrypt": true
            }
        },
        "KeyCheckValue": "3DCA31",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": true,
        "Exportable": true,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "EXTERNAL"
    }
}
```

Der Schlüssel ist jetzt in AWS Payment Cryptography importiert und einsatzbereit. Das `KeyOrigin` Feld zeigt an`EXTERNAL`, dass der Schlüssel von außerhalb des Dienstes stammt.

Weitere Hinweise

- Diese Beispiele werden mithilfe der AWS-CLI gezeigt. Dieselbe Funktionalität ist in allen AWS-SDKs verfügbar, einschließlich Java, Python, Go und Rust.
- Wenn Sie mit einer selbstsignierten CA testen, können Sie OpenSSL verwenden, um eine Test-CA zu erstellen und die CSR zu signieren. Verwenden Sie in der Produktion die etablierte CA-Infrastruktur Ihres Unternehmens.

Physischer Schlüsselaustausch

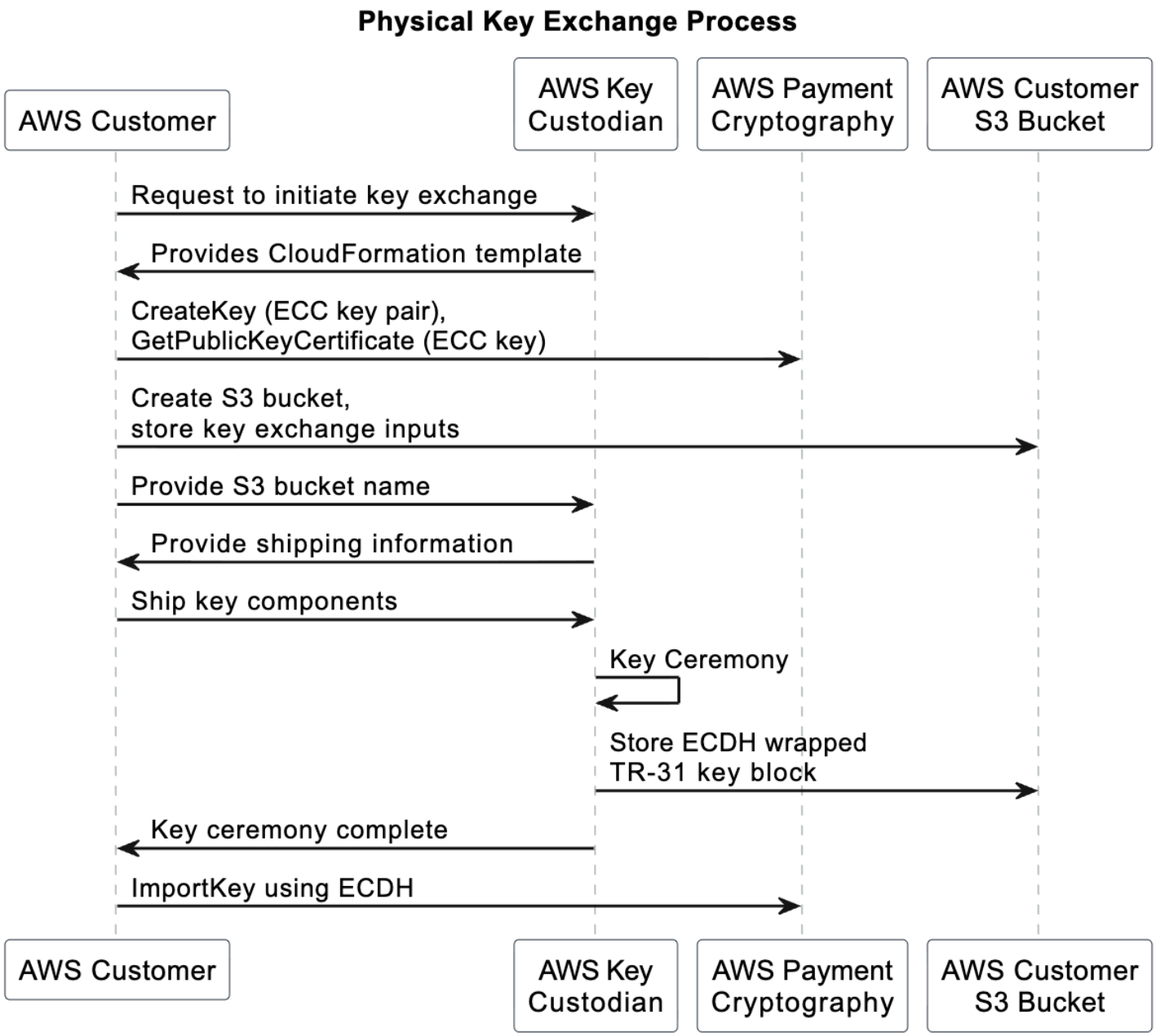
Sie können den physischen Schlüsselaustausch verwenden, um papierbasierte kryptografische Schlüsselkomponenten sicher in ein elektronisches Format zu konvertieren, wenn Ihre Partner oder Anbieter den elektronischen Schlüsselaustausch nicht unterstützen. Geschulte AWS Schlüsselverwalter führen wichtige Zeremonien in PCI-PIN- und P2PE-zertifizierten AWS-operated sicheren Einrichtungen durch und konvertieren wichtige Papierkomponenten mithilfe eines Offline-HSM in ein elektronisches Format. Der Dienst verwendet den ECDH-based Schlüsselaustausch, um einen ECDH-wrapped TR-31 Schlüsselblock bereitzustellen, den Sie direkt in Ihr Payment Cryptography-Konto importieren. AWS

> **Note**
>
> Wir empfehlen, wann immer möglich auf Standards basierende Standards Schlüssel importieren und exportieren zu verwenden. Verwenden Sie Physical Key Exchange nur, wenn Ihre Partner oder Anbieter elektronische Schlüsselaustauschmethoden wie ANSI X9.24 TR-34, RSA wrap/unwrap oder ECDH nicht unterstützen. ???

So funktioniert der physische Schlüsselaustausch

Um den Austausch von Schlüsseln in Papierform zu initiieren, führt Sie eine CloudFormation Vorlage durch die Einrichtung der Voraussetzungen, einschließlich der Erstellung eines ECC-Schlüsselpaars und eines S3-Buckets in Ihrem Konto. Sie oder Ihr Partner senden dann wichtige Komponenten in Papierform an die AWS sichere Einrichtung, wo geschulte AWS Schlüsselverwalter die Schlüsselzeremonie mithilfe eines Offline-HSM durchführen. Die Ausgabe ist ein ECDH-wrapped TR-31 Schlüsselblock, der in Ihren S3-Bucket hochgeladen wurde und den Sie mithilfe der Methode in Ihr Konto importieren. Schlüssel mithilfe asymmetrischer Techniken (ECDH) importieren Physical Key Exchange unterstützt den Import von KEK- (Key Usage K1) oder BDK (Key Usage B0) Schlüsseln sowohl in TDES- als auch in AES-Schlüsselalgorithmen.

Das folgende Diagramm zeigt den vollständigen Prozess des physischen Schlüsselaustauschs.



1. Initiierung — Sie reichen ein Support-Ticket ein oder arbeiten mit Ihrem Kundenbetreuer zusammen, um eine Anfrage einzureichen.
2. Kundeneinrichtung — AWS Payment Cryptography stellt Ihnen eine CloudFormation Vorlage zur Verfügung, mit der Sie die folgenden erforderlichen Schritte ausführen können:
   - Erstellen Sie ein ECC P521-Schlüsselpaar in Ihrem AWS Payment Cryptography-Konto und rufen Sie das Public-Key-Zertifikat ab.

- Erstellen Sie einen Amazon S3-Bucket mit einer Richtlinie, die dem AWS Payment Cryptography Service Principal Zugriff gewährt. read/write
- Speichern Sie das öffentliche ECC-Zertifikat und die signierende Root-CA im Amazon S3-Bucket.
- Geben Sie die wichtigsten Attribute an: Schlüsselverwendung, wichtige Verwendungsarten und Anzahl der zu sendenden Papierschlüsselkomponenten.

3. S3-Bucket-Namen teilen — Der Kunde teilt den vom CloudFormation Stack erstellten S3-Bucket-Namen, in dem das öffentliche Schlüsselzertifikat, die Zertifikatskette und die Schlüsselattribute gespeichert werden, damit AWS Payment Cryptography den Schlüsselaustausch initiieren kann.
4. Versandkoordination — AWS Payment Cryptography stellt die Versanddetails für die US-based sichere Einrichtung bereit. Sie oder Ihr Partner liefern wichtige Komponenten in Papierform an die AWS wichtigsten Verwahrer.
5. Erhalt der Komponenten — Die AWS Schlüsselverwalter erhalten jedes Papierbauteil und senden für jedes Bauteil eine separate Bestätigung.
6. Schlüsselzeremonie — Die AWS Schlüsselverwalter führen die Schlüsselzeremonie mithilfe eines Offline-HSM durch. Der resultierende TR-31 Schlüsselblock, der mit einem ECDH-derived AES-256 Schlüssel, dem öffentlichen ECC-Zertifikat aus dem Offline-HSM und dem zugehörigen Signaturzertifikat umschlossen wird, werden in Ihren Amazon S3-Bucket hochgeladen.
7. Abschluss — AWS Payment Cryptography sendet eine Bestätigung, dass die Schlüsselzeremonie abgeschlossen ist. Anschließend können Sie den mit ECDH verpackten TR-31 Schlüsselblock mithilfe dieser Methode in Ihr AWS Payment Cryptography-Konto importieren. Schlüssel mithilfe asymmetrischer Techniken (ECDH) importieren
8. Abrechnung — Nach erfolgreichem Abschluss der Schlüsselzeremonie wird Ihnen der ausgetauschte Schlüssel in Rechnung gestellt.

Sicherheit und Compliance

Physical Key Exchange wird in AWS sicheren Einrichtungen betrieben, die so konzipiert sind, dass sie die physischen und logischen Sicherheitsanforderungen von PCI PIN und PCI P2PE erfüllen. Die folgenden Kontrollen sind vorhanden:

Doppelte Kontrolle und Aufgabentrennung

AWS Die wichtigsten Verwalter kommen aus verschiedenen Teams mit getrennten Berichtsstrukturen. Es gibt Verfahren, um sicherzustellen, dass die wichtigsten Schritte der Zeremonien unter doppelter Kontrolle durchgeführt werden.

Offline-HSM

Die wichtigsten Zeremonien werden mit zertifizierten HSM-listed PCI-PTS-Hardwaresicherheitsmodulen durchgeführt, die offline und ohne Netzwerkkonnektivität funktionieren. Ihr Schlüssel existiert niemals im Klartext außerhalb der HSM-Grenze.

Lieferung kryptografischer Schlüssel

Schlüsselmaterial wird mithilfe des ECDH-based Schlüsselaustauschs vom Offline-HSM auf Ihr AWS Payment Cryptography-Konto übertragen, wodurch ein umfassender kryptografischer Schutz gewährleistet ist.

Prüfung und Einhaltung von Vorschriften

AWS verfügt über Verfahren zur Erfüllung der geltenden Compliance-Anforderungen, die in regelmäßigen Abständen für PCI-PIN- und P2PE-Bescheinigungen überprüft werden. Suchen Sie im Compliance-Paket in AWS Artifact nach Berichten, auf die Sie in Ihren eigenen PCI-Assessments Bezug genommen haben.

Verwenden von Aliassen

Ein Alias ist ein benutzerfreundlicher Name für einen AWS Payment Cryptography Key. Mit einem Alias können Sie beispielsweise einen Schlüssel als `alias/test-key` statt als bezeichnen. `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`

Sie können einen Alias verwenden, um einen Schlüssel bei den meisten Vorgängen der Schlüsselverwaltung (Steuerungsebene) und bei kryptografischen Vorgängen (Datenebene) zu identifizieren.

Sie können den Zugriff auf AWS Payment Cryptography-Schlüssel auch anhand ihrer Aliase zulassen und verweigern, ohne die Richtlinien zu bearbeiten oder Zuschüsse zu verwalten. Diese Funktion ist Teil der Serviceunterstützung für attributebasierte Zugriffskontrolle (ABAC).

Darüber hinaus erleichtern Aliasse die Wiederverwendung des gleichen Codes in verschiedenen AWS-Regionen. Erstellen Sie Aliase mit demselben Namen in mehreren Regionen und verknüpfen Sie jeden Alias mit einem AWS Payment Cryptography-Schlüssel in seiner Region. Wenn der Code in jeder Region ausgeführt wird, bezieht sich der Alias auf den zugehörigen AWS Payment Cryptography-Schlüssel in dieser Region.

Mithilfe der API können Sie einen Alias für einen AWS Payment Cryptography-Schlüssel erstellen. `CreateAlias`

Die AWS Payment Cryptography API bietet die vollständige Kontrolle über Aliase in jedem Konto und jeder Region. Die API umfasst Operationen zum Erstellen eines Alias (CreateAlias), zum Anzeigen von Aliasnamen und des verknüpften KeyArn (Listen-Aliases), zum Ändern des mit einem Alias verknüpften AWS Payment Cryptography-Schlüssels (update-alias) und zum Löschen eines Alias (delete-alias).

Themen

- Über Aliasse
- Verwenden von Aliassen in Ihren Anwendungen
- Verbundene APIs

Über Aliasse

Erfahre, wie Aliase in Payment Cryptography funktionieren. AWS

Ein Alias ist eine unabhängige Ressource AWS

Ein Alias ist kein Eigentum eines AWS Payment Cryptography-Schlüssels. Die Aktionen, die Sie für den Alias ausführen, wirken sich nicht auf den zugehörigen Schlüssel aus. Sie können einen Alias für einen AWS Payment Cryptography-Schlüssel erstellen und den Alias dann aktualisieren, sodass er einem anderen AWS Payment Cryptography-Schlüssel zugeordnet ist. Sie können den Alias sogar löschen, ohne dass dies Auswirkungen auf den zugehörigen AWS Payment Cryptography-Schlüssel hat. Wenn Sie einen AWS Payment Cryptography-Schlüssel löschen, wird die Zuweisung aller mit diesem Schlüssel verknüpften Aliase aufgehoben.

Wenn Sie in einer IAM-Richtlinie einen Alias als Ressource angeben, bezieht sich die Richtlinie auf den Alias, nicht auf den zugehörigen AWS Payment Cryptography-Schlüssel.

Jeder Alias hat einen benutzerfreundlichen Namen

Wenn Sie einen Alias erstellen, geben Sie den Aliasnamen mit dem Präfix an`alias/`. Zum Beispiel `alias/test_1234`

Jeder Alias ist jeweils einem AWS Payment Cryptography-Schlüssel zugeordnet

Der Alias und sein AWS Payment Cryptography-Schlüssel müssen sich im selben Konto und in derselben Region befinden.

Ein AWS Payment Cryptography-Schlüssel kann mehreren Alias gleichzeitig zugeordnet werden, aber jeder Alias kann nur einem einzigen Schlüssel zugeordnet werden

Diese `list-aliases` Ausgabe zeigt beispielsweise, dass der `alias/sampleAlias1` Alias genau einem AWS Payment Cryptography-Zielschlüssel zugeordnet ist, der durch die Eigenschaft repräsentiert wird. `KeyArn`

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

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

Diesem AWS Payment Cryptography-Schlüssel können mehrere Aliase zugeordnet werden

Beispielsweise können Sie die `alias/sampleAlias2` Aliase `alias/sampleAlias1;` und demselben Schlüssel zuordnen.

```
$ 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"
            }
        ]
```

```
}
```

Ein Alias muss für ein bestimmtes Konto und eine bestimmte Region eindeutig sein

Zum Beispiel können Sie nur einen `alias/sampleAlias1`-Alias in jedem Konto und in jeder Region haben. Aliase unterscheiden zwischen Groß- und Kleinschreibung, wir raten jedoch davon ab, Aliase zu verwenden, die sich nur in der Groß- und Kleinschreibung unterscheiden, da sie fehleranfällig sein können. Sie können keinen Aliasnamen ändern. Sie können den Alias jedoch löschen und einen neuen Alias mit dem gewünschten Namen erstellen.

Sie können jedoch einen Alias mit demselben Namen in verschiedenen Regionen erstellen.

Sie können beispielsweise einen Alias `alias/sampleAlias2` in den USA Ost (Nord-Virginia) und einen Alias `alias/sampleAlias2` in den USA West (Oregon) verwenden. Jeder Alias wäre mit einem AWS Payment Cryptography-Schlüssel in seiner Region verknüpft. Wenn Ihr Code auf einen Aliasnamen wie `alias/finance-key` verweist, können Sie ihn in mehreren Regionen ausführen. In jeder Region wird ein anderes alias/sampleAlias2 verwendet. Details hierzu finden Sie unter Verwenden von Aliassen in Ihren Anwendungen.

Sie können den AWS Payment Cryptography-Schlüssel ändern, der einem Alias zugeordnet ist

Sie können den `UpdateAlias` Vorgang verwenden, um einen Alias einem anderen AWS Payment Cryptography-Schlüssel zuzuordnen. Wenn der `alias/sampleAlias2` Alias beispielsweise mit dem `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h` AWS Payment Cryptography-Schlüssel verknüpft ist, können Sie ihn aktualisieren, sodass er dem `arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi` Schlüssel zugeordnet wird.

> **Warning**
>
> AWS Payment Cryptography überprüft nicht, ob der alte und der neue Schlüssel dieselben Attribute wie die Schlüsselverwendung aufweisen. Das Aktualisieren mit einem anderen Schlüsseltyp kann zu Problemen in Ihrer Anwendung führen.

> **Warning**
>
> Aktualisierungen von Aliasnamen unterliegen einer etwaigen Konsistenz. Nachdem Sie einen Alias aktualisiert haben, gibt es einen kurzen Zeitraum, in dem der Alias

möglicherweise in den alten oder den neuen Schlüssel aufgelöst wird. Aktualisieren Sie keine Aliase, wenn dieses Verhalten Ihre Prozesse unterbrechen würde.

Einige Schlüssel haben keine Aliase

Ein Alias ist eine optionale Funktion, und nicht alle Schlüssel haben Aliase, es sei denn, Sie entscheiden sich dafür, Ihre Umgebung auf diese Weise zu betreiben. Schlüssel können mithilfe des Befehls mit Aliasen verknüpft werden. `create-alias` Sie können auch den Vorgang update-alias verwenden, um den AWS Payment Cryptography-Schlüssel zu ändern, der einem Alias zugeordnet ist, und den Vorgang delete-alias, um einen Alias zu löschen. Daher haben einige AWS Payment Cryptography-Schlüssel möglicherweise mehrere Aliase, andere wiederum keine.

Einen Schlüssel einem Alias zuordnen

Mit dem `create-alias` Befehl können Sie einen Schlüssel (dargestellt durch einen ARN) einem oder mehreren Aliasen zuordnen. Dieser Befehl ist nicht idempotent. Verwenden Sie den Befehl update-alias, um einen Alias zu aktualisieren.

```
$ 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"
    }
}
```

Verwenden von Aliassen in Ihren Anwendungen

Sie können einen Alias verwenden, um einen AWS Payment Cryptography-Schlüssel in Ihrem Anwendungscode darzustellen. Der `key-identifier` Parameter in AWS Payment Cryptography-Datenoperationen sowie in anderen Operationen wie List Keys akzeptiert einen Aliasnamen oder einen Alias-ARN.

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

Denken Sie bei der Verwendung eines Alias-ARN daran, dass die Aliaszuordnung zu einem AWS Payment Cryptography-Schlüssel in dem Konto definiert ist, dem der AWS Payment Cryptography-Schlüssel gehört, und dass sie in jeder Region unterschiedlich sein kann.

Eine der mächtigsten Verwendungen von Aliasen ist in Anwendungen, die in mehreren AWS-Regionen ausgeführt werden.

Sie könnten in jeder Region eine andere Version Ihrer Anwendung erstellen oder ein Wörterbuch, eine Konfiguration oder eine Switch-Anweisung verwenden, um den richtigen AWS Payment Cryptography-Schlüssel für jede Region auszuwählen. Es könnte jedoch einfacher sein, in jeder Region einen Alias mit demselben Aliasnamen zu erstellen. Bei dem Aliasnamen wird zwischen Groß-/Kleinschreibung unterschieden.

Verbundene APIs

[Tags]

> Tags sind Schlüssel- und Wertepaare, die als Metadaten für die Organisation Ihrer AWS Payment Cryptography-Schlüssel dienen. Sie können verwendet werden, um Schlüssel flexibel zu identifizieren oder einen oder mehrere Schlüssel zu gruppieren.

Holen Sie sich Schlüssel

Ein AWS Payment Cryptography Key stellt eine einzelne Einheit kryptografischen Materials dar und kann nur für kryptografische Operationen für diesen Dienst verwendet werden. Die GetKeys API verwendet einen KeyIdentifier als Eingabe und gibt wichtige Metadaten wie Attribute, Status und Zeitstempel zurück, gibt jedoch kein echtes kryptografisches Schlüsselmaterial zurück.

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"
  }
}
```

Ordnet die Öffentlichkeit key/certificate einem Schlüsselpaar zu

Get Public Key/Certificate gibt den durch den angegebenen öffentlichen Schlüssel zurück`KeyArn`. Dies kann der öffentliche Schlüsselteil eines Schlüsselpaars sein, das mit AWS Payment Cryptography generiert wurde, oder ein zuvor importierter öffentlicher Schlüssel. Der häufigste Anwendungsfall ist die Bereitstellung des öffentlichen Schlüssels für einen externen Dienst, der Daten verschlüsselt. Diese Daten können dann an eine Anwendung weitergegeben werden, die AWS Payment Cryptography nutzt, und die Daten können mithilfe des privaten Schlüssels, der in Payment Cryptography gesichert ist, entschlüsselt werden. AWS

Der Dienst gibt öffentliche Schlüssel als öffentliches Zertifikat zurück. Das API-Ergebnis enthält die CA und das Public-Key-Zertifikat. Beide Datenelemente sind Base64-kodiert.

> **Note**
>
> Das zurückgegebene öffentliche Zertifikat soll kurzlebig sein und nicht idempotent sein. Möglicherweise erhalten Sie bei jedem API-Aufruf ein anderes Zertifikat, auch wenn der öffentliche Schlüssel selbst unverändert ist.

Example

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


```
{
  "KeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2VENDQXFXZ0F3SUJBZ0lSQUo1OWd2VkpDd3dlYldMNldYZEpYY
  "KeyCertificateChain":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThtZ0F3SUJBZ0lSQUtlN2piaHFKZjJPd3FGUWI5c3VuC
}
```

Tagging von Schlüsseln

In der AWS Zahlungskryptografie können Sie einem Schlüssel für AWS Zahlungskryptografie Tags hinzufügen, wenn Sie einen Schlüssel erstellen, und vorhandene Schlüssel kennzeichnen oder deren Markierung aufheben, sofern sie nicht gelöscht werden müssen. Tags sind optional, aber sie können sehr nützlich sein.

Allgemeine Informationen zu Tags, einschließlich bewährter Methoden, Tagging-Strategien sowie Format und Syntax von Tags, finden Sie unter Ressourcen zum AWS Taggen in der. Allgemeine Amazon Web Services-Referenz

Themen

- Informationen zu Stichwörtern in AWS Kryptografie im Zahlungsverkehr
- Wichtige Tags in der Konsole anzeigen
- Verwaltung von Schlüsseltags mit API-Vorgängen
- Zugriffssteuerung mit Tags
- Verwendung von Tags zur Steuerung des Zugriffs auf Schlüssel

Informationen zu Stichwörtern in AWS Kryptografie im Zahlungsverkehr

Ein Tag ist ein optionales Metadatenlabel, das Sie einer AWS Ressource zuweisen (oder zuweisen AWS können). Jedes Tag besteht aus einem Tag-Schlüssel und einem Tag-Wert, wobei beide Zeichenfolgen zwischen Groß-/Kleinschreibung unterscheiden. Der Wert kann auch eine leere (Null) Zeichenfolge sein. Jedes Tag auf einer Ressource muss einen anderen Tag-Schlüssel haben, aber Sie können dasselbe Tag mehreren AWS Ressourcen hinzufügen. Eine Ressource kann bis zu 50 Tags besitzen, die von Benutzern erstellt wurden.

Geben Sie keine vertraulichen oder sensiblen Informationen in den Tag-Schlüssel oder Tag-Wert ein. Stichwörter sind für viele zugänglich AWS-Services, auch für die Abrechnung.

In AWS Payment Cryptography können Sie einem Schlüssel bei der Erstellung des Schlüssels Tags hinzufügen und vorhandene Schlüssel kennzeichnen oder deren Markierung aufheben, sofern sie nicht gelöscht werden müssen. Sie können Aliase nicht taggen. Tags sind optional, aber sie können sehr nützlich sein.

Sie können beispielsweise allen AWS Payment Cryptography-Schlüsseln und Amazon S3-Buckets, die Sie für das Alpha-Projekt verwenden, ein `"Project"="Alpha"` Tag hinzufügen. Ein anderes

Beispiel ist das Hinzufügen eines `"BIN"="20130622"` Tags zu allen Schlüsseln, die einer bestimmten Bankidentifikationsnummer (BIN) zugeordnet sind.

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

Allgemeine Informationen zu Tags, einschließlich des Formats und der Syntax, finden Sie unter AWS Tagging-Ressourcen im Allgemeine Amazon Web Services-Referenz.

Tags sind für folgende Aktivitäten nützlich:

- Identifizieren und organisieren Sie Ihre AWS Ressourcen. Viele AWS Dienste unterstützen Tagging, sodass Sie Ressourcen aus verschiedenen Diensten dasselbe Tag zuweisen können, um anzuzeigen, dass die Ressourcen miteinander verwandt sind. Sie können beispielsweise einem AWS Payment Cryptography Keys und einem Amazon Elastic Block Store (Amazon EBS) -Volume oder Secret dasselbe Tag zuweisen. AWS Secrets Manager Sie können Tags auch verwenden, um Schlüssel für die Automatisierung zu identifizieren.
- Verfolgen Sie Ihre AWS Kosten. Wenn Sie Ihren AWS Ressourcen Stichwörter hinzufügen, AWS wird ein Kostenzuordnungsbericht generiert, in dem Nutzung und Kosten nach Stichwörtern zusammengefasst werden. Mit dieser Funktion können Sie die Kosten für AWS Payment Cryptography für ein Projekt, eine Anwendung oder eine Kostenstelle verfolgen.

  Weitere Informationen zur Verwendung von Tags für die Kostenzuordnung finden Sie unter Verwenden von Kostenzuordnungs-Tags im AWS Billing -Benutzerhandbuch. Informationen zu den Regeln für Tag-Schlüssel und Tag-Werte finden Sie unter User-Defined Tag-Einschränkungen im AWS Billing Benutzerhandbuch.
- Kontrollieren Sie den Zugriff auf Ihre AWS Ressourcen. Das Zulassen und Verweigern des Zugriffs auf Schlüssel anhand ihrer Tags ist Teil der AWS Payment Cryptography-Unterstützung für attributebasierte Zugriffskontrolle (ABAC). Informationen zur Steuerung des Zugriffs auf

AWS Payment Cryptography anhand ihrer Tags finden Sie unter. Autorisierung basiert auf AWS Kryptografie-Tags für Zahlungen Weitere allgemeine Informationen zur Verwendung von Tags zur Steuerung des Zugriffs auf AWS Ressourcen finden Sie unter Steuern des Zugriffs auf AWS Ressourcen mithilfe von Ressourcen-Tags im IAM-Benutzerhandbuch.

AWS Payment Cryptography schreibt einen Eintrag in Ihr AWS CloudTrail Protokoll, wenn Sie die Operationen TagResource, UntagResource, oder ListTagsForResource verwenden.

Wichtige Tags in der Konsole anzeigen

Um Tags in der Konsole anzuzeigen, benötigen Sie die Tagging-Berechtigung für den Schlüssel aus einer IAM-Richtlinie, die den Schlüssel enthält. Sie benötigen diese Berechtigungen zusätzlich zu den Berechtigungen zum Anzeigen von Schlüsseln in der Konsole.

Verwaltung von Schlüsseltags mit API-Vorgängen

Sie können die AWS Payment Cryptography API verwenden, um Tags für die von Ihnen verwalteten Schlüssel hinzuzufügen, zu löschen und aufzulisten. Für diese Beispiele wird die AWS Command Line Interface (AWS CLI) verwendet. Sie können aber jede unterstützte Programmiersprache nutzen. Sie können nicht taggen Von AWS verwaltete Schlüssel.

Um Tags für einen Schlüssel hinzuzufügen, zu bearbeiten, anzuzeigen und zu löschen, benötigen Sie die erforderlichen Berechtigungen. Details hierzu finden Sie unter Zugriffssteuerung mit Tags.

Themen

- CreateKey: Fügen Sie Tags zu einem neuen Schlüssel hinzu
- TagResource: Hinzufügen oder Ändern von Stichwörtern für einen Schlüssel
- ListResourceTags: Ruft die Schlagworte für einen Schlüssel ab
- UntagResource: Löscht Tags aus einem Schlüssel

CreateKey: Fügen Sie Tags zu einem neuen Schlüssel hinzu

Sie können Tags hinzufügen, wenn Sie einen Schlüssel erstellen. Verwenden Sie den `Tags` Parameter der CreateKey Operation, um die Tags anzugeben.

Um beim Erstellen eines Schlüssels Tags hinzuzufügen, muss der Aufrufer über die `payment-cryptography:TagResource` entsprechende Berechtigung in einer IAM-Richtlinie verfügen. Die

Berechtigung muss mindestens alle Schlüssel im Konto und in der Region abdecken. Details hierzu finden Sie unter Zugriffssteuerung mit Tags.

Der Wert des `Tags`-Parameters von `CreateKey` ist eine Sammlung von Tag-Schlüssel- und Tag-Wert-Paaren, unter Beachtung der Groß-/Kleinschreibung. Jedes Tag auf einem Schlüssel muss einen anderen Tag-Namen haben. Der Tag-Wert kann auch eine leere (Null) Zeichenfolge sein.

Mit dem folgenden AWS CLI Befehl wird beispielsweise ein symmetrischer Verschlüsselungsschlüssel mit einem `Project:Alpha` Tag erstellt. Wenn Sie mehr als ein Schlüssel-Wert-Paar angeben, verwenden Sie ein Leerzeichen, um die einzelnen Paare zu trennen.

```
$ aws 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"}]'
```

Wenn dieser Befehl erfolgreich ist, gibt er ein `Key` Objekt mit Informationen über den neuen Schlüssel zurück. Die `Key` enthalten jedoch keine Tags. Verwenden Sie die ListResourceTags Operation, um die Tags abzurufen.

TagResource: Hinzufügen oder Ändern von Stichwörtern für einen Schlüssel

Die TagResource Operation fügt einem Schlüssel ein oder mehrere Tags hinzu. Sie können diese Operation nicht verwenden, um Tags in einem anderen AWS-Konto hinzuzufügen oder zu bearbeiten.

Geben Sie zum Hinzufügen eines Tags einen neuen Tag-Schlüssel und einen Tag-Wert an. Um ein Tag zu bearbeiten, geben Sie einen vorhandenen Tag-Schlüssel und einen neuen Tag-Wert an. Jedes Tag auf einem Schlüssel muss einen anderen Tag-Schlüssel haben. Der Tag-Wert kann auch eine leere (Null) Zeichenfolge sein.

Beispielsweise fügt **`UseCase`** der folgende Befehl einem Beispielschlüssel **`BIN`** Tags hinzu.

```
$ 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"}]'
```

Wenn der Befehl erfolgreich war, erfolgt keine Ausgabe. Verwenden Sie die ListResourceTags Operation, um die Tags auf einer Taste anzuzeigen.

Sie können auch TagResource verwenden, um den Tag-Wert für ein vorhandenes Tag zu ändern. Um einen Tag-Wert zu ersetzen, geben Sie den gleichen Tag-Schlüssel mit einem unterschiedlichen Wert an. Tags, die nicht in einem Änderungsbefehl aufgeführt sind, werden nicht geändert oder entfernt.

Beispielsweise ändert dieser Befehl den Wert des `Project`-Tag von `Alpha` auf `Noe`.

Der Befehl wird http/200 ohne Inhalt zurückgegeben. Um Ihre Änderungen zu sehen, verwenden Sie `ListTagsForResource`

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

ListResourceTags: Ruft die Schlagworte für einen Schlüssel ab

Die ListResourceTags Operation ruft die Tags für einen Schlüssel ab. Der Parameter `ResourceArn` (KeyArn oder KeyAlias) ist erforderlich. Sie können diesen Vorgang nicht verwenden, um die Tags auf Schlüsseln in einer anderen Datei anzuzeigen. AWS-Konto

Mit dem folgenden Befehl werden beispielsweise die Tags für einen Beispielschlüssel abgerufen.

```
$ 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: Löscht Tags aus einem Schlüssel

Die UntagResource Operation löscht Tags aus einem Schlüssel. Um die zu löschenden Tags zu identifizieren, geben Sie die Tag-Schlüssel an. Sie können diesen Vorgang nicht verwenden, um Tags aus Schlüsseln anderer AWS-Konto Art zu löschen.

Wenn sie erfolgreich ist, gibt die `UntagResource`-Operation keine Ausgabe zurück. Wenn der angegebene Tag-Schlüssel auf dem Schlüssel nicht gefunden wird, wird außerdem keine Ausnahme ausgelöst und keine Antwort zurückgegeben. Verwenden Sie den Vorgang, um zu bestätigen, dass der ListResourceTags Vorgang funktioniert hat.

Beispielsweise löscht dieser Befehl das **`Purpose`** Tag und seinen Wert aus dem angegebenen Schlüssel.

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

Zugriffssteuerung mit Tags

Um mithilfe der API Tags hinzuzufügen, anzuzeigen und zu löschen, benötigen Principals Tagging-Berechtigungen in den IAM-Richtlinien.

Sie können diese Berechtigungen auch einschränken, indem Sie AWS globale Bedingungsschlüssel für Tags verwenden. In der AWS Zahlungskryptografie können diese Bedingungen den Zugriff auf Tagging-Operationen wie TagResource und steuern. UntagResource

Beispielrichtlinien und weitere Informationen finden Sie unter Zugriffssteuerung anhand von Tag-Schlüsseln im IAM-Benutzerhandbuch.

Berechtigungen zum Erstellen und Verwalten von Tags funktionieren wie folgt.

Zahlungskryptografie: TagResource

Erlaubt es Prinzipalen, Tags hinzuzufügen oder zu bearbeiten. Um beim Erstellen eines Schlüssels Tags hinzuzufügen, muss der Prinzipal über die entsprechenden Berechtigungen in einer IAM-Richtlinie verfügen, die nicht auf bestimmte Schlüssel beschränkt ist.

Zahlungs-Kryptografie: ListTagsForResource

Ermöglicht es Principals, Stichwörter auf Schlüsseln einzusehen.

Zahlungs-Kryptografie: UntagResource

Ermöglicht es Prinzipalen, Stichwörter aus Schlüsseln zu löschen.

Tag-Berechtigungen in Richtlinien

Sie können Markierungs-Berechtigungen in einer Schlüsselrichtlinie oder einer IAM-Richtlinie bereitstellen. Beispielsweise erteilt das folgende Beispiel einer Schlüsselrichtlinie ausgewählten Benutzern die Berechtigung, den Schlüssel zu taggen. Sie erteilt allen Benutzern, die die Beispiel-Administrator- oder Entwicklerrollen übernehmen können, die Berechtigung zum Anzeigen von Tags.

JSON

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

Um Hauptbenutzern die Berechtigung zum Taggen mehrerer Schlüssel zu erteilen, können Sie eine IAM-Richtlinie verwenden. Damit diese Richtlinie wirksam ist, muss die Schlüsselrichtlinie für jeden Schlüssel es dem Konto ermöglichen, IAM-Richtlinien zur Steuerung des Zugriffs auf den Schlüssel zu verwenden.

Die folgende IAM-Richtlinie ermöglicht es den Prinzipalen beispielsweise, Schlüssel zu erstellen. Sie ermöglicht ihnen auch, Tags für alle Schlüssel im angegebenen Konto zu erstellen und zu verwalten. Diese Kombination ermöglicht es den Principals, den Tags-Parameter des CreateKey Vorgangs zu verwenden, um einem Schlüssel Tags hinzuzufügen, während sie ihn erstellen.

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/*"
    }
  ]
}
```

Beschränken von Tag-Berechtigungen

Sie können Markierungs-Berechtigungen einschränken, indem Sie Richtlinienbedingungen verwenden. Die folgenden Richtlinienbedingungen können auf die `payment-cryptography:TagResource`- und `payment-cryptography:UntagResource`-Berechtigungen angewendet werden. Beispielsweise können Sie mit der `aws:RequestTag/`*`tag-key`*-Bedingung einem Prinzipal erlauben, nur bestimmte Tags hinzuzufügen, oder verhindern, dass ein Prinzipal Tags mit bestimmten Tag-Schlüsseln hinzufügen kann.

- war: RequestTag
- aws:ResourceTag/tag-key (nur IAM-Richtlinien)
- aws: TagKeys

Wenn Sie Tags verwenden, um den Zugriff auf Schlüssel zu kontrollieren, empfiehlt es sich, die `aws:TagKeys` Bedingungstaste `aws:RequestTag/`*`tag-key`* oder zu verwenden, um zu ermitteln, welche Tags (oder Tag-Schlüssel) zulässig sind.

Die folgende IAM-Richtlinie ähnelt beispielsweise der vorherigen. Diese Richtlinie erlaubt den Prinzipalen jedoch das Erstellen von Tags (`TagResource`) und das Löschen von Tags (`UntagResource`) nur für Tags mit einem `Project`-Tag-Schlüssel.

Da `TagResource` `UntagResource` N-Anfragen mehrere Tags enthalten können, müssen Sie einen Operator `ForAllValues` or `ForAnyValue` set mit der TagKeys Bedingung aws: angeben. Der `ForAnyValue`-Operator erfordert, dass mindestens einer der Tag-Schlüssel in der Anforderung mit einem der Tag-Schlüssel in der Richtlinie übereinstimmen muss. Der `ForAllValues`-Operator erfordert, dass alle der Tag-Schlüssel in der Anforderung mit einem der Tag-Schlüssel in der Richtlinie übereinstimmen müssen. Der `ForAllValues` Operator kehrt auch zurück, `true` wenn die Anfrage keine Tags enthält. Er schlägt jedoch TagResource UntagResource fehl, wenn keine Tags angegeben sind. Ausführliche Informationen zu den Satz-Operatoren finden Sie unter Verwenden mehrerer Schlüssel und Werte im IAM-Benutzerhandbuch.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKey",
      "Effect": "Allow",
      "Action": "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"}
      }
    }
  ]
}
```

Verwendung von Tags zur Steuerung des Zugriffs auf Schlüssel

Sie können den Zugriff auf AWS Payment Cryptography anhand der Tags auf dem Schlüssel steuern. Sie können beispielsweise eine IAM-Richtlinie schreiben, die es Principals ermöglicht, nur die Schlüssel zu aktivieren und zu deaktivieren, die über ein bestimmtes Tag verfügen. Oder Sie können eine IAM-Richtlinie verwenden, um zu verhindern, dass Principals Schlüssel bei kryptografischen Vorgängen verwenden, es sei denn, der Schlüssel hat ein bestimmtes Tag.

Diese Funktion ist Teil der AWS Payment Cryptography-Unterstützung für attributbasierte Zugriffskontrolle (ABAC). Informationen zur Verwendung von Tags zur Steuerung des Zugriffs auf AWS Ressourcen finden Sie unter Wofür ist ABAC gedacht? AWSund Steuern des Zugriffs auf AWS Ressourcen mithilfe von Ressourcen-Tags im IAM-Benutzerhandbuch.

AWS Payment Cryptography unterstützt den globalen Bedingungskontextschlüssel aws:ResourceTag/tag-key, mit dem Sie den Zugriff auf Schlüssel anhand der Tags auf dem Schlüssel steuern können. Da mehrere Schlüssel dasselbe Tag haben können, können Sie mit dieser Funktion die Berechtigung auf einen ausgewählten Satz von Schlüsseln anwenden. Sie können die Schlüssel im Satz auch einfach ändern, indem Sie ihre Tags ändern.

In AWS Payment Cryptography wird der `aws:ResourceTag/`*`tag-key`* Bedingungsschlüssel nur in IAM-Richtlinien unterstützt. Er wird nicht in wichtigen Richtlinien unterstützt, die nur für einen Schlüssel gelten, oder für Vorgänge, die keinen bestimmten Schlüssel verwenden, wie z. B. die ListKeys ListAliases OR-Operationen.

Die Steuerung des Zugriffs mit Tags bietet eine einfache, skalierbare und flexible Möglichkeit, Berechtigungen zu verwalten. Wenn es jedoch nicht ordnungsgemäß konzipiert und verwaltet wird, kann es den unbeabsichtigten Zugriff auf Ihre Schlüssel zulassen oder verweigern. Wenn Sie Tags verwenden, um den Zugriff zu steuern, sollten Sie die folgenden Methoden berücksichtigen.

- Verwenden Sie Tags, um beim Zugriff die bewährte Methode der geringsten Berechtigung zu befolgen. Erteilen Sie IAM-Prinzipalen nur die Berechtigungen, die sie benötigen, und zwar nur für die Schlüssel, die sie verwenden oder verwalten müssen. Verwenden Sie beispielsweise Tags, um die für ein Projekt verwendeten Schlüssel zu kennzeichnen. Erteilen Sie dem Projektteam dann die Erlaubnis, nur Schlüssel mit dem Projekt-Tag zu verwenden.
- Seien Sie vorsichtig, wenn Sie Prinzipalen die `payment-cryptography:TagResource`- und `payment-cryptography:UntagResource`-Berechtigung erteilen, mit denen sie Tags hinzufügen, bearbeiten und löschen können. Wenn Sie Stichwörter verwenden, um den Zugriff auf Schlüssel zu kontrollieren, kann das Ändern eines Stichworts dazu führen, dass Schulleiter Schlüssel verwenden dürfen, zu deren Verwendung sie sonst nicht berechtigt wären. Es kann auch den Zugriff auf Schlüssel verweigern, die andere Schulleiter für ihre Arbeit benötigen. Schlüsseladministratoren, die nicht berechtigt sind, wichtige Richtlinien zu ändern oder Zuschüsse zu gewähren, können den Zugriff auf Schlüssel kontrollieren, sofern sie über die Berechtigung zur Verwaltung von Stichwörtern verfügen.

Verwenden Sie nach Möglichkeit eine Richtlinienbedingung, z. B. `aws:RequestTag/`*`tag-key`* oder, `aws:TagKeys` um die Tagging-Berechtigungen eines Principals auf bestimmte Stichwörter oder Tag-Muster für bestimmte Schlüssel zu beschränken.

- Prüfen Sie die Principals in Ihrem AWS-Konto , die derzeit über die Berechtigungen zum Markieren und Aufheben der Markierung verfügen, und passen Sie sie gegebenenfalls an. IAM-Richtlinien ermöglichen möglicherweise Berechtigungen zum Markieren und Aufheben von Tags für alle Schlüssel. Die vom Administrator verwaltete Richtlinie ermöglicht es beispielsweise Hauptbenutzern, Tags für alle Schlüssel zu kennzeichnen, zu entfernen und sie aufzulisten.
- Bevor Sie eine Richtlinie festlegen, die von einem Schlagwort abhängt, überprüfen Sie die Stichwörter auf den Schlüsseln in Ihrem. AWS-Konto Stellen Sie sicher, dass Ihre Richtlinie nur für die Tags gilt, die Sie einschließen möchten. Verwenden Sie CloudTrail Protokolle und CloudWatch Alarme, um Sie vor Tag-Änderungen zu warnen, die den Zugriff auf Ihre Schlüssel beeinträchtigen könnten.
- Die tag-basierten Richtlinienbedingungen verwenden Musterabgleich; sie sind nicht an eine bestimmte Instance eines Tags gebunden. Eine Richtlinie, die Tag-basierte Bedingungsschlüssel verwendet, wirkt sich auf alle neuen und vorhandenen Tags aus, die dem Muster entsprechen. Wenn Sie ein Tag löschen und neu erstellen, das einer Richtlinienbedingung entspricht, gilt die Bedingung für das neue Tag, genau wie für das alte Tag.

Betrachten Sie beispielsweise die folgende IAM-Richtlinie: Dadurch können die Hauptbenutzer die https://docs.aws.amazon.com/payment-cryptography/latest/APIReference/API_Decrypt.html Entschlüsselungsvorgänge nur für Schlüssel in Ihrem Konto aufrufen, die der Region USA Ost (Nord-Virginia) entsprechen und mit einem `"Project"="Alpha"` Tag versehen sind. Sie können diese Richtlinie an Rollen im Beispiel Alpha-Projekt anfügen.

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"
        }
      }
    }
  ]
}
```

Das folgende Beispiel für eine IAM-Richtlinie ermöglicht es den Prinzipalen, jeden Schlüssel im Konto für bestimmte kryptografische Operationen zu verwenden. Sie verbietet den Prinzipalen jedoch, diese kryptografischen Operationen für Schlüssel mit oder ohne Tag zu verwenden. `"Type"="Reserved"` `"Type"`

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"
        }
      }
    }
  ]
}
```

Grundlegendes zu Schlüsselattributen für AWS Kryptografie-Schlüssel für Zahlungen

Ein Grundsatz einer ordnungsgemäßen Schlüsselverwaltung ist, dass Schlüssel einen angemessenen Gültigkeitsbereich haben und nur für zulässige Operationen verwendet werden können. Daher können bestimmte Schlüssel nur mit bestimmten Tastenverwendungsmodi erstellt werden. Wann immer möglich, entspricht dies den verfügbaren Nutzungsmodi, wie sie von TR-31 definiert sind.

Obwohl AWS Payment Cryptography Sie daran hindert, ungültige Schlüssel zu erstellen, finden Sie hier der Einfachheit halber gültige Kombinationen.

Symmetrische Schlüssel

- TR31_B0_BASE_DERIVATION_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256

  - Zulässige Kombination der wichtigsten Nutzungsmodi: {= true}, {= true} DeriveKey NoRestrictions
- TR31_C0_CARD_VERIFICATION_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128*, AES_192*, AES_256*
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Generate = true}, {Verify = true}, {Generate = true, Verify= true}, {= true} NoRestrictions
- TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true}, {Encrypt = true, Wrap = true}, {Decrypt = true, Unwrap = true}, {= true} NoRestrictions
- TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY*, AES_128*, AES_192*, AES_256*
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {= true DeriveKey }, {= true} NoRestrictions
- TR31_E1_EMV_MKEY_CONFIDENTIALITY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128*, AES_192*, AES_256*
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {= DeriveKey true}, {= true} NoRestrictions
- TR31_E2_EMV_MKEY_INTEGRITY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128*, AES_192*, AES_256*
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {= DeriveKey true}, {= true} NoRestrictions
- TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128*, AES_192*, AES_256*
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {= DeriveKey true}, {= true} NoRestrictions
- TR31_E5_EMV_MKEY_CARD_PERSONALIZATION
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128*, AES_192*, AES_256*
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {= DeriveKey true}, {= true} NoRestrictions
- TR31_E6_EMV_MKEY_OTHER
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128*, AES_192*, AES_256*
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {= DeriveKey true}, {= true} NoRestrictions
- TR31_K0_KEY_ENCRYPTION_KEY

  - Es wird empfohlen, TR31_K1_KEY_BLOCK_PROTECTION_KEY zu verwenden. Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true}, {Encrypt = true, Wrap = true}, {Decrypt = true, Unwrap = true}, {= true} NoRestrictions
- TR31_K1_KEY_BLOCK_PROTECTION_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true}, {Encrypt = true, Wrap = true}, {Decrypt = true, Unwrap = true}, {= true} NoRestrictions
- TR31_M1_ISO_9797_1_MAC_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Generate = true}, {Verify = true}, {Generate = true, Verify= true}, {= true} NoRestrictions
- TR31_M3_ISO_9797_3_MAC_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Generate = true}, {Verify = true}, {Generate = true, Verify= true}, {= true} NoRestrictions
- TR31_M6_ISO_9797_5_CMAC_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Generate = true}, {Verify = true}, {Generate = true, Verify= true}, {= true} NoRestrictions
- TR31_M7_HMAC_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Generate = true}, {Verify = true}, {Generate = true, Verify= true}, {= true} NoRestrictions
- TR31_P0_PIN_ENCRYPTION_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true}, {Encrypt = true, Wrap = true}, {Decrypt = true, Unwrap = true}, {= true} NoRestrictions
- TR31_V1_IBM3624_PIN_VERIFICATION_KEY

  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Generate = true}, {Verify = true}, {Generate = true, Verify= true}, {= true} NoRestrictions
- TR31_V2_VISA_PIN_VERIFICATION_KEY
  - Zulässige Schlüsselalgorithmen: TDES_2KEY, TDES_3KEY, AES_128, AES_192, AES_256
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Generate = true}, {Verify = true}, {Generate = true, Verify= true}, {= true} NoRestrictions

Asymmetrische Schlüssel

- TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION
  - Zulässige Schlüsselalgorithmen: RSA_2048, RSA_3072, RSA_4096
  - Zulässige Kombination von Schlüsselverwendungsmodi: {Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true}, {Encrypt = true, Wrap = true}, {Decrypt = true, Unwrap = true}
  - HINWEIS: {Encrypt = true, Wrap = true} ist die einzig gültige Option beim Import eines öffentlichen Schlüssels, der zum Verschlüsseln von Daten oder zum Umschließen eines Schlüssels vorgesehen ist
- TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE
  - Zulässige Schlüsselalgorithmen: RSA_2048, RSA_3072, RSA_4096
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {Sign = true}, {Verify = true}
  - HINWEIS: {Verify = true} ist die einzig gültige Option, wenn ein Schlüssel importiert wird, der zum Signieren bestimmt ist, z. B. ein Stammzertifikat, ein Zwischenzertifikat oder ein Signaturzertifikat für TR-34.
- TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT
  - Wird für wichtige Vereinbarungsalgorithmen wie ECDH verwendet
  - Zulässige Schlüsselalgorithmen: ECC_NIST_P256, ECC_NIST_P384, ECC_NIST_P521
  - Zulässige Kombination der wichtigsten Nutzungsmodi: {= true} DeriveKey .
  - HINWEIS: DeriveKeyUsage wird verwendet, um anzugeben, welche Art von Schlüssel von diesem Basisschlüssel abgeleitet wird. Dies ist bei Schlüssel behoben creation/import.
- TR31_K2_TR34_ASYMMETRIC_KEY
  - Asymmetrischer Schlüssel, der für kompatible Schlüsselaustauschmechanismen verwendet wird, wie X9.24 TR-34

- Zulässige Schlüsselalgorithmen: RSA_2048, RSA_3072, RSA_4096
- Zulässige Kombination der wichtigsten Nutzungsmodi: {= true}. DeriveKey
- Zulässige Kombination der wichtigsten Nutzungsmodi: {Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true}, {Encrypt = true, Wrap = true}, {Decrypt = true, Unwrap = true}
- HINWEIS: {Encrypt = true, Wrap = true} ist die einzig gültige Option beim Import eines öffentlichen Schlüssels, der zum Verschlüsseln von Daten oder zum Umschließen eines Schlüssels vorgesehen ist

* Diese algorithm/key Typkombination wird derzeit von keinen kryptografischen Operationen unterstützt

Datenoperationen

Nachdem Sie einen Schlüssel für die AWS Zahlungskryptografie eingerichtet haben, kann dieser zur Durchführung kryptografischer Operationen verwendet werden. Verschiedene Operationen führen unterschiedliche Arten von Aktivitäten aus, die von Verschlüsselung und Hashing bis hin zu domänenspezifischen Algorithmen wie der CVV2-Generierung reichen.

Verschlüsselte Daten können ohne den entsprechenden Entschlüsselungsschlüssel (je nach Verschlüsselungstyp der symmetrische Schlüssel oder der private Schlüssel) nicht entschlüsselt werden. Hashing- und domänenspezifische Algorithmen können ohne den symmetrischen Schlüssel oder den öffentlichen Schlüssel ebenfalls nicht verifiziert werden.

Informationen zu gültigen Schlüsseltypen für bestimmte Operationen finden Sie unter Gültige Schlüssel für kryptografische Operationen

> **Note**
>
> Wir empfehlen, Testdaten in einer Umgebung außerhalb der Produktionsumgebung zu verwenden. Die Verwendung von Produktionsschlüsseln und -daten (PAN, BDK-ID usw.) in einer Umgebung außerhalb der Produktion kann sich auf Ihren Compliance-Umfang auswirken, z. B. für PCI DSS und PCI P2PE.

Themen

- Daten verschlüsseln, entschlüsseln und erneut verschlüsseln
- Kartendaten generieren und verifizieren
- PIN-Daten generieren, übersetzen und verifizieren
- Generieren und verifizieren Sie ARQC
- MAC generieren und verifizieren
- Gültige Schlüssel für kryptografische Operationen

Daten verschlüsseln, entschlüsseln und erneut verschlüsseln

Verschlüsselungs- und Entschlüsselungsmethoden können verwendet werden, um Daten mithilfe einer Vielzahl von symmetrischen und asymmetrischen Techniken wie TDES, AES und RSA zu

verschlüsseln oder zu entschlüsseln. Diese Methoden unterstützen auch Schlüssel, die mit DUKPT- und EMV-Techniken abgeleitet wurden. Für Anwendungsfälle, in denen Sie Daten unter einem neuen Schlüssel sichern möchten, ohne die zugrunde liegenden Daten offenzulegen, kann der ReEncrypt Befehl auch verwendet werden.

> **Note**
>
> Bei der Verwendung der encrypt/decrypt Funktionen wird davon ausgegangen, dass alle Eingaben in HexBinary vorliegen. Beispielsweise wird ein Wert von 1 als 31 (Hex) eingegeben und ein kleingeschriebenes t wird als 74 (Hex) dargestellt. Alle Ausgaben sind ebenfalls in HexBinary.

Einzelheiten zu allen verfügbaren Optionen finden Sie im API-Leitfaden für Encrypt, Decrypt and Re-Encrypt.

Themen

- Daten verschlüsseln
- Daten entschlüsseln

Daten verschlüsseln

Die `Encrypt Data` API wird verwendet, um Daten mit symmetrischen und asymmetrischen Datenverschlüsselungsschlüsseln sowie von DUKPT und EMV abgeleiteten Schlüsseln zu verschlüsseln. Verschiedene Algorithmen und Varianten werden unterstützt, darunter, und. `TDES` `RSA` `AES`

Die wichtigsten Eingaben sind der Verschlüsselungsschlüssel, der zur Verschlüsselung der Daten verwendet wird, die Klartextdaten im HexBinary-Format, die verschlüsselt werden sollen, und Verschlüsselungsattribute wie Initialisierungsvektor und Modus für Blockchiffren wie TDES. Die Klartextdaten müssen ein Vielfaches von 8 Byte für`TDES`, 16 Byte für `AES` und der Länge des Schlüssels im Fall von sein. `RSA` Symmetrische Tasteneingaben (TDES, AES, DUKPT, EMV) sollten aufgefüllt werden, falls die Eingabedaten diese Anforderungen nicht erfüllen. Die folgende Tabelle zeigt die maximale Klartext-Länge für jeden Schlüsseltyp und den Fülltyp, den Sie für RSA-Schlüssel definieren. `EncryptionAttributes`

Füllungstyp	RSA_2048	RSA_3072	RSA_4096
OAEP SHA1	428	684	940
OAEP SHA256	380	636	892
OAEP SHA512	252	508	764
PKCS1	488	744	1000
None	488	744	1000

Die primären Ausgaben enthalten die verschlüsselten Daten als Chiffretext im HexBinary-Format und den Prüfsummenwert für den Verschlüsselungsschlüssel. Einzelheiten zu allen verfügbaren Optionen finden Sie im API-Leitfaden für Encrypt.

Beispiele

- Verschlüsseln Sie Daten mit einem symmetrischen AES-Schlüssel
- Verschlüsseln Sie Daten mit dem DUKPT-Schlüssel
- Verschlüsseln Sie Daten mit einem von EMV abgeleiteten symmetrischen Schlüssel
- Verschlüsseln Sie Daten mit einem RSA-Schlüssel

Verschlüsseln Sie Daten mit einem symmetrischen AES-Schlüssel

> **Note**
>
> Alle Beispiele gehen davon aus, dass der entsprechende Schlüssel bereits existiert. Schlüssel können mithilfe der CreateKeyOperation erstellt oder mithilfe der ImportKeyOperation importiert werden.

Example

In diesem Beispiel verschlüsseln wir Klartextdaten mit einem symmetrischen Schlüssel, der mithilfe der Operation erstellt oder mithilfe der CreateKeyOperation importiert wurde. ImportKey Für diesen Vorgang muss der Schlüssel auf `Encrypt` und KeyUsage auf KeyModesOfUse gesetzt sein. `TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY` Weitere Optionen finden Sie unter Schlüssel für kryptografische Operationen.

```
$ 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"
}
```

Verschlüsseln Sie Daten mit dem DUKPT-Schlüssel

Example

In diesem Beispiel verschlüsseln wir Klartextdaten mit einem DUKPT-Schlüssel. AWS Zahlungskryptografie unterstützt TDES und DUKPT-Schlüssel. AES Für diesen Vorgang muss der Schlüssel auf DeriveKey und KeyUsage auf KeyModesOfUse gesetzt sein. TR31_B0_BASE_DERIVATION_KEY Weitere Optionen finden Sie unter Schlüssel für kryptografische Operationen.

```
$ 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"
}
```

Verschlüsseln Sie Daten mit einem von EMV abgeleiteten symmetrischen Schlüssel

Example

In diesem Beispiel verschlüsseln wir Klartextdaten mit einem von EMV abgeleiteten symmetrischen Schlüssel, der bereits erstellt wurde. Sie könnten einen Befehl wie diesen verwenden, um Daten an eine EMV-Karte zu senden. Für diesen Vorgang muss KeyModesOfUse der Schlüssel auf Derive und auf TR31_E1_EMV_MKEY_CONFIDENTIALITY oder KeyUsage TR31_E6_EMV_MKEY_OTHER gesetzt sein. Weitere Informationen finden Sie unter Schlüssel für kryptografische Operationen.

```
$ 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"
}
```

Verschlüsseln Sie Daten mit einem RSA-Schlüssel

Example

In diesem Beispiel verschlüsseln wir Klartextdaten mit einem öffentlichen RSA-Schlüssel, der mit der Operation importiert wurde. ImportKey Für diesen Vorgang muss der Schlüssel auf `Encrypt` und KeyUsage auf KeyModesOfUse gesetzt sein. `TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION` Weitere Optionen finden Sie unter Schlüssel für kryptografische Operationen.

Für PKCS #7 oder andere Padding-Schemata, die derzeit nicht unterstützt werden, bewerben Sie sich bitte, bevor Sie den Service aufrufen, und wählen Sie kein Padding aus, indem Sie den Padding-Indikator 'Asymmetric= {} 'weglassen

```
$ aws payment-cryptography-data encrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/thfezpmsalcfwmsg
 --plain-text 313233343132333431323334313233343132333431323334 --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"
}
```

Daten entschlüsseln

Die `Decrypt Data` API wird verwendet, um Daten mit symmetrischen und asymmetrischen Datenverschlüsselungsschlüsseln sowie von DUKPT und EMV abgeleiteten Schlüsseln zu entschlüsseln. Verschiedene Algorithmen und Varianten werden unterstützt, darunter, und. `TDES` `RSA` `AES`

Die wichtigsten Eingaben sind der Entschlüsselungsschlüssel, der zum Entschlüsseln der Daten verwendet wird, die Chiffretextdaten im HexBinary-Format, die entschlüsselt werden sollen, und Entschlüsselungsattribute wie Initialisierungsvektor, Modus als Blockchiffren usw. Die primären Ausgaben umfassen die entschlüsselten Daten als Klartext im HexBinary-Format und den Prüfsummenwert für den Entschlüsselungsschlüssel. Einzelheiten zu allen verfügbaren Optionen finden Sie im API-Leitfaden für Decrypt.

Beispiele

- Entschlüsseln Sie Daten mit dem symmetrischen AES-Schlüssel
- Entschlüsseln Sie Daten mit dem DUKPT-Schlüssel
- Entschlüsseln Sie Daten mithilfe eines von EMV abgeleiteten symmetrischen Schlüssels
- Entschlüsseln Sie Daten mit einem RSA-Schlüssel

Entschlüsseln Sie Daten mit dem symmetrischen AES-Schlüssel

Example

In diesem Beispiel werden wir Chiffretextdaten mit einem symmetrischen Schlüssel entschlüsseln. Dieses Beispiel zeigt einen AES Schlüssel, aber TDES_2KEY er wird auch unterstützt. TDES_3KEY Für diesen Vorgang muss der Schlüssel auf KeyModesOfUse gesetzt Decrypt und auf KeyUsage gesetzt seinTR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY. Weitere Optionen finden Sie unter Schlüssel für kryptografische Operationen.

```
$ 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"
}
```

Entschlüsseln Sie Daten mit dem DUKPT-Schlüssel

> **Note**
>
> Die Verwendung von Entschlüsselungsdaten mit DUKPT für P2PE-Transaktionen kann dazu führen, dass Kreditkarten-PAN und andere Karteninhaberdaten an Ihre Anwendung zurückgegeben werden, die bei der Bestimmung des PCI-DSS-Umfangs berücksichtigt werden müssen.

Example

In diesem Beispiel werden wir Chiffretextdaten mithilfe eines DUKPT-Schlüssels entschlüsseln, der mit der Operation erstellt oder mit der Operation importiert wurde. CreateKeyImportKey Für diesen Vorgang muss der Schlüssel auf gesetzt und auf KeyModesOfUse gesetzt sein`DeriveKey`. KeyUsage `TR31_B0_BASE_DERIVATION_KEY` Weitere Optionen finden Sie unter Schlüssel für kryptografische Operationen. Wenn Sie für den `TDES` Algorithmus verwendenDUKPT, muss die Länge der Chiffretext-Daten ein Vielfaches von 16 Byte sein. Für `AES` Algorithmen muss die Länge der Chiffretext-Daten ein Vielfaches von 32 Byte sein.

```
$ 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"
}
```

Entschlüsseln Sie Daten mithilfe eines von EMV abgeleiteten symmetrischen Schlüssels

Example

In diesem Beispiel werden wir Chiffretextdaten mit einem von EMV abgeleiteten symmetrischen Schlüssel entschlüsseln, der mit der Operation erstellt oder mit der Operation importiert wurde. CreateKeyImportKey Für diesen Vorgang muss der Schlüssel auf und auf oder KeyModesOfUse gesetzt sein. `Derive` KeyUsage `TR31_E1_EMV_MKEY_CONFIDENTIALITY` `TR31_E6_EMV_MKEY_OTHER` Weitere Informationen finden Sie unter Schlüssel für kryptografische Operationen.

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

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

Entschlüsseln Sie Daten mit einem RSA-Schlüssel

Example

In diesem Beispiel werden wir Chiffretextdaten mit einem RSA-Schlüsselpaar entschlüsseln, das mit der Operation erstellt wurde. CreateKey Für diesen Vorgang muss der Schlüssel auf aktiviert `Decrypt` und auf KeyModesOfUse gesetzt sein. KeyUsage `TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION` Weitere Optionen finden Sie unter Schlüssel für kryptografische Operationen.

Für PKCS #7 oder andere Padding-Schemata, die derzeit nicht unterstützt werden, wählen Sie bitte kein Padding aus, indem Sie den Padding-Indikator 'Asymmetric= {} 'weglassen und das Padding nach dem Aufrufen des Dienstes entfernen.

```
$ 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": "31323334313233343132333431323334"
}
```

Kartendaten generieren und verifizieren

Das Generieren und Überprüfen von Kartendaten umfasst Daten, die aus Kartendaten abgeleitet wurden, z. B. CVV CVV2, CVC und CVV.

Themen

- Kartendaten generieren
- Überprüfen Sie die Kartendaten

Kartendaten generieren

Die `Generate Card Data` API wird verwendet, um Kartendaten mithilfe von Algorithmen wie CVV CVV2 oder Dynamic CVV2 zu generieren. Welche Schlüssel für diesen Befehl verwendet werden können, finden Sie im Abschnitt Gültige Schlüssel für kryptografische Operationen.

Viele kryptografische Werte wie CVV, iCVV CVV2, CAV V7 verwenden denselben kryptografischen Algorithmus, variieren jedoch die Eingabewerte. Zum Beispiel hat CardVerificationValue1 Eingaben für Kartennummer und Ablaufdatum. ServiceCode Während CardVerificationValue2 nur zwei dieser Eingaben hat, liegt das daranCVC2, dass für CVV2/der auf 000 festgelegt ServiceCode ist. In ähnlicher Weise ServiceCode ist der Wert für iCVV auf 999 festgelegt. Einige Algorithmen können die vorhandenen Felder wiederverwenden, z. B. CAVV V8. In diesem Fall müssen Sie die korrekten Eingabewerte im Handbuch Ihres Anbieters nachlesen.

> **Note**
>
> Das Ablaufdatum muss für die Generierung und Validierung im gleichen Format eingegeben werden (z. B. MMYY oder YYMM), damit korrekte Ergebnisse erzielt werden.

Generieren CVV2

Example

In diesem Beispiel generieren wir eine CVV2 für eine bestimmte PAN mit den Eingaben von PAN und dem Ablaufdatum der Karte. Dies setzt voraus, dass Sie einen Kartenbestätigungsschlüssel generiert haben.

```
$ 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"
  }
```

Generieren Sie iCVV

Example

In diesem Beispiel generieren wir ein iCVV für eine bestimmte PAN mit Eingaben vonPAN, einem Servicecode von 999 und dem Ablaufdatum der Karte. Dies setzt voraus, dass Sie einen Kartenbestätigungsschlüssel generiert haben.

Alle verfügbaren Parameter finden Sie unter CardVerificationValue1 im API-Referenzhandbuch.

```
$ 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"
}
```

Überprüfen Sie die Kartendaten

`Verify Card Data`wird verwendet, um Daten zu verifizieren, die mithilfe von Zahlungsalgorithmen erstellt wurden, die auf Verschlüsselungsprinzipien basieren, wie `DISCOVER_DYNAMIC_CARD_VERIFICATION_CODE` z.

Die Eingabewerte werden in der Regel im Rahmen einer eingehenden Transaktion an einen Emittenten oder einen unterstützenden Plattformpartner weitergegeben. Informationen zur Überprüfung eines ARQC-Kryptogramms (das für EMV-Chipkarten verwendet wird) finden Sie unter ARQC verifizieren.

Weitere Informationen finden Sie im API-Leitfaden. VerifyCardValidationData

Wenn der Wert verifiziert ist, gibt die API http/200 zurück. Wenn der Wert nicht verifiziert ist, wird http/400 zurückgegeben.

Verifizieren CVV2

Example

In diesem Beispiel validieren wir einen CVV/ CVV2 für eine bestimmte PAN. Das CVV2 wird in der Regel vom Karteninhaber oder Benutzer während der Transaktion zur Validierung bereitgestellt. Um ihre Eingaben zu validieren, werden zur Laufzeit die folgenden Werte bereitgestellt: Key to Use for Validation (CVK)PAN, Ablaufdatum der Karte und CVV2 Eingabe. Das Kartenablaufformat muss mit dem Format übereinstimmen, das bei der ursprünglichen Wertgenerierung verwendet wurde.

Alle verfügbaren Parameter finden Sie unter CardVerificationValue2 im API-Referenzhandbuch.

```
$ 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"
  }
```

Überprüfen Sie iCVV

Example

In diesem Beispiel verifizieren wir ein iCVV für eine bestimmte PAN mit Eingaben von Key to Use for Validation (CVK), einem Servicecode von 999PAN, dem Ablaufdatum der Karte und dem iCVV, das von der zu validierenden Transaktion bereitgestellt wurde.

iCVV ist kein vom Benutzer eingegebener Wert (wie CVV2), sondern auf einer EMV-Karte eingebettet. Es sollte geprüft werden, ob es immer validiert werden sollte, wenn es bereitgestellt wird.

Alle verfügbaren Parameter finden Sie unter CardVerificationValue1 im API-Referenzhandbuch.

```
$ 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-Daten generieren, übersetzen und verifizieren

Mit den PIN-Datenfunktionen können Sie zufällige Pins und PIN-Bestätigungswerte (PVV) generieren und eingehende verschlüsselte Pins anhand von PVV- oder PIN-Offsets validieren.

Mit der Pin-Übersetzung können Sie einen Pin von einer Funktionstaste auf eine andere übertragen, ohne dass der Pin im Klartext angezeigt wird, wie in PCI-PIN-Anforderung 1 festgelegt.

> **Note**
>
> Da die PIN-Generierung und -Validierung in der Regel Funktionen des Ausstellers sind und die PIN-Übersetzung eine typische Acquirer-Funktion ist, empfehlen wir, den Zugriff mit den

geringsten Zugriffsrechten in Betracht zu ziehen und Richtlinien festzulegen, die für den Anwendungsfall Ihres Systems geeignet sind.

Themen

- PIN-Daten übersetzen
- Generieren Sie PIN-Daten
- Überprüfen Sie die PIN-Daten

PIN-Daten übersetzen

Die Funktionen zum Übersetzen von PIN-Daten werden verwendet, um verschlüsselte PIN-Daten von einem Schlüsselsatz in einen anderen zu übersetzen, ohne dass die verschlüsselten Daten das HSM verlassen. Es wird für die P2PE-Verschlüsselung verwendet, bei der sich die Arbeitsschlüssel ändern sollten, das Verarbeitungssystem die Daten jedoch entweder nicht entschlüsseln muss oder darf. Die Haupteingaben sind die verschlüsselten Daten, der zum Verschlüsseln der Daten verwendete Verschlüsselungsschlüssel und die Parameter, die zum Generieren der Eingabewerte verwendet werden. Die anderen Eingaben sind die angeforderten Ausgabeparameter, z. B. der Schlüssel, der zum Verschlüsseln der Ausgabe verwendet werden soll, und die Parameter, die zum Erstellen dieser Ausgabe verwendet werden. Die primären Ausgaben sind ein neu verschlüsselter Datensatz sowie die Parameter, die zu seiner Generierung verwendet wurden.

> **Note**
>
> Für die PCI-Konformität müssen die eingehenden und ausgehenden PrimaryAccountNumber Werte übereinstimmen. Das Übersetzen einer PIN von einer PAN in eine andere ist nicht zulässig.

Themen

- PIN von PEK nach DUKPT
- PIN von PEK zu PEK

PIN von PEK nach DUKPT

Example

In diesem Beispiel übersetzen wir eine PIN aus einem AES-ISO-4-PIN-Block mithilfe der DUKPT - in die PEK-TDES-Verschlüsselung unter Verwendung des ISO-0-PIN-Blocks. Dies ist häufig der Fall, wenn ein Zahlungsterminal eine PIN in ISO 4 verschlüsselt und diese dann zur Weiterverarbeitung zurück in TDES übersetzt wird, falls die nächste Verbindung AES noch nicht unterstützt.

```
$ 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"
        }
```

PIN von PEK zu PEK

Example

In diesem Beispiel übersetzen wir eine unter einem PEK (PIN Encryption Key) verschlüsselte PIN in ein anderes PEK. Dies wird häufig verwendet, wenn Transaktionen zwischen verschiedenen Systemen oder Partnern weitergeleitet werden, die unterschiedliche Verschlüsselungsschlüssel verwenden, während gleichzeitig die PCI-PIN-Konformität gewahrt bleibt, indem die PIN während des gesamten Vorgangs verschlüsselt bleibt. Beide Schlüssel verwenden in diesem Beispiel die TDES 3KEY-Verschlüsselung, es sind jedoch eine Vielzahl von Optionen verfügbar, darunter AES ISO-4 zu TDES ISO-0, DUKPT zu PEK oder AS2805 zu PEK.

```
$ 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"
}
```

Der Ausgangs-PIN-Block ist jetzt unter dem zweiten PEK verschlüsselt und kann sicher an das nachgelagerte System übertragen werden, das den entsprechenden Schlüssel enthält.

Generieren Sie PIN-Daten

Funktionen zum Generieren von PIN-Daten werden zur Generierung von PIN-related Werten wie PVV - und Pin-Block-Offsets verwendet, die zur Validierung der PIN-Eingabe durch Benutzer während der Transaktion oder Autorisierung verwendet werden. Diese API kann auch mithilfe verschiedener Algorithmen eine neue zufällige PIN generieren.

Generieren Sie eine zufällige PIN und einen passenden Visa PVV

Example

In diesem Beispiel generieren wir einen neuen (zufälligen) Pin, wobei die Ausgänge ein verschlüsselter `PIN block` (PinData.PinBlock) und ein PVV (PinData.Offset) sind. Die wichtigsten Eingaben sindPAN, the`Pin Verification Key`, the `Pin Encryption Key` und the`PIN block format`.

Dieser Befehl erfordert, dass der Schlüssel vom Typ ist`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"
            }
        }
```

Generieren Sie ein Visa-PVV für eine bekannte PIN

Example

In diesem Beispiel generieren wir eine PVV für eine bestimmte (verschlüsselte) PIN. Eine verschlüsselte PIN kann im Vorfeld empfangen werden, z. B. von einem Zahlungsterminal oder von einem Karteninhaber, indem der vom Benutzer auswählbare PIN-Flow verwendet wird. Die wichtigsten Eingaben sindPAN, derPin Verification Key, derPin Encryption Key, der Encrypted Pin Block und der. 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"
            }
        }
```

Generieren Sie den IBM3624-Pin-Offset für einen Pin

IBM 3624 PIN Offset wird manchmal auch als IBM-Methode bezeichnet. Diese Methode generiert eine natural/intermediate PIN anhand der Validierungsdaten (in der Regel die PAN) und eines PIN-Schlüssels (PVK). Natürliche Pins sind quasi ein abgeleiteter Wert, und da sie deterministisch sind, sind sie für den Aussteller sehr effizient zu handhaben, da keine PIN-Daten auf der Ebene des Karteninhabers gespeichert werden müssen. Der offensichtlichste Nachteil ist, dass dieses Schema die vom Karteninhaber auswählbaren oder zufälligen PINs nicht berücksichtigt. Um diese Arten von Pins zu berücksichtigen, wurde dem Schema ein Offset-Algorithmus hinzugefügt. Der Offset stellt den Unterschied zwischen dem vom Benutzer ausgewählten (oder zufälligen) Pin und dem natürlichen

Schlüssel dar. Der Offsetwert wird vom Kartenaussteller oder Kartenprozessor gespeichert. Zum Zeitpunkt der Transaktion berechnet der AWS Payment Cryptography Service intern den natürlichen Pin neu und wendet den Offset an, um den Pin zu finden. Dieser Wert wird dann mit dem Wert verglichen, der in der Transaktionsautorisierung angegeben wurde.

Für IBM3624 gibt es mehrere Optionen:

- `Ibm3624NaturalPin`gibt den natürlichen Pin und einen verschlüsselten Pin-Block aus
- `Ibm3624PinFromOffset`generiert bei gegebenem Offset einen verschlüsselten Pin-Block
- `Ibm3624RandomPin`generiert einen zufälligen Pin und dann den passenden Offset und den verschlüsselten Pin-Block.
- `Ibm3624PinOffset`generiert den Pin-Offset bei einem vom Benutzer ausgewählten Pin.

Im Rahmen der AWS Zahlungskryptografie werden die folgenden Schritte ausgeführt:

- Füllen Sie den bereitgestellten Plan auf 16 Zeichen aus. Wenn <16 angegeben sind, füllen Sie die rechte Seite mit dem mitgelieferten Füllzeichen auf.
- Verschlüsselt die Validierungsdaten mithilfe des PIN-Generierungsschlüssels.
- Dezimalisieren Sie die verschlüsselten Daten mithilfe der Dezimalisierungstabelle. Dadurch werden Hexadezimalziffern Dezimalziffern zugeordnet, zum Beispiel kann 'A' 9 und 1 1 1 zugeordnet werden.
- Ermittelt die ersten 4 Ziffern aus einer Hexadezimaldarstellung der Ausgabe. Das ist der natürliche Pin.
- Wenn ein vom Benutzer ausgewählter Pin oder ein zufälliger Pin generiert wurde, subtrahiert Modulo den natürlichen Pin vom Kunden-Pin. Das Ergebnis ist der Pin-Offset.

Beispiele

- Beispiel: Generieren Sie den IBM3624-Pin-Offset für einen Pin

Beispiel: Generieren Sie den IBM3624-Pin-Offset für einen Pin

In diesem Beispiel generieren wir einen neuen (zufälligen) Pin, bei dem die Ausgänge ein verschlüsselter `PIN block` (PinData.PinBlock) und ein `IBM3624` Offsetwert (Pin) sind. Data.Offset Bei den Eingaben handelt es sich `PAN` um Validierungsdaten (in der Regel das Pan), das Füllzeichen, das`Pin Verification Key`, das `Pin Encryption Key` und das`PIN block format`.

Für diesen Befehl muss der PIN-Generierungsschlüssel vom Typ TR31_V1_IBM3624_PIN_VERIFICATION_KEY und der Verschlüsselungsschlüssel vom Typ A sein TR31_P0_PIN_ENCRYPTION_KEY

Example

Das folgende Beispiel zeigt, wie ein zufälliger Pin generiert und dann der verschlüsselte Pin-Block und der IBM3624-Offsetwert ausgegeben werden 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"
             }
         }
```

Überprüfen Sie die PIN-Daten

Die Funktionen zur Überprüfung der PIN-Daten werden verwendet, um zu überprüfen, ob eine PIN korrekt ist. Dabei wird in der Regel der zuvor gespeicherte PIN-Wert mit dem verglichen, den der Karteninhaber an einem POI eingegeben hat. Diese Funktionen vergleichen zwei Werte, ohne den zugrunde liegenden Wert einer der beiden Quellen offenzulegen.

Validieren Sie die verschlüsselte PIN mit der PVV-Methode

Example

In diesem Beispiel validieren wir eine PIN für eine bestimmte PAN. Die PIN wird in der Regel vom Karteninhaber oder Benutzer während der Transaktion zur Validierung bereitgestellt und mit dem hinterlegten Wert verglichen (die Eingabe des Karteninhabers wird als verschlüsselter Wert vom Terminal oder einem anderen Upstream-Anbieter bereitgestellt). Um diese Eingabe zu validieren, werden während der Laufzeit auch die folgenden Werte bereitgestellt: Der Schlüssel, der zur Verschlüsselung der eingegebenen PIN verwendet wird (dies ist häufig ein`IWK`), PAN und der Wert, mit dem verifiziert werden soll (entweder ein `PVV` oder). `PIN offset`

Wenn AWS Payment Cryptography die PIN validieren kann, http/200 wird eine zurückgegeben. Wenn die PIN nicht validiert wird, wird eine http/400 zurückgegeben.

```
$ 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",

        }
```

Bestätigen Sie die verschlüsselte PIN mit der PVV-Methode - Fehler: falsche PIN

Example

In diesem Beispiel werden wir versuchen, eine PIN für eine bestimmte PAN zu validieren, aber es wird fehlschlagen, weil die PIN falsch ist.

Bei Verwendung von SDKs erscheint dies als {"Message“ :"PIN-Block-Verifizierung fehlgeschlagen.“ , „Grund“ :"INVALID_PIN "}

```
$ 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.
```

Bestätigen Sie die verschlüsselte PIN mit der PVV-Methode - Fehler, falsche Eingaben

Example

In diesem Beispiel werden wir versuchen, eine PIN für eine bestimmte PAN zu validieren, aber es wird aufgrund falscher Eingaben fehlschlagen und die eingehenden Daten waren keine gültige PIN. Häufige Ursachen sind: 1/wrong Der verwendete Schlüssel, 2/input Parameter wie das Pan- oder Pin-Blockformat sind falsch, der 3/pin Block ist beschädigt.

```
$ 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.
```

Überprüfen Sie eine PIN anhand des zuvor gespeicherten IBM3624-Pin-Offsets

In diesem Beispiel validieren Sie eine vom Karteninhaber angegebene PIN anhand des PIN-Offsets, der in der Datei beim Kartenaussteller oder Prozessor gespeichert ist. Zu den Eingaben gehören der Bestätigungsschlüssel (PVK), der Verschlüsselungsschlüssel (PEK), die primäre Kontonummer, der verschlüsselte PIN-Block vom Zahlungsterminal (oder einem anderen Upstream-Anbieter wie einem Kartennetzwerk) und der zuvor gespeicherte PIN-Offset. Wenn die PIN übereinstimmt, gibt die API eine HTTP/200 Antwort zurück.

Für diesen Befehl muss der PIN-Bestätigungsschlüssel vom Typ `TR31_V1_IBM3624_PIN_VERIFICATION_KEY` und der Verschlüsselungsschlüssel vom Typ A sein`TR31_P0_PIN_ENCRYPTION_KEY`.

> ⚠ Überprüfung der PIN-Länge
>
> Der optionale `--pin-data-length` Parameter gibt die Länge der PIN an, die überprüft wird (4 bis 12 Ziffern). Wenn Sie diesen Parameter nicht angeben, validiert der Dienst die

gesamte PIN-Länge, die durch den Offsetwert geschützt ist. Wenn Sie nur einen Teil der PIN überprüfen möchten, verwenden Sie den `--pin-data-length` Parameter. Wenn der Wert beispielsweise auf 4 gesetzt `--pin-data-length` ist, die tatsächliche PIN jedoch aus 6 Ziffern besteht, validiert der Dienst nur die 4 Ziffern ganz rechts. Die verbleibenden 2 Ziffern werden nicht geprüft. Sie könnten dies mit Absicht tun, wenn die PIN des Karteninhabers aus 6 Ziffern besteht, die eingehende PIN jedoch nur aus 4 Ziffern besteht, z. B. wenn das eingehende Netzwerk oder der Händler nur eine 4-stellige PIN unterstützt.

Example

```
$ 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 \
    --encrypted-pin-block AC17DC148BDA645E \
    --pin-data-length 4 \
    --verification-attributes
 Ibm3624PinOffset="{DecimalizationTable=9876543210654321,PinValidationDataPadCharacter=D,PinVal
```

```
{
    "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"
}
```

Generieren und verifizieren Sie ARQC

Ein Authorization Request Cryptogram (ARQC) ist ein Kryptogramm, das von einer EMV-Chipkarte (oder einem digitalen Äquivalent wie einer mobilen Geldbörse) während eines Zahlungsvorgangs generiert wird. Das ARQC validiert die Karte kryptografisch und gewährleistet die Integrität der Transaktionsdaten. Es umfasst Kartendaten, Terminaldaten und transaktionsspezifische Daten, um für jede Transaktion einen eindeutigen Wert zu ermitteln.

AWS Die Zahlungskryptografie unterstützt zwei ARQC-Operationen. Sie können einen ARQC für Entwicklungs- und Testzwecke generieren und einen ARQC für die Autorisierung von Produktionstransaktionen verifizieren.

Themen

- Kryptogramm für Authentifizierungsanfragen (ARQC) generieren
- Kryptogramm für Authentifizierungsanfragen (ARQC) verifizieren

Kryptogramm für Authentifizierungsanfragen (ARQC) generieren

Die Kryptogramm-API zum Generieren von Authentifizierungsanforderungen wird zur Generierung von ARQC verwendet. Mit dieser API können Sie ARQC mithilfe von AWS Payment Cryptography für Entwicklungs- und Testzwecke generieren.

> **Important**
>
> Dieser Vorgang ist nur für Entwicklungs- und Testszenarien vorgesehen. Es wird nicht empfohlen, diesen Vorgang als Ersatz für die kartengestützte Kryptogrammgenerierung im Produktionszahlungsfluss zu verwenden.

Alle verfügbaren Optionen finden Sie GenerateAuthRequestCryptogramin der API-Referenz.

ARQC-Kryptogramme erfordern in der Regel die folgenden Eingaben (obwohl dies je nach Implementierung variieren kann):

- PAN — Im Feld angegeben PrimaryAccountNumber
- PAN-Sequenznummer (PSN) — Im PanSequenceNumber Feld angegeben
- Methode zur Schlüsselableitung wie Common Session Key (CSK) — Spezifiziert in SessionKeyDerivationAttributes
- Hauptschlüsselableitungsmodus (z. B. EMV-Option A) — Spezifiziert in MajorKeyDerivationMode
- Transaktionsdaten — Eine Reihe von Transaktions-, Terminal- und Kartendaten wie Betrag und Datum. Im TransactionData Feld angegeben.
- Hauptschlüssel des Ausstellers — Der Hauptschlüssel, der zur Ableitung des Kryptogrammschlüssels (AC) verwendet wird. Dieser Schlüssel schützt einzelne Transaktionen und ist in dem Feld angegeben. KeyIdentifier

Weitere Informationen zum Erstellen und Hinzufügen von Transaktionsdaten finden Sie unter Verify Auth Request (ARQC) -Kryptogramm.

Themen

- Beispiele

Beispiele

Visum CVN 10

Example

Das folgende Beispiel generiert einen ARQC mit Visa CVN10.

Wenn AWS Payment Cryptography den ARQC erfolgreich generiert, wird ein http/200 mit dem generierten Kryptogramm zurückgegeben.

```
$ aws payment-cryptography-data generate-auth-request-cryptogram \
--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",
  "AuthRequestCryptogram": "D791093C8A921769"
}
```

Visa CVN-18 und Visa CVN-22

Example

Das folgende Beispiel generiert einen ARQC mit Visa CVN18 oder CVN22. Die kryptografischen Operationen zwischen CVN18 und CVN22 sind dieselben, aber die in den Transaktionsdaten enthaltenen Daten variieren. Im Vergleich zu CVN10 wird selbst bei denselben Eingaben ein völlig anderes Kryptogramm generiert.

Wenn AWS Payment Cryptography den ARQC erfolgreich generiert, http/200 wird ein mit dem generierten Kryptogramm zurückgegeben.

```
$ aws payment-cryptography-data generate-auth-request-cryptogram \
--key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
--major-key-derivation-mode EMV_OPTION_A \
--transaction-data
 0000000017000000000000000840008000800008401605170000000093800000B1F22010300000000000
 \
00000000000000000000000000000000000000000800000000000000 \
--session-key-derivation-attributes='{"EmvCommon":
{"ApplicationTransactionCounter":"000B", \
"PanSequenceNumber":"01","PrimaryAccountNumber":"9137631040001422"}}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk",
  "KeyCheckValue": "08D7B4",
  "AuthRequestCryptogram": "61EDCC708B4C97B4"
}
```

Kryptogramm für Authentifizierungsanfragen (ARQC) verifizieren

Die Kryptogramm-API für Verify Auth Requests wird zur Überprüfung von ARQC verwendet. Die Generierung des ARQC erfolgt in der Regel während der Transaktionsautorisierung auf einer EMV-Chipkarte (oder einem digitalen Äquivalent wie einer mobilen Geldbörse), obwohl der Dienst Generierungen für Entwicklungszwecke bereitstellen kann. Ein ARQC ist für jede Transaktion einzigartig und dient dazu, sowohl die Gültigkeit der Karte kryptografisch nachzuweisen als auch sicherzustellen, dass die Transaktionsdaten exakt mit der aktuellen (erwarteten) Transaktion übereinstimmen.

AWS Die Zahlungskryptografie bietet eine Vielzahl von Optionen zur Validierung von ARQC und zur Generierung optionaler ARPC-Werte, einschließlich der in EMV 4.4 Buch 2 definierten und anderen von Visa und Mastercard verwendeten Schemata. Eine vollständige Liste aller verfügbaren Optionen finden Sie im Abschnitt im VerifyCardValidationData API-Leitfaden.

ARQC-Kryptogramme erfordern normalerweise die folgenden Eingaben (obwohl dies je nach Implementierung variieren kann):

- PAN — Im Feld angegeben PrimaryAccountNumber
- PAN-Sequenznummer (PSN) — im PanSequenceNumber Feld angegeben
- Methode zur Schlüsselableitung wie Common Session Key (CSK) — Spezifiziert in SessionKeyDerivationAttributes
- Hauptschlüsselableitungsmodus (z. B. EMV-Option A) — Spezifiziert in MajorKeyDerivationMode
- Transaktionsdaten — eine Zeichenfolge verschiedener Transaktions-, Terminal- und Kartendaten wie Betrag und Datum —, die im Feld angegeben sind TransactionData
- Hauptschlüssel des Ausstellers — der Hauptschlüssel, der zur Ableitung des Kryptogrammschlüssels (AC) verwendet wird, der zum Schutz einzelner Transaktionen verwendet und im Feld angegeben ist KeyIdentifier

Themen

- Transaktionsdaten erstellen
- Auffüllen von Transaktionsdaten
- Beispiele

Transaktionsdaten erstellen

Der genaue Inhalt (und die Reihenfolge) des Transaktionsdatenfeldes variiert je nach Implementierung und Netzwerkschema, aber die empfohlenen Mindestfelder (und die Verkettungsreihenfolge) sind in EMV 4.4, Buch 2, Abschnitt 8.1.1 — Datenauswahl, definiert. Wenn die ersten drei Felder Betrag (17,00), sonstiger Betrag (0,00) und Land des Kaufs lauten, würde dies dazu führen, dass die Transaktionsdaten wie folgt beginnen würden:

- 000000001700 — Betrag — 12 Stellen impliziert eine zweistellige Dezimalzahl
- 000000000000 — anderer Betrag — 12 Stellen impliziert eine zweistellige Dezimalzahl
- 0124 — vierstelliger Ländercode

- Transaktionsdaten (teilweise) ausgeben - 00000000170000000000000000124

Auffüllen von Transaktionsdaten

Transaktionsdaten sollten vor dem Senden an den Dienst aufgefüllt werden. Die meisten Schemas verwenden das Auffüllen nach ISO 9797 Methode 2. Dabei wird an eine Hexadezimalzahl 80 gefolgt von 00 angehängt, bis das Feld ein Vielfaches der Größe des Verschlüsselungsblocks ist: 8 Byte oder 16 Zeichen für TDES und 16 Byte oder 32 Zeichen für AES. Die Alternative (Methode 1) ist nicht so üblich, verwendet aber nur 00 als Füllzeichen.

ISO 9797 Methode 1: Innenabstand

Ohne Füllung: 00000000170000000000000008400080008000084016051700000000093800000B03011203 (74 Zeichen oder 37 Byte)

Gepolstert: 00000000170000000000008400080008000084016051700000000093800000B03011203 000000 (80 Zeichen oder 40 Byte)

Polsterung nach ISO 9797 Methode 2

Ohne Füllung: 00000000170000000000000008400080008000084016051700000000093800000B1F220103000000 (80 Zeichen oder 40 Byte)

Gepolstert: 00000000170000000000000008400080008000084016051700000000093800000B1F220103000000 8000000000000000 (88 Zeichen oder 44 Byte)

Beispiele

Visum CVN10

Example

In diesem Beispiel validieren wir einen mit Visa CVN10 generierten ARQC.

Wenn AWS Payment Cryptography den ARQC validieren kann, wird ein zurückgegeben. http/200 Wenn ARCQ (Authorization Request Cryptogram) dann nicht validiert wird, wird eine Antwort zurückgegeben. http/400

```
$ 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"
}
```

Visum CVN-18 und Visum CVN-22

Example

In diesem Beispiel validieren wir einen ARQC, der mit Visa CVN18 oder CVN22 generiert wurde. Die kryptografischen Operationen zwischen CVN18 und CVN22 sind dieselben, aber die in den Transaktionsdaten enthaltenen Daten variieren. Im Vergleich zu CVN10 wird selbst bei denselben Eingaben ein völlig anderes Kryptogramm generiert.

Wenn AWS Payment Cryptography den ARQC validieren kann, wird ein zurückgegeben. http/200 Wenn der ARCQ nicht validiert ist, wird ein zurückgegeben. http/400

```
$ 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
 \
00000000000000000000000000000000000000000800000000000000
--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 generieren und verifizieren

Message Authentication Codes (MAC) werden normalerweise verwendet, um die Integrität einer Nachricht zu authentifizieren (unabhängig davon, ob sie geändert wurde). Kryptografische Hashes wie HMAC (Hash-Based Message Authentication Code) CBC-MAC und CMAC (Cipher-based Message Authentication Code) bieten durch den Einsatz von Kryptografie zusätzliche Sicherheit beim Absender der MAC. HMAC basiert auf Hash-Funktionen, während CMAC auf Blockchiffren basiert. Der Dienst unterstützt auch die ISO9797-Algorithmen 1 und 3, bei denen es sich um Typen von handelt. CBC-MACs

Alle MAC-Algorithmen dieses Dienstes kombinieren eine kryptografische Hash-Funktion und einen gemeinsamen geheimen Schlüssel. Sie nehmen eine Nachricht und einen geheimen Schlüssel, z. B. das Schlüsselmaterial in einem Schlüssel, und geben ein eindeutiges Tag oder einen eindeutigen Mac zurück. Wenn sich auch nur ein Zeichen der Nachricht ändert oder wenn sich der geheime Schlüssel ändert, ist das resultierende Tag völlig anders. Durch die Anforderung eines geheimen Schlüssels bietet kryptografische MACs auch Authentizität. Ohne den geheimen Schlüssel ist es unmöglich, einen identischen Mac zu generieren. Kryptografische MACs werden manchmal als symmetrische Signaturen bezeichnet, weil sie wie digitale Signaturen funktionieren, aber einen einzigen Schlüssel sowohl für das Signieren als auch für die Überprüfung verwenden.

AWS Die Zahlungskryptografie unterstützt verschiedene Arten von MACs:

ISO9797, ALGORITHMUS 1

Bezeichnet mit oder ISO9797_ALGORITHM1. `KeyUsage` Wenn das Feld kein Vielfaches der Blockgröße ist (8 bytes/16 Hexadezimalzeichen für TDES, 16 bytes/32 Zeichen für AES), wendet Payment Cryptography automatisch ISO9797 Padding Method 1 an. AWS Wenn andere Füllmethoden erforderlich sind, können Sie diese anwenden, bevor Sie den Dienst aufrufen.

ISO9797-ALGORITHMUS 3 (MAC für den Einzelhandel)

Bezeichnet mit oder ISO9797_ALGORITHM3. `KeyUsage` Es gelten dieselben Füllregeln wie bei Algorithmus 1.

ISO9797 ALGORITHMUS 5 (CMAC)

Wird mit TR31_M6_ISO_9797_5_CMAC_KEY bezeichnet `KeyUsage`

HMAC

Wird mit `KeyUsage` TR31_M7_HMAC_KEY bezeichnet, einschließlich HMAC_SHA224, HMAC_SHA256, HMAC_SHA384 und HMAC_SHA512

AS2805.4.1 MAC

Bezeichnet mit `KeyUsage` TR31_M0_ISO_16609_MAC_KEY. Weitere Informationen zu AS2805 finden Sie unter ???

DUPT MAC

DUKPT MAC wird in der Regel verwendet, um die Quelle und die Nutzlast von Zahlungsterminals für Nachrichten to/from zu bestätigen. Es leitet mithilfe von DUKPT-Ableitungstechniken einen Schlüssel ab und führt dann den MAC durch. Schlüssel, die mit dieser Option verwendet werden, werden mit einem von TR31_B0_BASE_DERIVATION_KEY gekennzeichnet. `KeyUsage`

EMV MAC

EMV MAC wird in der EMV-Dokumentation in der Regel als Integritätsschlüssel bezeichnet. Es leitet einen Schlüssel mithilfe von EMV-Ableitungstechniken ab und verwendet dann intern ISO9797_ALGORITHM3. Es wird normalerweise verwendet, um Ausstellerskripte zur Neuprogrammierung an eine Chipkarte zu senden. Schlüssel, die mit dieser Option verwendet werden, werden mit einem von TR31_E2_EMV_MKEY_INTEGRITY gekennzeichnet. `KeyUsage` Wenn Sie sowohl ein Skript senden als auch eine Offline-PIN aktualisieren, achten Sie darauf, dass diese beiden Operationen ausgeführt werden. GenerateMacEmvPinChange

Themen

- MAC generieren
- Überprüfen Sie den MAC

MAC generieren

Die Generate MAC API wird verwendet, um kartenbezogene Daten zu authentifizieren, z. B. Tracking-Daten von einem Kartenmagnetstreifen. Dabei werden bekannte kryptografische Schlüssel verwendet, um einen MAC (Message Authentication Code) für die Datenvalidierung zwischen sendenden und empfangenden Parteien zu generieren. Die zur Generierung von MAC verwendeten Daten umfassen Nachrichtendaten, einen geheimen MAC-Verschlüsselungsschlüssel und einen MAC-Algorithmus zur Generierung eines eindeutigen MAC-Werts für die Übertragung. Die empfangende Partei des MAC verwendet dieselben MAC-Nachrichtendaten, denselben MAC-Verschlüsselungsschlüssel und denselben Algorithmus, um einen anderen MAC-Wert für den Vergleich und die Datenauthentifizierung zu reproduzieren. Selbst wenn sich ein Zeichen der Nachricht ändert oder der zur Überprüfung verwendete MAC-Schlüssel nicht identisch ist, ist der resultierende MAC-Wert unterschiedlich. Die API unterstützt ISO 9797-1 Algorithmus 1 und ISO 9797-1 Algorithmus 3 MAC (unter Verwendung eines statischen MAC-Schlüssels und eines abgeleiteten DUKPT-Schlüssels), HMAC- und EMV-MAC-Verschlüsselungsschlüssel für diesen Vorgang.

Der Eingabewert für muss HexBinary-Daten sein. `message-data`

Weitere Informationen zu allen Optionen für diese API finden Sie unter GenerateMacund VerifyMac.

Mit dem optionalen Parameter mac-length können Sie den Ausgabewert kürzen (obwohl dies auch in Ihrem Code möglich ist). Eine Länge von 8 bezieht sich auf 8 Byte oder 16 Hex-Zeichen.

MAC-Schlüssel können entweder mit AWS Payment Cryptography per Anruf erstellt CreateKeyoder per Anruf ImportKeyimportiert werden.

> **Note**
>
> Für CMAC- und HMAC-Algorithmen ist kein Padding erforderlich. Alle anderen erfordern, dass die Daten entsprechend der Blockgröße des Algorithmus aufgefüllt werden, die einem Vielfachen von 8 Byte (16 Hexadezimalzeichen) für TDES und 16 Byte (32 Hexadezimalzeichen) für AES entspricht.

Beispiele

- Generieren Sie HMAC
- Generieren Sie MAC mit dem ISO 9797-1-Algorithmus 3
- Generieren Sie MAC mit CMAC
- Generieren Sie MAC mit DUKPT CMAC

Generieren Sie HMAC

In diesem Beispiel werden wir einen HMAC (Hash-Based Message Authentication Code) für die Kartendatenauthentifizierung mithilfe des HMAC-Algorithmus `HMAC_SHA256` und des HMAC-Verschlüsselungsschlüssels generieren. Der Schlüssel muss auf und auf KeyUsage eingestellt sein`TR31_M7_HMAC_KEY`. KeyModesOfUse `Generate` Die Hash-Länge (z. B. 256) wird bei der Erstellung des Schlüssels definiert und kann nicht geändert werden.

Der optionale Parameter mac-length kürzt den ausgegebenen MAC, obwohl dies auch außerhalb des Dienstes erfolgen kann. Dieser Wert wird in Byte angegeben, sodass bei einem Wert von 16 eine Hexadezimalzeichenfolge der Länge 32 erwartet wird.

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"
}
```

Generieren Sie MAC mit dem ISO 9797-1-Algorithmus 3

In diesem Beispiel generieren wir einen MAC mit dem ISO 9797-1-Algorithmus 3 (Retail MAC) für die Kartendatenauthentifizierung. Der Schlüssel muss auf und auf KeyUsage gesetzt seinTR31_M3_ISO_9797_3_MAC_KEY. KeyModesOfUse `Generate`

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"
}
```

Generieren Sie MAC mit CMAC

CMAC wird am häufigsten verwendet, wenn es sich bei den Schlüsseln um AES handelt, unterstützt aber auch TDES. In diesem Beispiel generieren wir einen MAC mit CMAC (ISO 9797-1 Algorithm 5) für die Kartendatenauthentifizierung mit einem AES-Schlüssel. Der Schlüssel muss auf und auf KeyUsage gesetzt sein. `TR31_M6_ISO_9797_5_CMAC_KEY` KeyModesOfUse `Generate`

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"
}
```

Generieren Sie MAC mit DUKPT CMAC

In diesem Beispiel wird ein MAC mithilfe von DUKPT (Derived Unique Key Per Transaction) mit CMAC für die Kartendatenauthentifizierung generiert. Der Schlüssel muss auf `TR31_B0_BASE_DERIVATION_KEY` und auf true KeyUsage KeyModesOfUse `DeriveKey` gesetzt sein.

DUKPT-Schlüssel leiten mithilfe eines Base Derivation Key (BDK) und einer Key Serial Number (KSN) für jede Transaktion einen eindeutigen Schlüssel ab.

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",DukptKeyVariant=BIDIRECTIONAL
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6",
    "KeyCheckValue": "C1EB8F",
    "Mac": "1F8C36E63F91E4E93DF7842BF5E2E5F7"
}
```

Überprüfen Sie den MAC

Überprüfen Sie, ob die MAC-API verwendet wird, um den MAC (Message Authentication Code) für die kartenbezogene Datenauthentifizierung zu verifizieren. Es muss derselbe Verschlüsselungsschlüssel verwendet werden, der bei der MAC-Generierung verwendet wurde, um den MAC-Wert für die Authentifizierung zu reproduzieren. Der MAC-Verschlüsselungsschlüssel kann entweder mit AWS Payment Cryptography per Anruf erstellt CreateKeyoder per Anruf importiert werden. ImportKey Die API unterstützt DUKPT MAC-, HMAC- und EMV-MAC-Verschlüsselungsschlüssel für diesen Vorgang.

Wenn der Wert verifiziert ist, kehrt der Antwortparameter zurück`Http/200`, andernfalls `MacDataVerificationSuccessful` wird eine `Http/400` entsprechende Meldung angezeigt. `Mac verification failed`

Beispiele

- Überprüfen Sie HMAC
- Überprüfen Sie den MAC mit DUKPT CMAC

Überprüfen Sie HMAC

In diesem Beispiel verifizieren wir einen HMAC (Hash-Based Message Authentication Code) für die Kartendatenauthentifizierung mithilfe des HMAC-Algorithmus `HMAC_SHA256` und des HMAC-

Verschlüsselungsschlüssels. Der Schlüssel muss auf `TR31_M7_HMAC_KEY` und KeyModesOfUse `Verify` auf true KeyUsage gesetzt sein.

Example

```
$ aws payment-cryptography-data verify-mac \
     --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6 \
     --message-data
 "3b343038383439303031303733393431353d323430383232363030303730303033f33" \
     --mac ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C \
     --verification-attributes Algorithm=HMAC_SHA256
```

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

Überprüfen Sie den MAC mit DUKPT CMAC

In diesem Beispiel verifizieren wir einen MAC mithilfe von DUKPT (Derived Unique Key Per Transaction) mit CMAC für die Kartendatenauthentifizierung. Der Schlüssel muss auf `TR31_B0_BASE_DERIVATION_KEY` und auf true KeyUsage KeyModesOfUse `DeriveKey` gesetzt sein. DUKPT-Schlüssel leiten mithilfe eines Base Derivation Key (BDK) und einer Key Serial Number (KSN) für jede Transaktion einen eindeutigen Schlüssel ab. Der Wert von DukptKeyVariant muss zwischen Sender und Empfänger übereinstimmen. REQUEST wird normalerweise von Terminal zu Backend verwendet, VERIFY von Backend zu Terminal und BIDIRECTIONAL, wenn ein einziger Schlüssel in beide Richtungen verwendet wird.

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"
}
```

Gültige Schlüssel für kryptografische Operationen

Bestimmte Schlüssel können nur für bestimmte Operationen verwendet werden. Darüber hinaus können einige Operationen die Tastenverwendungsmodi für Tasten einschränken. In der folgenden Tabelle finden Sie die zulässigen Kombinationen.

> **Note**
>
> Bestimmte Kombinationen sind zwar zulässig, können jedoch zu unbrauchbaren Situationen führen, z. B. wenn CVV-Codes generiert werden, diese dann (`generate`) aber nicht verifiziert werden können. (`verify`)

Themen

- GenerateCardData
- VerifyCardData
- GeneratePinData (für VISA/ABA Schemata)
- GeneratePinData (für IBM3624)
- VerifyPinData (für VISA/ABA Schemata)

- VerifyPinData (für IBM3624)
- Daten entschlüsseln
- Encrypt Data
- Pin-Daten übersetzen
- Generate/Verify MAC
- GenerateMacEmvPinChange
- VerifyAuthRequestCryptogram
- Import/Export Schlüssel
- Unbenutzte Schlüsseltypen

GenerateCardData

API-Endpunkt	Kryptografischer Vorgang oder Algorithmus	Erlaubte Verwendung von Schlüsseln	Zulässige r Schlüssel algorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
GenerateCardData	• AMEX_CARD_SECURITY_CODE_VERSION_1 • AMEX_CARD_SECURITY_CODE_VERSION_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	{Generieren = wahr}, {Generieren = wahr, Überprüfen = wahr}
GenerateCardData	• WERT_1 FÜR DIE KARTENÜBERPRÜFUNG • WERT_2 FÜR DIE KARTENÜBERPRÜFUNG	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2-SCHLÜSSEL	{Generieren = wahr}, {Generieren = wahr, Überprüfen = wahr}

API-Endpunkt	Kryptografischer Vorgang oder Algorithmus	Erlaubte Verwendung von Schlüsseln	Zulässige r Schlüssel algorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
GenerateCardData	• AUTHENTIFIZIERUNGSWERT DES KARTENINHABERS	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{= wahr} DeriveKey
GenerateCardData	• DYNAMISCHER_KARTENPRÜFCODE	TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS	• TDES_2-SCHLÜSSEL	{= wahr} DeriveKey
GenerateCardData	• DYNAMISCHER_KARTENVERIFIZIERUNGSWERT	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{= wahr} DeriveKey

VerifyCardData

Kryptografische Operation oder Algorithmus	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüssel algorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
• AMEX_CARD_SECURITY_CODE_VERSION_1 • AMEX_CARD_SECURITY_CODE_VERSION_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	{Generieren = wahr}, {Generieren = wahr, Überprüfen = wahr}

Kryptografische Operation oder Algorithmus	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
• WERT_1 FÜR DIE KARTENÜBERPRÜFUNG • WERT_2 FÜR DIE KARTENÜBERPRÜFUNG	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2-SCHLÜSSEL	{Generieren = wahr}, {Generieren = wahr, Überprüfen = wahr}
• AUTHENTIFIZIERUNGSWERT DES KARTENINHABERS	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{= wahr} DeriveKey
• DYNAMISCHER_KARTENPRÜFCODE	TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS	• TDES_2-SCHLÜSSEL	{= wahr} DeriveKey
• DYNAMISCHER_KARTENVERIFIZIERUNGSWERT	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{= wahr} DeriveKey

GeneratePinData (für VISA/ABA Schemata)

`VISA_PIN or VISA_PIN_VERIFICATION_VALUE`

Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
PIN-Verschlüsselungsschlüssel	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Encrypt = wahr, Wrap = wahr} • {Encrypt = wahr, Decrypt = wahr,

Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
			Wrap = wahr, Unwrap = wahr} • {= wahr} NoRestrictions
Schlüssel zur PIN-Generierung	TR31_V2_VISA_PIN_VERIFICATION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Generieren = wahr} • {Generieren = wahr, verifizieren = wahr}

GeneratePinData (für **IBM3624**)

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
PIN-Verschlüsselungsschlüssel	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	Für IBM3624_NATURAL_PIN, IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET • {Encrypt = wahr, Wrap = wahr} • {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr}

Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
			• {= wahr} NoRestrictions Für IBM3624_PIN_OFFSET • {Verschlüsseln = wahr, Unwrap = wahr} • {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr} • {= wahr} NoRestrictions
Schlüssel zur PIN-Generierung	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Generieren = wahr} • {Generieren = wahr, verifizieren = wahr}

VerifyPinData (für VISA/ABA Schemata)

`VISA_PIN`

Schlüsseltyp	Zulässige Schlüsselverwendung	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
PIN-Verschlüsselungsschlüssel	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Decrypt = wahr, Unwrap = wahr}

Schlüsseltyp	Zulässige Schlüsselverwendung	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
			• {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr} • {= wahr} NoRestrictions
Schlüssel zur PIN-Generierung	TR31_V2_VISA_PIN_VERIFICATION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Verifizieren = wahr} • {Generieren = wahr, Verifizieren = wahr}

VerifyPinData (für **IBM3624**)

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
PIN-Verschlüsselungsschlüssel	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	Für IBM3624_NATURAL_PIN, IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET • {Entschlüsseln = wahr, Unwrap = wahr}

Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
			• {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr} • {= wahr} NoRestrictions
PIN-Bestätigungsschlüssel	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Verifizieren = wahr} • {Generieren = wahr, Verifizieren = wahr}

Daten entschlüsseln

Schlüsseltyp	Zulässige Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
DUMPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2-SCHLÜSSEL • AES_128 • AES_192 • AES_256	• { DeriveKey = wahr} • { NoRestrictions = wahr}
EMV	TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	• {= wahr} DeriveKey

Schlüsseltyp	Zulässige Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
RSA	TR31_D1_ASYMMETRISCHER_SCHLÜSSEL_FÜR_DATENVERSCHLÜSSELUNG	• RSA_2048 • RSA_3072 • RSA_4096	• {Entschlüsseln = wahr, unwrap=WAHR} • {encrypt=TRUE, WRAP=TRUE, DECRYPT = WAHR, UNWRAP=WAHR}
Symmetrische Schlüssel	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Entschlüsseln = wahr, unwrap=Wahr} • {encrypt=TRUE, WRAP=TRUE, DECRYPT = WAHR, UNWRAP=WAHR} • NoRestrictions {= wahr}

Encrypt Data

Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
DUMPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2-SCHLÜSSEL • AES_128 • AES_192	• { DeriveKey = wahr} • { NoRestrictions = wahr}

Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
		• AES_256	
EMV	TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	• {= wahr} DeriveKey
RSA	TR31_D1_ASYMMETRISCHER_SCHLÜSSEL_FÜR_DATENVERSCHLÜSSELUNG	• RSA_2048 • RSA_3072 • RSA_4096	• {Verschlüsseln = wahr, wrap=True} • {encrypt=TRUE, WRAP=TRUE, DECRYPT = WAHR, UNWRAP=WAHR}
Symmetrische Schlüssel	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Encrypt = wahr, wrap=True} • {encrypt=TRUE, WRAP=TRUE, DECRYPT = WAHR, UNWRAP=WAHR} • NoRestrictions {= wahr}

Pin-Daten übersetzen

Richtung	Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässige r Schlüssel algorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
Eingehende Datenquelle	DUMPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2-SCHLÜSSEL • AES_128 • AES_192 • AES_256	• { DeriveKey = wahr} • { NoRestrictions = wahr}
Eingehende Datenquelle	Nicht-DUPT (PEK, AWK, IWK usw.)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Decrypt = wahr, Unwrap = wahr} • {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr} • {= wahr} NoRestrictions
Ziel für ausgehende Daten	DUMPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2-SCHLÜSSEL • AES_128 • AES_192 • AES_256	• { DeriveKey = wahr} • { NoRestrictions = wahr}
Ziel für ausgehende Daten	Nicht DUPT (PEK, IWK, AWK usw.)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128	• {Encrypt = wahr, Wrap = wahr}

Richtung	Schlüsseltyp	Erlaubte Verwendung von Schlüsseln	Zulässige r Schlüssel algorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
			• AES_192 • AES_256	• {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr} • {= wahr} NoRestrictions

Generate/Verify MAC

MAC-Schlüssel werden verwendet, um kryptografische Hashes einer message/body Reihe von Daten zu erstellen. Es wird nicht empfohlen, einen Schlüssel mit eingeschränkten Schlüsselverwendungsmodi zu erstellen, da Sie den Abgleichvorgang dann nicht ausführen können. Sie können import/export einen Schlüssel jedoch nur mit einer Operation verwenden, wenn das andere System die andere Hälfte des Operationspaars ausführen soll.

Zulässige Verwendung von Schlüsseln	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüssel algorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
MAC-Schlüssel	TR31_M1_ISO_9797_1_MAC_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Generieren = wahr} • {Generieren = wahr, verifizieren = wahr} • {Verifizieren = wahr} • {Generieren = wahr}

Zulässige Verwendung von Schlüsseln	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
MAC-Schlüssel (MAC für den Einzelhandel)	TR31_M1_ISO_9797_3_MAC_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Generieren = wahr} • {Generieren = wahr, verifizieren = wahr} • {Verifizieren = wahr} • {Generieren = wahr}
MAC-Schlüssel (CMAC)	TR31_M6_ISO_9797_5_CMAC_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Generieren = wahr} • {Generieren = wahr, verifizieren = wahr} • {Verifizieren = wahr} • {Generieren = wahr}
MAC-Schlüssel (HMAC)	TR31_M7_HMAC_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Generieren = wahr} • {Generieren = wahr, verifizieren = wahr} • {Verifizieren = wahr}
MAC-Schlüssel (AS2805)	TR31_M0_ISO_16609_MAC_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY	• {Generieren = wahr} • {Generieren = wahr, verifizieren = wahr} • {Verifizieren = wahr}

GenerateMacEmvPinChange

GenerateMacEmvPinChange kombiniert MAC-Generierung und PIN-Verschlüsselung für EMV-Offline-PIN-Änderungsvorgänge. Für diesen Vorgang sind zwei verschiedene Schlüsseltypen erforderlich: ein Integritätsschlüssel für die MAC-Generierung und ein Vertraulichkeitsschlüssel für die PIN-Verschlüsselung.

Schlüsseltyp	Zulässige Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
Integritätsschlüssel für sicheres Messaging	TR31_E2_EMV_MKEY_INTEGRITY	• TDES_2-SCHLÜSSEL	• {= wahr} NoRestrictions
Schlüssel zur Vertraulichkeit von Secure Messaging	TR31_E1_EMV_MKEY_CONFIDENTIALITY	• TDES_2-SCHLÜSSEL	• { DeriveKey = wahr}
Aktueller PIN PEK (PIN-Verschlüsselungsschlüssel)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Decrypt = wahr, Unwrap = wahr} • {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr} • {= wahr} NoRestrictions
Neue PIN PEK (PIN-Verschlüsselungsschlüssel)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Decrypt = wahr, Unwrap = wahr} • {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr} • {= wahr} NoRestrictions

Schlüsseltyp	Zulässige Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
ARQC-Schlüssel **Note** Gilt nur für Ableitungssysteme von Visa und Amex.	TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS	• TDES_2-SCHLÜSSEL	• {= wahr} DeriveKey

VerifyAuthRequestCryptogram

Erlaubte Verwendung von Schlüsseln	EMV-Option	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
• OPTION A • OPTION B	TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS	• TDES_2-SCHLÜSSEL	• {= wahr} DeriveKey

Import/Export Schlüssel

Vorgangstyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
TR-31 Schlüssel einwickeln	TR31_K1_KEY_BLOCK_PROTECTION_KEY	• TDES_2-SCHLÜSSEL • TDES_3KEY • AES_128	• {Encrypt = true, Wrap = true} (nur Export)

Vorgangstyp	Erlaubte Verwendung von Schlüsseln	Zulässiger Schlüsselalgorithmus	Zulässige Kombination der wichtigsten Nutzungsmodi
	TR31_K0_KEY_ENCRYPTION_KEY	• AES_192 • AES_256	• {Decrypt = true, Unwrap = true} (nur Import) • {Encrypt = wahr, Decrypt = wahr, Wrap = wahr, Unwrap = wahr}
Import einer vertrauenswürdigen CA	TR31_S0_ASYMMETRISCHER_SCHLÜSSEL_FÜR_DIGITALE_SIGNATUR	• RSA_2048 • RSA_3072 • RSA_4096	• {Verifizieren = wahr}
Import eines Public-Key-Zertifikats für asymmetrische Verschlüsselung	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• {encrypt=WAHR, WRAP=WAHR}
Schlüssel, der für wichtige Vereinbarungsalgorithmen wie ECDH verwendet wird	TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT	• ECC_NIST_P256 • ECC_NIST_P384 • ECC_NIST_P521	• {= wahr} DeriveKey

Unbenutzte Schlüsseltypen

Die folgenden Schlüsseltypen werden derzeit nicht von AWS Payment Cryptography verwendet

- TR31_P1_PIN_GENERATION_KEY

Häufige Anwendungsfälle

AWS Die Zahlungskryptografie unterstützt viele typische kryptografische Zahlungsvorgänge. Die folgenden Themen dienen als Leitfaden für die Verwendung dieser Operationen für typische allgemeine Anwendungsfälle. Eine Liste aller Befehle finden Sie in der AWS Payment Cryptography API.

Themen

- Emittenten und Emittentenverarbeiter
- Akquisitions- und Zahlungsvermittler

Emittenten und Emittentenverarbeiter

Anwendungsfälle für Emittenten bestehen in der Regel aus wenigen Teilen. Dieser Abschnitt ist nach Funktionen gegliedert (z. B. Arbeiten mit Pins). In einem Produktionssystem sind die Schlüssel in der Regel einem bestimmten Kartenfach zugeordnet und werden bei der Einrichtung des Bins erstellt und nicht direkt, wie hier gezeigt.

Themen

- Allgemeine Funktionen
- Netzwerkspezifische Funktionen

Allgemeine Funktionen

Themen

- Generieren Sie eine zufällige PIN und das zugehörige PVV und überprüfen Sie dann den Wert
- Generieren oder verifizieren Sie einen CVV für eine bestimmte Karte
- Generieren oder verifizieren Sie einen CVV2 für eine bestimmte Karte
- Generieren oder verifizieren Sie ein iCVV für eine bestimmte Karte
- Verifizieren Sie einen EMV-ARQC und generieren Sie einen ARPC
- Generieren und verifizieren Sie einen EMV-MAC
- Generieren Sie EMV MAC für die PIN-Änderung

Generieren Sie eine zufällige PIN und das zugehörige PVV und überprüfen Sie dann den Wert

Themen

- Erstellen Sie den oder die Schlüssel
- Generieren Sie eine zufällige PIN, generieren Sie PVV und geben Sie die verschlüsselte PIN und PVV zurück
- Validieren Sie die verschlüsselte PIN mit der PVV-Methode
- Benutzerverwaltete PINs

Erstellen Sie den oder die Schlüssel

Um eine zufällige PIN und den PVV zu generieren, benötigen Sie zwei Schlüssel, einen Pin-Bestätigungsschlüssel (PVK) zum Generieren des PVV und einen Pin-Verschlüsselungsschlüssel zum Verschlüsseln des Pins. Die PIN selbst wird per Zufallsprinzip sicher innerhalb des Dienstes generiert und steht kryptografisch mit keinem der beiden Schlüssel in Beziehung.

Das PGK muss ein Schlüssel des Algorithmus TDES_2KEY sein, der auf dem PVV-Algorithmus selbst basiert. Ein PEK kann TDES_2KEY, TDES_3KEY oder AES_128 sein. In diesem Fall wäre AES_128 eine gute Wahl, da das PEK für den internen Gebrauch in Ihrem System vorgesehen ist. Wenn ein PEK für den Austausch mit anderen Systemen (z. B. Kartennetzwerken, Acquirern, Geldautomaten) verwendet wird oder im Rahmen einer Migration verschoben wird, ist TDES_2KEY aus Kompatibilitätsgründen möglicherweise die geeignetere Wahl.

Erstellen Sie das PEK

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
```

```
            "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"
            }
        }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/ivi5ksfsuplneuyt Das brauchst du im nächsten Schritt.

Erstellen Sie das PVK

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
```

```
                "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"
                }
            }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/ov6icy4ryas4zcza Das brauchst du im nächsten Schritt.

Generieren Sie eine zufällige PIN, generieren Sie PVV und geben Sie die verschlüsselte PIN und PVV zurück

Example

In diesem Beispiel generieren wir einen neuen (zufälligen) 4-stelligen Pin, wobei die Ausgänge ein verschlüsselter PIN block (PinData.PinBlock) und ein PVV (PinData.VerificationValue) sind. Die wichtigsten Eingaben sind PAN das Pin Verification Key (auch als PIN-Generierungsschlüssel bekannt), das Pin Encryption Key und das ??? PIN-Blockformat.

Für diesen Befehl muss der Schlüssel vom Typ A sein`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"
            }
        }
```

Validieren Sie die verschlüsselte PIN mit der PVV-Methode

Example

In diesem Beispiel validieren wir eine PIN für eine bestimmte PAN. Die PIN wird in der Regel vom Karteninhaber oder Benutzer während der Transaktion zur Validierung bereitgestellt und mit dem hinterlegten Wert verglichen (die Eingabe des Karteninhabers wird als verschlüsselter Wert vom Terminal oder einem anderen Upstream-Anbieter bereitgestellt). Um diese Eingabe zu validieren, werden zur Laufzeit auch die folgenden Werte bereitgestellt: Der verschlüsselte Pin, der Schlüssel, der zum Verschlüsseln des Eingabe-Pins verwendet wird (oft als IWK bezeichnet), PAN und der Wert, mit dem verifiziert werden soll (entweder ein PVV oder`PIN offset`).

Wenn AWS Payment Cryptography die PIN validieren kann, wird eine http/200 zurückgegeben. Wenn die PIN nicht validiert wird, wird eine http/400 zurückgegeben.

```
$ 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",
}
```

Benutzerverwaltete PINs

Themen

- -Übersicht
- Erstellen Sie die Schlüssel
- Stellen Sie eine vom Benutzer auswählbare PIN ein
- Setzen Sie eine PIN zurück (grüne PIN)
- Zeigen Sie eine PIN an

-Übersicht

AWS Payment Cryptography unterstützt benutzerverwaltete (vom Kunden verwaltete) PIN-Flows, bei denen eine PIN über Ihr Kunden-Backend sicher zwischen einem Gerät des Karteninhabers und dem Service ausgetauscht wird. Das Gerät und der Dienst verwenden die Schlüsselvereinbarung Elliptic Curve Diffie-Hellman (ECDH), um auf jeder Seite einen gemeinsamen symmetrischen Schlüssel abzuleiten. Dieser Schlüssel schützt einen 4-PIN-Block im ISO-Format. ??? Da der Schlüssel abgeleitet und nicht ausgetauscht wird, werden die Clear-PIN und der Clear-PIN-Block niemals über das Gerät hinaus übertragen.

An diesen Datenströmen sind drei Parteien beteiligt. Das Gerät des Karteninhabers generiert sein EC-Schlüsselpaar, leitet den AES-128 ECDH-Schlüssel ab, verschlüsselt oder entschlüsselt den 4-PIN-Block im ISO-Format und speichert die eindeutige PIN; es hat keine AWS-Anmeldeinformationen. Das Kunden-Backend (Ihr Aussteller oder Prozessorserver) speichert die AWS-Anmeldeinformationen, ruft AWS Payment Cryptography auf, signiert das Gerätezertifikat und

speichert das PVV. ??? AWS Payment Cryptography führt die kryptografischen Operationen durch. Das Gerät kommuniziert nur mit dem Kunden-Backend, und das Kunden-Backend tätigt alle Aufrufe an Payment Cryptography. AWS

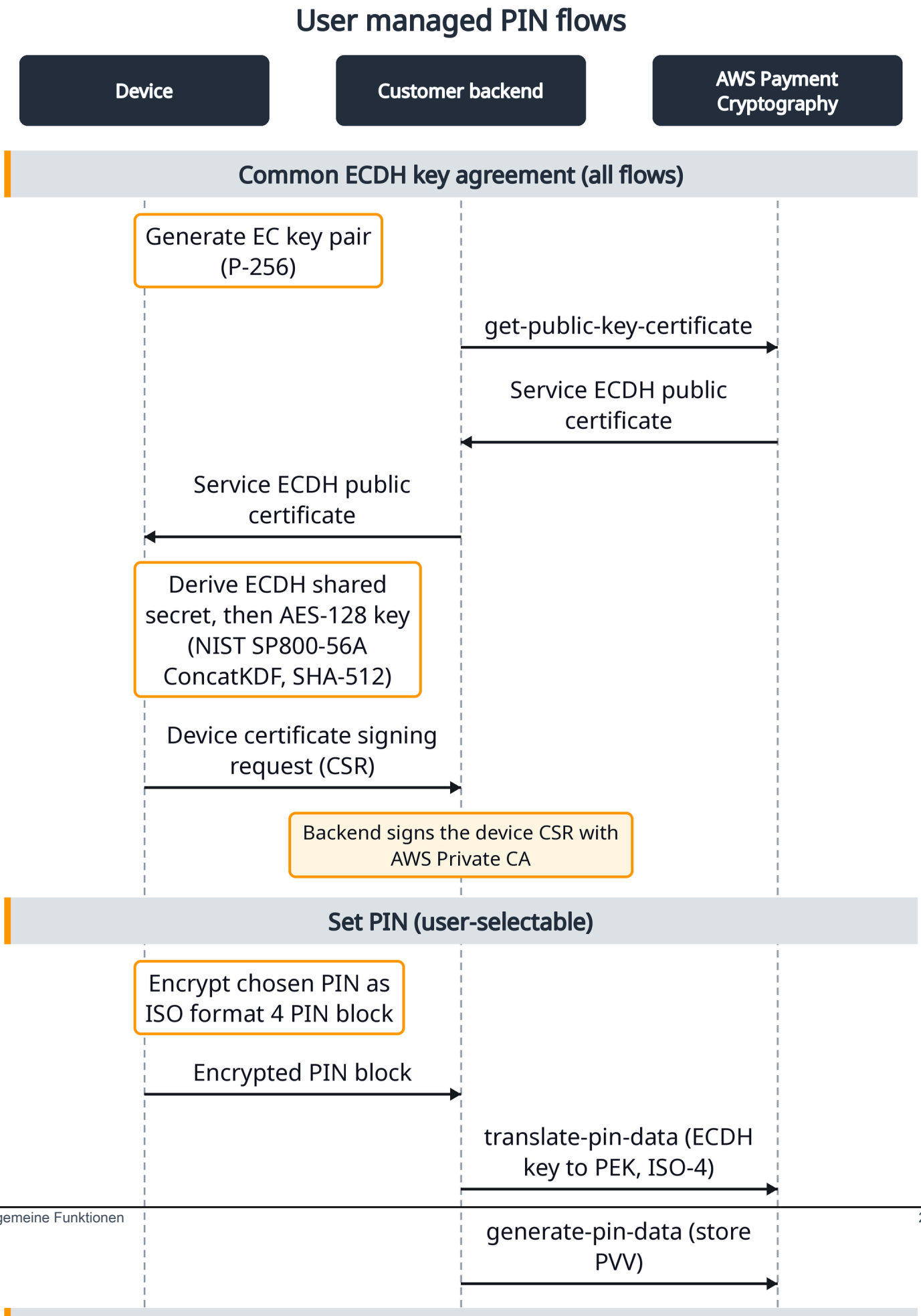
Dieses Thema behandelt drei Abläufe:

- User-selectable PIN (PIN festlegen) — Der Karteninhaber wählt seine eigene PIN auf dem Gerät.
- Grüne PIN (PIN zurücksetzen) — der Dienst generiert eine zufällige PIN, die dem Karteninhaber zugestellt wird, zum Beispiel bei der ersten Ausgabe oder beim Zurücksetzen einer vergessenen PIN.
- PIN anzeigen — ein vorhandener PIN-Block wird für enge Anwendungsfälle an ein autorisiertes Gerät zurückgegeben.

Alle drei Flows verwenden dieselbe Tastenkonfiguration und dieselbe ECDH-Mechanik zur Gerätevereinbarung, wie unter Schlüssel erstellen beschrieben. Einzelheiten zur Schlüsselvereinbarung selbst finden Sie unter Schlüssel mithilfe asymmetrischer Techniken importieren (ECDH).

> **Note**
>
> Diese Datenflüsse verarbeiten klare PINs und PIN-Blöcke nur auf dem Gerät, niemals während der Übertragung an AWS Payment Cryptography. Prüfen Sie jeden Flow sorgfältig mit Ihrem Sicherheitsteam, bevor Sie ihn bereitstellen.

Das folgende Diagramm zeigt die vom Benutzer verwalteten PIN-Flows.


User managed PIN flows
Device
Customer backend
AWS Payment Cryptography
Common ECDH key agreement (all flows)
Generate EC key pair (P-256)
get-public-key-certificate
Service ECDH public certificate
Service ECDH public certificate
Derive ECDH shared secret, then AES-128 key (NIST SP800-56A ConcatKDF, SHA-512)
Device certificate signing request (CSR)
Backend signs the device CSR with AWS Private CA
Set PIN (user-selectable)
Encrypt chosen PIN as ISO format 4 PIN block
Encrypted PIN block
translate-pin-data (ECDH key to PEK, ISO-4)
generate-pin-data (store PVV)

Ein vollständiges, lauffähiges Beispiel für diese Flows mit dem AWS-SDK für Python finden Sie im Beispiel für ECDH-PIN-Sets und Reveal-Flows im AWS Beispiel-Repository unter. GitHub

> **Note**
>
> Der typische Ablauf beinhaltet ein Gerät des Karteninhabers, das die ECDH-Schlüsselvereinbarung durchführt und den 4-PIN-Block im ISO-Format verschlüsselt oder entschlüsselt. Wenn Sie die PINs zwischen dem Gerät und Ihrem Kunden-Backend bereits auf andere Weise schützen, können Sie eine modifizierte Version dieser Datenflüsse direkt zwischen dem Kunden-Backend und Payment Cryptography verwenden. AWS In dieser Variante führt das Kunden-Backend anstelle des Geräts die ECDH-Schlüsselvereinbarung und die PIN-Sperrvorgänge durch.

Erstellen Sie die Schlüssel

Alle drei Datenflüsse verwenden denselben Schlüsselsatz: einen ECDH-Vereinbarungsschlüssel für die Schlüsselvereinbarung mit dem Gerät, einen Generation/Verification PIN-Schlüssel (PGK) zur Generierung des PVV und einen Pin-Verschlüsselungsschlüssel (PEK), in den der PIN-Block für die Backend-Speicherung übersetzt wird. Sie importieren auch den öffentlichen Schlüssel der Gerätezertifizierungsstelle (CA), sodass AWS Payment Cryptography den bei der Übersetzung vorgelegten Gerätezertifikaten vertraut.

Erstellen Sie den ECDH-Vereinbarungsschlüssel. Dieser Schlüssel wird verwendet, um den gemeinsamen symmetrischen Schlüssel mit dem Gerät abzuleiten.

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

Die Antwort gibt die Anforderungsparameter zurück und enthält den ARN des ECDH-Schlüssels.

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxx",
        "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": "1A2B3C",
        "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"
    }
}
```

Erstellen Sie den Generation/Verification Pin-Schlüssel (PGK), der zur Generierung des PVV verwendet wurde.

```
$ aws payment-cryptography create-key \
    --exportable \
    --key-attributes KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_V2_VISA_PIN_VERIFICATION_KEY,
\
        KeyClass=SYMMETRIC_KEY,\
        KeyModesOfUse='{Generate=true,Verify=true}'
```

```
{
    "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"
    }
}
```

Erstellen Sie den PIN-Verschlüsselungsschlüssel (PEK), in den der PIN-Block des Geräts für die Backend-Speicherung übersetzt wird.

```
$ 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}'
```

```
{
    "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": true,
                "Decrypt": true,
```

```
                "Wrap": true,
                "Unwrap": true,
                "Generate": false,
                "Sign": false,
                "Verify": false,
                "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"
    }
}
```

Importieren Sie den öffentlichen Schlüssel der Gerätezertifizierungsstelle, damit AWS Payment Cryptography die bei späteren Übersetzungsaufrufen vorgelegten Gerätezertifikate überprüfen kann. Ersetzen Sie es <base64-CA-cert> durch das Base64-kodierte CA-Zertifikat.

```
$ aws payment-cryptography import-key \
    --key-material='{"RootCertificatePublicKey":{"KeyAttributes":
{"KeyAlgorithm":"ECC_NIST_P256","KeyClass":"PUBLIC_KEY","KeyModesOfUse":
{"Verify":true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"},"PublicKeyCertificat
CA-cert>"}}'
```

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
yyyyyyyyyyyyyyyy",
        "KeyAttributes": {
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE",
            "KeyClass": "PUBLIC_KEY",
            "KeyAlgorithm": "ECC_NIST_P256",
            "KeyModesOfUse": {
                "Verify": true
            }
        },
```

```
        "KeyCheckValue": "4D5E6F",
        "KeyCheckValueAlgorithm": "CMAC",
        "Enabled": true,
        "Exportable": false,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "EXTERNAL",
        "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
        "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
    }
}
```

Notieren Sie sich jede Rückgabe. `KeyArn` In den folgenden Beispielen werden die folgenden literalen Platzhalter verwendet:

- Schlüssel zur ECDH-Vereinbarung: `arn:aws:payment-cryptography:us-east-1:111122223333:key/xxxxxxxxxxxxxxxx`
- Importierter öffentlicher CA-Schlüssel: `arn:aws:payment-cryptography:us-east-1:111122223333:key/yyyyyyyyyyyyyyyy`
- PGK: `arn:aws:payment-cryptography:us-east-2:111122223333:key/ov6icy4ryas4zcza`
- PEK: `arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt`

Auf dem Gerät des Karteninhabers (in der Regel ein Mobiltelefon oder ein ähnliches Gerät) werden die folgenden Schritte im Anwendungscode des Geräts für jeden der Flows „PIN setzen“, „PIN zurücksetzen“ und „PIN anzeigen“ ausgeführt: Generieren Sie ein EC-Schlüsselpaar auf einer Kurve P-256; beziehen Sie das öffentliche ECDH-Zertifikat vom Kunden-Backend; leiten Sie das gemeinsame ECDH-Geheimnis ab und dann einen AES-128 16-Byte-Schlüssel mit dem Wert (NIST SP800-56A ConcatKDF (SHA-512) mit dem `SharedInformation` Wert; und verschlüsseln oder entschlüsseln Sie den 4-PIN-Block im ISO-Format damit abgeleiteter Schlüssel. Das Gerät hat keine AWS-Anmeldeinformationen und ruft AWS Payment Cryptography nicht direkt auf. Es tauscht diese Daten mit dem Kunden-Backend aus, das die Serviceanrufe tätigt.

Da es sich bei dem Gerät in der Regel um ein Telefon handelt, werden das EC-Schlüsselpaar und das Zertifikat bei der ersten Verwendung bereitgestellt und danach regelmäßig aktualisiert. Das Gerät generiert das EC-Schlüsselpaar und eine Certificate Signing Request (CSR), und das Kunden-Backend signiert das CSR mithilfe AWS Private CA einer oder einer ähnlichen CA. Das Kunden-Backend verwendet das resultierende signierte Gerätezertifikat in seinen Translate-Pin-Data-

Aufrufen, und AWS Payment Cryptography verifiziert es anhand des öffentlichen CA-Schlüssels, den Sie importiert haben.

Das Kunden-Backend erhält das öffentliche ECDH-Zertifikat des Dienstes und stellt es dem Gerät zur Verfügung. Das Zertifikat wird als Base64-kodiertes PEM in dem `KeyCertificate` Feld zurückgegeben, das in einer PEM-Datei base64-dekodiert wird. Das Gerät verwendet dieses Zertifikat als öffentlichen Dienstschlüssel, wenn es das gemeinsame Geheimnis ableitet.

```
$ aws payment-cryptography get-public-key-certificate \
    --key-identifier arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxxx
```

> **Note**
>
> `SharedInformation`besteht aus 32 zufälligen Bytes, die auf dem Gerät generiert und als Hexadezimalzeichenfolge über das Kunden-Backend an AWS Payment Cryptography weitergeleitet werden. Sie muss mit dem Wert übereinstimmen, den das Gerät in der KDF verwendet hat, sonst stimmen die abgeleiteten Schlüssel nicht überein.

Stellen Sie eine vom Benutzer auswählbare PIN ein

Im Ablauf „PIN festlegen“ wählt der Karteninhaber seine eigene PIN auf dem Gerät aus (z. B.). `1234` Das Gerät leitet den gemeinsamen ECDH-Schlüssel ab, verschlüsselt die gewählte PIN in einen 4-PIN-Block im ISO-Format unter diesem Schlüssel und sendet den verschlüsselten PIN-Block, seine Zertifikatsignieranforderung (CSR) und die geteilten Informationen an das Kunden-Backend. Das Kunden-Backend signiert den CSR und führt dann zwei Aufrufe auf Datenebene durch. In diesem Beispiel wird das Beispiel verwendet. PAN `171234567890123` Die angezeigten Werte für PIN-Block, Zertifikat und gemeinsame Informationen sind fiktive Platzhalter.

Übersetzen Sie zunächst den ECDH-protected PIN-Block auf das Server-PEK. Der eingehende Schlüssel ist der ECDH-Vereinbarungsschlüssel, der mit den Diffie-Hellman Parametern und dem signierten Gerätezertifikat versehen ist. Der ausgehende Schlüssel ist der PEK.

```
$ aws payment-cryptography-data translate-pin-data \
    --encrypted-pin-block A1B2C3D4E5F60718 \
    --incoming-key-identifier arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxxx \
    --incoming-wrapped-key
 'WrappedKeyMaterial={DiffieHellmanSymmetricKey={CertificateAuthorityPublicKeyIdentifier=arn:aw
```

```
cryptography:us-east-1:111122223333:key/
yyyyyyyyyyyyyyyy,KeyAlgorithm=AES_128,KeyDerivationFunction=NIST_SP800,KeyDerivationHashAlgorit
signed-device-
cert>,SharedInformation=0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef}}'
 \
    --incoming-translation-attributes
 'IsoFormat4={PrimaryAccountNumber=171234567890123}' \
    --outgoing-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt \
    --outgoing-translation-attributes
 'IsoFormat4={PrimaryAccountNumber=171234567890123}'
```

Die Antwort enthält den PIN-Block, der unter dem PEK im ISO-Format 4 erneut verschlüsselt wurde.

```
{
    "PinBlock": "F7E6D5C4B3A29180",
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
    "KeyCheckValue": "7CC9E2"
}
```

Generieren Sie als Nächstes das PVV für den PEK-encrypted PIN-Block. Das PGK generiert das PVV, und das PEK identifiziert den Schlüssel, unter dem der bereitgestellte PIN-Block verschlüsselt ist.

```
$ aws payment-cryptography-data generate-pin-data \
    --generation-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ov6icy4ryas4zcza \
    --encryption-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt \
    --pin-block-format ISO_FORMAT_4 \
    --primary-account-number 171234567890123 \
    --generation-attributes
 'VisaPinVerificationValue={PinVerificationKeyIndex=1,EncryptedPinBlock=F7E6D5C4B3A29180}'
```

Die Antwort enthält den PVV-Eingang `PinData.VerificationValue` und den verschlüsselten PIN-Block. Speichern Sie das PVV im Kunden-Backend, damit Sie die PIN später überprüfen können. Informationen zur Überprüfung finden Sie unter Generieren Sie eine zufällige PIN und die zugehörige PVV und überprüfen Sie dann den Wert.

```
{
```

```
    "GenerationKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
    "GenerationKeyCheckValue": "51A200",
    "EncryptionKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
    "EncryptionKeyCheckValue": "7CC9E2",
    "EncryptedPinBlock": "F7E6D5C4B3A29180",
    "PinData": {
        "VerificationValue": "5507"
    }
}
```

Weitere Informationen finden Sie unter TranslatePinData und GeneratePinData im API-Referenzhandbuch.

Setzen Sie eine PIN zurück (grüne PIN)

Beim Zurücksetzen der PIN (grüne PIN) generiert AWS Payment Cryptography eine zufällige PIN, die das Kunden-Backend an das Gerät zurückgibt, um sie dem Karteninhaber anzuzeigen, beispielsweise bei der ersten Ausgabe oder beim Zurücksetzen einer vergessenen PIN. Das Gerät führt dasselbe ECDH-Setup aus, das unter Schlüssel erstellen beschrieben ist, und das Kunden-Backend führt die Aufrufe auf der Datenebene in seinem Namen durch. In diesem Beispiel wird das Beispiel verwendet PAN`171234567890123`. Der PIN-Block und die gemeinsamen Informationswerte sind fiktive Platzhalter.

Generieren Sie zunächst eine zufällige PIN und deren PVV und verschlüsseln Sie dabei den PIN-Block unter dem PEK.

```
$ aws payment-cryptography-data generate-pin-data \
    --generation-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ov6icy4ryas4zcza \
    --encryption-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt \
    --pin-block-format ISO_FORMAT_4 \
    --primary-account-number 171234567890123 \
    --generation-attributes 'VisaPin={PinVerificationKeyIndex=1}'
```

Die Antwort enthält den PVV-Eingang `PinData.VerificationValue` und den PEK-encrypted PIN-Block (ISO-Format 4). Speichern Sie das PVV zur späteren Überprüfung im Kunden-Backend.

```
{
```

```
    "GenerationKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
    "GenerationKeyCheckValue": "51A200",
    "EncryptionKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
    "EncryptionKeyCheckValue": "7CC9E2",
    "EncryptedPinBlock": "F7E6D5C4B3A29180",
    "PinData": {
        "VerificationValue": "5507"
    }
}
```

Als Nächstes übersetzen Sie den PEK-encrypted PIN-Block (ISO-Format 4) in den ECDH-derived Geräteschlüssel (ISO-Format 4), sodass ihn nur das anfordernde Gerät lesen kann.

```
$ aws payment-cryptography-data translate-pin-data \
    --encrypted-pin-block F7E6D5C4B3A29180 \
    --incoming-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt \
    --incoming-translation-attributes
 'IsoFormat4={PrimaryAccountNumber=171234567890123}' \
    --outgoing-key-identifier arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxx \
    --outgoing-wrapped-key
 'WrappedKeyMaterial={DiffieHellmanSymmetricKey={CertificateAuthorityPublicKeyIdentifier=arn:aw
cryptography:us-east-1:111122223333:key/
yyyyyyyyyyyyyyyy,KeyAlgorithm=AES_128,KeyDerivationFunction=NIST_SP800,KeyDerivationHashAlgorit
signed-device-
cert>,SharedInformation=0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef}}'
 \
    --outgoing-translation-attributes
 'IsoFormat4={PrimaryAccountNumber=171234567890123}'
```

Die Antwort enthält den PIN-Block, der unter dem ECDH-derived Geräteschlüssel im ISO-Format 4 erneut verschlüsselt wurde.

```
{
    "PinBlock": "A1B2C3D4E5F60718",
    "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxx",
    "KeyCheckValue": "1A2B3C"
}
```

Das Kunden-Backend gibt den ECDH-encrypted PIN-Block an das Gerät zurück. Das Gerät entschlüsselt den zurückgegebenen PIN-Block mit seinem ECDH-derived Schlüssel (im Geräteanwendungscode) und zeigt die PIN dem Karteninhaber an. Da der Schlüssel nur von diesem Gerät abgeleitet wird, kann keine andere Partei den PIN-Block entschlüsseln.

Weitere Informationen finden Sie unter GeneratePinData und TranslatePinData im API-Referenzhandbuch.

Zeigen Sie eine PIN an

Der Vorgang „PIN anzeigen“ gibt einen vorhandenen PIN-Block an das Gerät des Karteninhabers zurück, sodass das Gerät die PIN anzeigen oder auf andere Weise preisgeben kann. Das Gerät führt dasselbe ECDH-Setup aus, das unter Schlüssel erstellen beschrieben wurde. Das Kunden-Backend übersetzt den gespeicherten PEK-encrypted PIN-Block zurück auf den ECDH-derived Schlüssel und gibt ihn an das Gerät zurück. In diesem Beispiel wird das Beispiel verwendet PAN`171234567890123`. Der PIN-Block und die gemeinsamen Informationswerte sind fiktive Platzhalter.

> **Note**
>
> Erwägen Sie, den Karteninhaber zu authentifizieren und ihn eine neue PIN festlegen zu lassen, anstatt die bestehende PIN preiszugeben. Eine bemerkenswerte Ausnahme bilden Länder, die eine Offline-PIN verwenden, z. B. in Europa, wo die PIN auch auf der Chipkarte gespeichert ist. In diesen Fällen kann es einfacher sein, die vorhandene PIN preiszugeben, als die Karte neu zu programmieren.

Übersetzen Sie den gespeicherten PEK-encrypted PIN-Block (ISO-Format 4) in den ECDH-derived Geräteschlüssel (ISO-Format 4).

```
$ aws payment-cryptography-data translate-pin-data \
    --encrypted-pin-block F7E6D5C4B3A29180 \
    --incoming-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt \
    --incoming-translation-attributes
 'IsoFormat4={PrimaryAccountNumber=171234567890123}' \
    --outgoing-key-identifier arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxxx \
    --outgoing-wrapped-key
 'WrappedKeyMaterial={DiffieHellmanSymmetricKey={CertificateAuthorityPublicKeyIdentifier=arn:aw
```

```
cryptography:us-east-1:111122223333:key/
yyyyyyyyyyyyyyyy,KeyAlgorithm=AES_128,KeyDerivationFunction=NIST_SP800,KeyDerivationHashAlgorit
signed-device-
cert>,SharedInformation=0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef}}'
 \
    --outgoing-translation-attributes
 'IsoFormat4={PrimaryAccountNumber=171234567890123}'
```

Die Antwort enthält den PIN-Block, der unter dem ECDH-derived Geräteschlüssel im ISO-Format 4 erneut verschlüsselt wurde.

```
{
    "PinBlock": "A1B2C3D4E5F60718",
    "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxx",
    "KeyCheckValue": "1A2B3C"
}
```

Das Kunden-Backend gibt den ECDH-encrypted PIN-Block an das Gerät zurück. Das Gerät entschlüsselt den zurückgegebenen PIN-Block mit seinem ECDH-derived Schlüssel (im Geräteanwendungscode), um die PIN anzuzeigen. Da der Schlüssel nur von diesem Gerät abgeleitet wird, kann keine andere Partei den PIN-Block entschlüsseln.

Weitere Informationen finden Sie TranslatePinData im API-Referenzhandbuch.

Generieren oder verifizieren Sie einen CVV für eine bestimmte Karte

CVV oder CVV1 ist ein Wert, der traditionell in den Magnetstreifen einer Karte eingebettet ist. Er ist nicht dasselbe wie CVV2 (für den Karteninhaber sichtbar und für Online-Käufe geeignet).

Der erste Schritt besteht darin, einen Schlüssel zu erstellen. Für dieses Tutorial erstellen Sie einen CVK-3DES-Schlüssel mit doppelter Länge (2KEY TDES).

> **Note**
>
> CVV, CVV2 und iCVV verwenden alle ähnliche, wenn nicht sogar identische Algorithmen, variieren jedoch die Eingabedaten. Alle verwenden den gleichen Schlüsseltyp TR31_C0_CARD_VERIFICATION_KEY, es wird jedoch empfohlen, für jeden Zweck separate Schlüssel zu verwenden. Diese können mithilfe von Alias-Tags wie im folgenden Beispiel unterschieden werden. and/or

Erstellen Sie den Schlüssel

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
            "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"
            }
        }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/r52o3wbqxyf6qlqr Das brauchst du im nächsten Schritt.

Generieren Sie einen CVV

Example

In diesem Beispiel generieren wir einen CVV für eine bestimmte PAN mit Eingaben vonPAN, einem Servicecode (wie in ISO/IEC 7813 definiert) von 121 und dem Ablaufdatum der Karte.

Alle verfügbaren Parameter finden Sie CardVerificationValue1 im API-Referenzhandbuch.

```
$ 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"
                }
```

Validieren Sie CVV

Example

In diesem Beispiel verifizieren wir eine CVV für eine bestimmte PAN anhand der Eingabe einer CVK, eines Servicecodes von 121PAN, des Ablaufdatums der Karte und der CVV, die während der zu validierenden Transaktion angegeben wurde.

Alle verfügbaren Parameter finden Sie CardVerificationValue1 im API-Referenzhandbuch.

> **Note**
>
> CVV ist kein vom Benutzer eingegebener Wert (wie CVV2), sondern ist normalerweise in einen Magnetstreifen eingebettet. Es sollte überlegt werden, ob der Wert immer gültig sein sollte, wenn er angegeben wird.

```
$ 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"
}
```

Generieren oder verifizieren Sie einen CVV2 für eine bestimmte Karte

CVV2 ist ein Wert, der traditionell auf der Rückseite einer Karte angegeben ist und für Online-Käufe verwendet wird. Bei virtuellen Karten kann er auch in einer App oder einem Bildschirm angezeigt werden. Kryptografisch gesehen ist es dasselbe wie CVV1, jedoch mit einem anderen Servicecode-Wert.

Erstellen Sie den Schlüssel

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
               "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"
            }
        }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/7f7g4spf3xcklhzu Das brauchst du im nächsten Schritt.

Generieren Sie einen CVV2

Example

In diesem Beispiel generieren wir eine CVV2 für eine bestimmte PAN mit Eingaben von PAN und Ablaufdatum der Karte.

Alle verfügbaren Parameter finden Sie CardVerificationValue2 im API-Referenzhandbuch.

```
$ 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"
                }
```

Validieren Sie einen CVV2

Example

In diesem Beispiel verifizieren wir eine CVV2 für eine bestimmte PAN mit der Eingabe einer CVK, des Ablaufdatums der Karte PAN und der CVV, die während der zu validierenden Transaktion angegeben wurde.

Alle verfügbaren Parameter finden Sie CardVerificationValue2 im API-Referenzhandbuch.

> **Note**
>
> CVV2 und die anderen Eingaben sind vom Benutzer eingegebene Werte. Daher ist es nicht unbedingt ein Anzeichen für ein Problem, dass dies regelmäßig nicht validiert wird.

```
$ 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"
   }
```

Generieren oder verifizieren Sie ein iCVV für eine bestimmte Karte

iCVV verwendet den gleichen Algorithmus wie iCVV ist CVV/CVV2 jedoch in eine Chipkarte eingebettet. Sein Servicecode lautet 999.

Erstellen Sie den Schlüssel

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
              "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"
              }
          }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/c7dsi763r6s7lfp3 Das brauchst du im nächsten Schritt.

Generieren Sie ein iCVV

Example

In diesem Beispiel generieren wir einen iCVV für eine bestimmte PAN mit Eingaben vonPAN, einem Servicecode (wie in ISO/IEC 7813 definiert) von 999 und dem Ablaufdatum der Karte.

Alle verfügbaren Parameter finden Sie CardVerificationValue1 im API-Referenzhandbuch.

```
$ 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"
                    }
```

Validieren Sie iCVV

Example

Für die Validierung sind die Eingaben CVK, ein Servicecode von 999PAN, das Ablaufdatum der Karte und der ICVV, der während der zu validierenden Transaktion angegeben wurde.

Alle verfügbaren Parameter finden Sie CardVerificationValue1 im API-Referenzhandbuch.

> **Note**
>
> iCVV ist kein vom Benutzer eingegebener Wert (wie CVV2), sondern ist normalerweise auf einer Karte eingebettet. EMV/chip Es sollte überlegt werden, ob der Wert immer gültig sein sollte, wenn er bereitgestellt wird.

```
$ 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"
   }
```

Verifizieren Sie einen EMV-ARQC und generieren Sie einen ARPC

ARQC (Authorization Request Cryptogram) ist ein Kryptogramm, das von einer EMV-Karte (Chip) generiert und zur Validierung der Transaktionsdetails sowie der Verwendung einer autorisierten Karte verwendet wird. Es enthält Daten von der Karte, dem Terminal und der Transaktion selbst.

Bei der Validierung im Backend werden dieselben Eingaben für AWS Payment Cryptography bereitgestellt. Das Kryptogramm wird intern neu erstellt und mit dem bei der Transaktion bereitgestellten Wert verglichen. In diesem Sinne ähnelt es einem MAC. EMV 4.4 Book 2 definiert drei Aspekte dieser Funktion: Methoden zur Schlüsselableitung (bekannt als Common Session Key — CSK) zur Generierung einmaliger Transaktionsschlüssel, eine Mindestnutzlast und Methoden zur Generierung einer Antwort (ARPC).

Einzelne Kartensysteme können zusätzliche Transaktionsfelder angeben, die integriert werden müssen, oder die Reihenfolge, in der diese Felder angezeigt werden. Andere (im Allgemeinen veraltete) schemaspezifische Ableitungsschemata existieren ebenfalls und werden an anderer Stelle in dieser Dokumentation behandelt.

Weitere Informationen finden Sie VerifyCardValidationData im API-Handbuch.

Erstellen Sie den Schlüssel

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
                "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"
                }
            }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/pw3s6nl62t5ushfk Das brauchst du im nächsten Schritt.

Generieren Sie einen ARQC

Der ARQC wird ausschließlich von einer EMV-Karte generiert. Daher bietet AWS Payment Cryptography keine Möglichkeit, eine solche Nutzlast zu generieren. Zu Testzwecken sind eine Reihe von Bibliotheken online verfügbar, die eine entsprechende Nutzlast sowie bekannte Werte generieren können, die in der Regel von den verschiedenen Schemata bereitgestellt werden.

Validieren Sie einen ARQC

Example

Wenn AWS Payment Cryptography den ARQC validieren kann, wird ein zurückgegeben. http/200 Ein ARPC (Antwort) kann optional bereitgestellt werden und wird nach der Validierung des ARQC in die Antwort aufgenommen.

```
$ 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
 00000000170000000000000008400080008000084016051700000000093800000B1F220103000000000000000000000
 --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"
}
```

Generieren und verifizieren Sie einen EMV-MAC

EMV MAC ist MAC, der die Eingabe eines von EMV abgeleiteten Schlüssels verwendet und dann einen ISO9797-3 (Retail-) MAC für die resultierenden Daten ausführt. EMV MAC wird in der Regel verwendet, um Befehle an eine EMV-Karte zu senden, z. B. um Skripts zum Entsperren zu sperren.

> **Note**
>
> AWS Payment Cryptography validiert den Inhalt des Skripts nicht. Einzelheiten zu bestimmten Befehlen, die Sie aufnehmen müssen, finden Sie in Ihrem Schema- oder Kartenhandbuch.

Weitere Informationen finden Sie MacAlgorithmEmv im API-Handbuch.

Themen

- Erstellen Sie den Schlüssel
- Generieren Sie einen EMV-MAC

Erstellen Sie den Schlüssel

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
        "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"
```

```
        }
    }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/pw3s6nl62t5ushfk Das brauchst du im nächsten Schritt.

Generieren Sie einen EMV-MAC

In der Regel generiert ein Backend-Prozess ein EMV-Skript (z. B. card unblock), signiert es mit diesem Befehl (der einen einmaligen Schlüssel für eine bestimmte Karte ableitet) und gibt dann den MAC zurück. Dann werden die Befehle + MAC an die Karte gesendet, um sie anzuwenden. Das Senden des Befehls an die Karte fällt nicht in den Geltungsbereich der AWS Zahlungskryptografie.

> **Note**
>
> Dieser Befehl ist für Befehle gedacht, bei denen keine verschlüsselten Daten (wie PIN) gesendet werden. EMV Encrypt kann mit diesem Befehl kombiniert werden, um verschlüsselte Daten an das Aussteller-Skript anzuhängen, bevor dieser Befehl aufgerufen wird

Daten der Nachricht

Zu den Nachrichtendaten gehören der APDU-Header und der Befehl. Obwohl dies je nach Implementierung variieren kann, handelt es sich bei diesem Beispiel um den APDU-Header für Unblock (84 24 00 00 08), gefolgt von ATC (0007) und dann ARQC der vorherigen Transaktion (999E57FD0F47CACE). Der Dienst überprüft den Inhalt dieses Felds nicht.

Modus zur Ableitung des Sitzungsschlüssels

Dieses Feld definiert, wie der Sitzungsschlüssel generiert wird. EMV_COMMON_SESSION_KEY wird im Allgemeinen für die neuen Implementierungen verwendet, während EMV2000 | AMEX | MASTERCARD_SESSION_KEY | VISA ebenfalls verwendet werden kann.

MajorKeyDerivationMode

EMV definiert Modus A, B oder C. Modus A ist am gebräuchlichsten und Payment Cryptography unterstützt derzeit Modus A oder Modus B. AWS

PAN

Die Kontonummer, die normalerweise in Chipfeld 5A oder ISO8583, Feld 2, verfügbar ist, kann aber auch aus dem Kartensystem abgerufen werden.

PSN

Die Sequenznummer der Karte. Wenn nicht verwendet, geben Sie 00 ein.

SessionKeyDerivationValue

Dies sind die Ableitungsdaten pro Sitzung. Je nach Ableitungsschema kann es entweder der letzte ARQC (ApplicationCryptogram) aus Feld 9F26 oder der letzte ATC aus 9F36 sein.

Padding

Das Padding wird automatisch angewendet und verwendet die 9797-1-Füllmethode 2. ISO/IEC

Example

```
$ aws payment-cryptography-data generate-mac --message-data
 84240000080007999E57FD0F47CACE --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"
}
```

Generieren Sie EMV MAC für die PIN-Änderung

Bei der EMV-PIN-Änderung werden zwei Operationen kombiniert: das Generieren eines MAC für ein Ausstellerskript und das Verschlüsseln einer neuen PIN für die Offline-PIN-Änderung auf einer EMV-Chipkarte. Dieser Befehl wird nur in bestimmten Ländern benötigt, in denen die PIN auf der Chipkarte gespeichert ist (dies ist in europäischen Ländern üblich). Dies wird häufig verwendet, wenn ein Karteninhaber seine PIN ändern muss und die neue PIN zusammen mit einem MAC sicher auf die Karte übertragen werden muss, um die Echtheit des Befehls zu überprüfen.

> **Note**
>
> Wenn Sie nur Befehle an die Karte senden, aber die PIN nicht ändern müssen, sollten Sie stattdessen die MAC-Befehle ARPC CSU oder Generate EMV verwenden.

Weitere Informationen finden Sie GenerateMacEmvPinChange im API-Handbuch.

Generieren Sie EMV-MAC und verschlüsselte PIN für die PIN-Änderung

Für diesen Vorgang sind zwei Schlüssel erforderlich: ein EMV-Integritätsschlüssel (KeyUsage: TR31_E2_EMV_MKEY_INTEGRITY) für die MAC-Generierung und ein EMV-Vertraulichkeitsschlüssel (: TR31_E4_EMV_MKEY_CONFIDENTIAL) für die PIN-Verschlüsselung. KeyUsage Der typische Ablauf besteht darin, dass ein Backend-Prozess ein EMV-PIN-Änderungsskript generiert, das sowohl den MAC für das Aussteller-Skript als auch die verschlüsselte neue PIN enthält. Der Befehl und die verschlüsselte PIN werden dann an die Karte gesendet, um die Offline-PIN zu aktualisieren. Das Senden des Befehls an die Karte fällt nicht in den Anwendungsbereich der AWS Zahlungskryptografie.

Daten der Nachricht

Zu den Nachrichtendaten gehört der APDU-Befehl für das Issuer-Skript. Der Dienst überprüft den Inhalt dieses Felds nicht.

Neuer verschlüsselter PIN-Block

Der neue verschlüsselte PIN-Block, der an die Karte gesendet wird. Dies muss als verschlüsselter Wert unter Verwendung eines PIN-Verschlüsselungsschlüssels bereitgestellt werden.

Neue PIN (PEK-Kennung)

Der Schlüssel, der verwendet wird, um die neue PIN zu verschlüsseln, bevor sie an diese API übergeben wird.

Integritätsschlüssel für sichere Nachrichten

Der EMV-Integritätsschlüssel (KeyUsage: TR31_E2_EMV_MKEY_INTEGRITY), der für die MAC-Generierung verwendet wird.

Schlüssel zur Vertraulichkeit von Secure Messaging

Der EMV-Vertraulichkeitsschlüssel (KeyUsage: TR31_E4_EMV_MKEY_CONFIDENTIALITY), der für die PIN-Verschlüsselung verwendet wird.

MajorKeyDerivationMode

EMV definiert Modus A, B oder C. Modus A ist am gebräuchlichsten und Payment Cryptography unterstützt derzeit Modus A oder Modus B. AWS

Mode

Der Verschlüsselungsmodus, in der Regel CBC für PIN-Änderungsvorgänge.

PAN

Die Kontonummer, die normalerweise in Chipfeld 5A oder ISO8583, Feld 2, verfügbar ist, kann aber auch vom Kartensystem abgerufen werden.

PanSequenceNumber

Die Sequenznummer der Karte. Wenn nicht verwendet, geben Sie 00 ein.

ApplicationCryptogram

Dies sind die Ableitungsdaten pro Sitzung, in der Regel der letzte ARQC aus Feld 9F26.

PinBlockLengthPosition

Gibt an, wo die PIN-Blocklänge kodiert wird. In der Regel auf NONE gesetzt. Wenn Sie sich nicht sicher sind, überprüfen Sie die Spezifikationen Ihres Kartenschemas.

PinBlockPaddingType

Gibt den Füllungstyp für den PIN-Block an. In der Regel auf NO_PADDING gesetzt. Wenn Sie sich nicht sicher sind, überprüfen Sie die Spezifikationen Ihres Kartenschemas.

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"
}
```

Netzwerkspezifische Funktionen

Themen

- Visaspezifische Funktionen
- Mastercard-spezifische Funktionen
- American Express-spezifische Funktionen
- JCB-spezifische Funktionen
- UnionPay spezifische Funktionen

Visaspezifische Funktionen

Themen

- ARQC - CVN18/CVN22
- ARQC - CVN10
- 3DS CAVV V7
- dCVV (Dynamic Card Verification Value) — CVN17

ARQC - CVN18/CVN22

CVN18 und CVN22 verwenden die CSK-Methode der Schlüsselableitung. Die genauen Transaktionsdaten variieren zwischen diesen beiden Methoden. Einzelheiten zur Erstellung des Transaktionsdatenfeldes finden Sie in der Schema-Dokumentation.

ARQC - CVN10

CVN10 ist eine ältere Visa-Methode für EMV-Transaktionen, bei der der Schlüssel pro Karte und nicht pro Sitzung (pro Transaktion) abgeleitet wird und auch eine andere Nutzlast verwendet wird. Für weitere Informationen zum Inhalt der Payload wenden Sie sich bitte an das Schema.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
            "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"
            }
        }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/pw3s6nl62t5ushfk Das brauchst du im nächsten Schritt.

Validieren Sie den ARQC

Example

In diesem Beispiel validieren wir einen ARQC, der mit Visa CVN10 generiert wurde.

Wenn AWS Payment Cryptography den ARQC validieren kann, wird ein zurückgegeben. http/200 Wenn der Arqc nicht validiert wird, wird eine Antwort zurückgegeben. http/400

```
$ 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"
  }
```

3DS CAVV V7

Für Transaktionen mit Visa Secure (3DS) wird vom Access Control Server (ACS) des Ausstellers ein CAVV (Cardholder Authentication Verification Value) generiert. Der CAVV ist ein Beweis dafür, dass die Karteninhaberauthentifizierung stattgefunden hat. Er ist für jede Authentifizierungstransaktion eindeutig und wird vom Acquirer in der Autorisierungsnachricht bereitgestellt. CAVV v7 bindet zusätzliche Daten über die Transaktion an die Genehmigung, darunter Elemente wie den Händlernamen, den Kaufbetrag und das Kaufdatum. Auf diese Weise handelt es sich effektiv um einen kryptografischen Hash der Transaktionsnutzlast.

Kryptografisch verwendet CAVV V7 den CVV-Algorithmus, aber die Eingaben wurden alle verwendet. changed/repurposed Informationen zur Erzeugung der Eingaben zur Generierung einer party/Visa CAVV V7-Nutzlast finden Sie in der entsprechenden dritten Dokumentation.

Erstellen Sie den Schlüssel

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
            "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"
            }
        }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/dnaeyrjgdjjtw6dk Das brauchst du im nächsten Schritt.

Generieren Sie ein CAVV V7

Example

In diesem Beispiel generieren wir ein CAVV V7 für eine bestimmte Transaktion mit Eingaben, wie in den Spezifikationen angegeben. Beachten Sie, dass bei diesem Algorithmus Felder für andere used/re Zwecke verwendet werden können. Daher sollte nicht davon ausgegangen werden, dass die Feldbezeichnungen mit den Eingaben übereinstimmen.

Alle verfügbaren Parameter finden Sie CardVerificationValue1 im API-Referenzhandbuch.

```
$ 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"
            }
```

Validieren Sie CAVV V7

Example

Für die Validierung sind die Eingaben CVK, die berechneten Eingabewerte und das CAVV, das während der zu validierenden Transaktion bereitgestellt wurde.

Alle verfügbaren Parameter finden Sie CardVerificationValue1 im API-Referenzhandbuch.

> **Note**
>
> CAVV ist kein vom Benutzer eingegebener Wert (wie CVV2), sondern wird vom ACS des Ausstellers berechnet. Es sollte überlegt werden, ob der Wert immer gültig sein sollte, wenn er angegeben wird.

```
$ 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 (Dynamic Card Verification Value) — CVN17

dCVV (Dynamic Card Verification Value) ist ein Visa-specific dynamisches Kryptogramm, das für kontaktlose EMV-Transaktionen verwendet wird. Es ist als frühes EMV bekannt und bietet erhöhte Sicherheit, indem es für jede Transaktion einen eindeutigen Bestätigungswert generiert. Das DCVV verwendet Eingaben wie die primäre Kontonummer (PAN), die PAN-Sequenznummer (PSN), den Application Transaction Counter (ATC), das Ablaufdatum der Karte und den Servicecode. Es wird an einigen Stellen immer noch verwendet, wurde aber meistens durch andere Algorithmen wie CVN18 ersetzt.

Alle verfügbaren Parameter finden Sie DynamicCardVerificationValue im API-Referenzhandbuch.

Die Eingaben müssen `DynamicCardVerificationValue` den folgenden Visa-Autorisierungs- und EMV-Transaktionsdaten zugeordnet werden:

`ApplicationTransactionCounter`(ATC)

Der vom Chip verwaltete Transaktionszähler, der bei jeder Transaktion erhöht wird. In EMV-Daten ist dies ein Tag. `9F36`

`PanSequenceNumber`

Die PAN-Sequenznummer, die Karten unterscheidet, die dasselbe verwenden. PAN In EMV-Daten ist dies ein Tag. `5F34`

`CardExpiryDate`

Das Ablaufdatum der Kartenanwendung (EMV-Tag`5F24`), das in demselben Format bereitgestellt wird, das auch bei Ihren anderen Kartenverifizierungsanrufen verwendet wurde.

`ServiceCode`

Der dreistellige Servicecode, der in den Track-2-Daten (gleichwertig) der Karte enthalten ist.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
            "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"
        }
    }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/mw7dn3qxvkfh8ztc Das brauchst du im nächsten Schritt.

Generieren Sie ein dCVV

Example

In diesem Beispiel generieren wir ein dCVV für eine kontaktlose EMV-Transaktion. Zu den Eingaben gehören die PAN, die PAN-Sequenznummer, der Anwendungstransaktionszähler, das Ablaufdatum der Karte und der Servicecode.

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/mw7dn3qxvkfh8ztc \
    --primary-account-number=4111112627662122 \
    --generation-attributes
 DynamicCardVerificationValue='{ApplicationTransactionCounter=01,PanSequenceNumber=00,CardExpir
 \
    --validation-data-length 5
```

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

Validieren Sie DCvV

Example

In diesem Beispiel validieren wir ein DCvV, das während einer Transaktion bereitgestellt wurde. Dieselben Eingaben, die für die Generierung verwendet wurden, müssen für die Validierung bereitgestellt werden.

Wenn AWS Payment Cryptography validieren kann, http/200 wird an zurückgegeben. Wenn der Wert nicht validiert wird, wird eine http/400 Antwort zurückgegeben.

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/mw7dn3qxvkfh8ztc \
    --primary-account-number=4111112627662122 \
    --validation-data=36667 \
    --verification-attributes
 DynamicCardVerificationValue='{ApplicationTransactionCounter=01,PanSequenceNumber=00,CardExpir
```

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

Mastercard-spezifische Funktionen

Themen

- DCVC3
- ARQC -/ CVN14CVN15
- ARQC -/ CVN12CVN13
- 3DS AAV SPA2

DCVC3

DCVC3 ist älter als die EMV CSK- und CVN12 Mastercard-Schemata und stellt einen weiteren Ansatz für die Verwendung dynamischer Schlüssel dar. Es wird manchmal auch für andere Anwendungsfälle wiederverwendet. In diesem Schema sind die Eingaben PAN-, PSN-, Track1/ Track2-Daten, eine unvorhersehbare Zahl und ein Transaktionszähler (ATC).

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
            "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"
            }
        }
```

Notieren Sie sich den Wert, der den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east - 2:111122223333:key/hrh6qgbi3sk4y3wq. Das benötigen Sie im nächsten Schritt.

Generieren Sie ein DCVC3

Example

Obwohl DCVC3 es typischerweise durch eine Chipkarte generiert wird, kann es auch manuell generiert werden, wie in diesem Beispiel

```
$ 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"
  }
```

Bestätigen Sie das DCVC3

Example

In diesem Beispiel validieren wir eine DCVC3. Beachten Sie, dass ATC als Hexadezimalzahl angegeben werden sollte. Beispielsweise sollte ein Zähler von 11 als 000B dargestellt werden. Der Dienst erwartet eine DCVC3 dreistellige Zahl. Wenn Sie also einen 4- (oder 5) stelligen Wert gespeichert haben, kürzen Sie einfach die linken Zeichen, bis Sie 3 Ziffern haben (zum Beispiel sollte 15321 zu einem Validierungsdatenwert von 321 führen).

Wenn AWS Payment Cryptography validieren kann, wird ein http/200 zurückgegeben. Wenn der Wert nicht validiert wird, wird eine http/400-Antwort zurückgegeben.

```
$ 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 -/ CVN14CVN15

CVN14 und CVN15 verwenden die EMV-CSK-Methode zur Schlüsselableitung. Die genauen Transaktionsdaten variieren zwischen diesen beiden Methoden. Einzelheiten zur Erstellung des Transaktionsdatenfeldes finden Sie in der Schemadokumentation.

ARQC -/ CVN12CVN13

CVN12 und CVN13 sind eine ältere Mastercard-spezifische Methode für EMV-Transaktionen, die eine unvorhersehbare Zahl in die Ableitung pro Transaktion einbezieht und außerdem eine andere Nutzlast verwendet. Für Informationen zum Inhalt der Payload wenden Sie sich bitte an das System.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
            "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"
        }
    }
```

Notieren Sie sich den Wert, der den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east - 2:111122223333:key/pw3s6nl62t5ushfk. Das benötigen Sie im nächsten Schritt.

Validieren Sie den ARQC

Example

In diesem Beispiel validieren wir einen mit Mastercard generierten ARQC. CVN12

Wenn AWS Payment Cryptography den ARQC validieren kann, wird ein http/200 zurückgegeben. Wenn der ARQC nicht validiert wird, gibt er eine http/400-Antwort zurück.

```
$ 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
 00000000170000000000000000840008000800008401605170000000009380000B1F220103000000000000000000000
 \
  --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 AAV SPA2

SPA2(Secure Payment Application) AAV (Account Authentication Value) wird für Mastercard 3DS-Transaktionen (auch bekannt als Mastercard Identity Check) verwendet. Es bietet eine kryptografische Authentifizierung für E-Commerce-Transaktionen mithilfe der HMAC-basierten MAC-Generierung. Das AAV wird mithilfe transaktionsspezifischer Daten und eines gemeinsamen geheimen Schlüssels generiert.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
            "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"
        }
    }
```

Notieren Sie sich den Wert, der den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-west - 2:111122223333:key/q5vjtshsg67cz5gn. Das benötigen Sie im nächsten Schritt.

Generieren Sie SPA2 AAV

Example

In diesem Beispiel werden wir die IAV-Komponente (Issuer Authentication Value) der SPA2 AAV mithilfe der HMAC-MAC-Generierung generieren. Die Nachrichtendaten enthalten die transaktionsspezifischen Informationen, die authentifiziert werden. Das Format der Nachrichtendaten sollte den SPA2 Spezifikationen von Mastercard entsprechen und wird in diesem Beispiel nicht behandelt.

> **Note**
>
> Bitte überprüfen Sie Ihre Mastercard-Spezifikationen für die Formatierung, um die IAV in den AAV-Wert einzufügen.

```
$ 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"
}
```

Überprüfen Sie AAV SPA2

Example

In diesem Beispiel verifizieren wir ein SPA2 AAV. Dieselben Nachrichtendaten und der gleiche MAC-Wert werden zur Überprüfung bereitgestellt.

Wenn AWS Payment Cryptography den MAC validieren kann, wird ein http/200 zurückgegeben. Wenn der MAC nicht validiert ist, wird eine http/400-Antwort zurückgegeben.

```
$ 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-spezifische Funktionen

Themen

- CSC1
- CSC2
- iCSC
- 3DS AEVV

CSC1

CSC Version 1 wird auch als klassischer CSC-Algorithmus bezeichnet. Der Dienst kann sie als 3,4- oder 5-stellige Zahl bereitstellen.

Alle verfügbaren Parameter finden Sie unter AmexCardSecurityCodeVersion1 im API-Referenzhandbuch.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
                "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"
                }
            }
```

Notieren Sie sich den Wert, der den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east - 2:111122223333:key/esh6hn7pxdtttzgq. Das benötigen Sie im nächsten Schritt.

Generieren Sie ein CSC1

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"
  }
```

Bestätigen Sie das CSC1

Example

In diesem Beispiel validieren wir ein CSC1.

Wenn AWS Payment Cryptography validieren kann, wird ein http/200 zurückgegeben. Wenn der Wert nicht validiert wird, wird eine http/400-Antwort zurückgegeben.

```
$ 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 Version 2 wird auch als erweiterter CSC-Algorithmus bezeichnet. Der Dienst kann sie als 3,4- oder 5-stellige Zahl bereitstellen. Der Servicecode für CSC2 ist in der Regel 000.

Alle verfügbaren Parameter finden Sie unter AmexCardSecurityCodeVersion2 im API-Referenzhandbuch.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
            "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"
            }
        }
```

Notieren Sie sich den Wert, der den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east - 2:111122223333:key/erlm445qvunmvoda. Das brauchst du im nächsten Schritt.

Generieren Sie ein CSC2

In diesem Beispiel werden wir eine CSC2 mit einer Länge von 4 generieren. CSC kann mit einer Länge von 3,4 oder 5 generiert werden. Für American Express PANs sollte es 15 Ziffern sein und mit 34 oder 37 beginnen. Das Ablaufdatum wird normalerweise als JJMM formatiert. Der Servicecode kann variieren. Lesen Sie in Ihrem Handbuch nach, aber typische Werte sind 000, 201 oder 702

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"
}
```

Bestätigen Sie CSC2

Example

In diesem Beispiel validieren wir ein CSC2.

Wenn AWS Payment Cryptography validieren kann, wird ein http/200 zurückgegeben. Wenn der Wert nicht validiert wird, wird eine http/400-Antwort zurückgegeben.

```
$ 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 ist auch als statischer CSC-Algorithmus bekannt und wird mit CSC Version 2 berechnet. Der Dienst kann sie als 3,4- oder 5-stellige Zahl bereitstellen.

Verwenden Sie den Servicecode 999, um den iCSC für eine Kontaktkarte zu berechnen. Verwenden Sie den Servicecode 702, um den iCSC für eine kontaktlose Karte zu berechnen.

Alle verfügbaren Parameter finden Sie unter AmexCardSecurityCodeVersion2 im API-Referenzhandbuch.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
        "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"
        }
     }
```

Notieren Sie sich den Wert, der den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east - 1:111122223333:key/7vrybrbvjcvwtunv. Das benötigen Sie im nächsten Schritt.

Generieren Sie ein iCSC

In diesem Beispiel generieren wir ein iCSC mit einer Länge von 4 für eine kontaktlose Karte mit dem Servicecode 702. CSC kann mit einer Länge von 3,4 oder 5 generiert werden. Für American Express PANs sollte es 15 Ziffern sein und mit 34 oder 37 beginnen.

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"
```

```
}
```

Bestätigen Sie das iCSC

Example

In diesem Beispiel validieren wir ein iCSC.

Wenn AWS Payment Cryptography validiert werden kann, wird ein http/200 zurückgegeben. Wenn der Wert nicht validiert wird, wird eine http/400-Antwort zurückgegeben.

```
$ 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) wird für die American Express 3-D Secure-Authentifizierung verwendet. Es verwendet denselben Algorithmus wie, CSC2 jedoch mit unterschiedlichen Eingabeparametern. Das Feld mit dem Ablaufdatum sollte mit einer unvorhersehbaren (zufälligen) Zahl gefüllt werden, und der Servicecode besteht aus dem AEVV-Authentifizierungsergebniscode (1 Ziffer) und dem zweiten Faktor-Authentifizierungscode (2 Ziffern). Die Ausgabelänge sollte dreistellig sein.

Alle verfügbaren Parameter finden Sie unter AmexCardSecurityCodeVersion2 im API-Referenzhandbuch.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).


```
{
        "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"
        }
    }
```


Notieren Sie sich den Wert, der den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east - 2:111122223333:key/kw8djn5qxvfh3ztm. Das benötigen Sie im nächsten Schritt.

Generieren Sie ein 3DS AEVV

In diesem Beispiel werden wir ein 3DS-AEVV mit einer Länge von 3 generieren. Das Feld für das Ablaufdatum enthält eine unvorhersehbare (zufällige) Zahl (z. B. 1234), und der Servicecode

besteht aus dem AEVV-Authentifizierungsergebniscode (1 Ziffer) plus dem zweiten Faktor-Authentifizierungscode (2 Ziffern), zum Beispiel 543, wobei 5 der Authentifizierungsergebniscode und 43 der zweite Faktor-Authentifizierungscode ist. Für American Express PANs sollte es 15 Ziffern sein und mit 34 oder 37 beginnen.

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"
}
```

Validieren Sie den 3DS AEVV

Example

In diesem Beispiel validieren wir ein 3DS AEVV.

Wenn AWS Payment Cryptography validiert werden kann, wird ein http/200 zurückgegeben. Wenn der Wert nicht validiert wird, wird eine http/400-Antwort zurückgegeben.

```
$ 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-spezifische Funktionen

Themen

- ARQC - CVN04
- ARQC - CVN01

ARQC - CVN04

JCB CVN04 verwendet die CSK-Methode der Schlüsselableitung. ??? Einzelheiten zur Erstellung des Transaktionsdatenfeldes finden Sie in der Schema-Dokumentation.

ARQC - CVN01

CVN01 ist eine ältere JCB-Methode für EMV-Transaktionen, bei der der Schlüssel pro Karte und nicht pro Sitzung (pro Transaktion) abgeleitet wird und auch eine andere Nutzlast verwendet wird. Diese Nachricht wird auch von Visa verwendet, daher hat der Elementname diesen Namen, obwohl er auch für JCB verwendet wird. Informationen zum Inhalt der Nutzdaten erhalten Sie in der Schema-Dokumentation.

Schlüssel erstellen

```
$ 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"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe sowie eines Key Check Value (KCV).

```
{
                        "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"
                    }
                }
```

Notieren Sie sich das, was den Schlüssel darstelltKeyArn, zum Beispiel arn:aws:payment-cryptography:us-east- 2:111122223333:. key/pw3s6nl62t5ushfk Das brauchst du im nächsten Schritt.

Validieren Sie den ARQC

Example

In diesem Beispiel validieren wir einen ARQC, der mit JCB CVN01 generiert wurde. Dabei werden dieselben Optionen wie bei der Visa-Methode verwendet, daher der Name des Parameters.

Wenn AWS Payment Cryptography in der Lage ist, den ARQC zu validieren, wird ein http/200 zurückgegeben. Wenn der Arqc nicht validiert wird, wird eine Antwort zurückgegeben. http/400

```
$ 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"
              }
```

UnionPay spezifische Funktionen

Die folgenden Beispiele beziehen sich auf UnionPay International (UPI), manchmal auch als China UnionPay (CUP) bezeichnet.

Themen

- ARQC-Validierung
- CVN2 (Kartenprüfnummer)
- AVN (Authentifizierungsnummer)

ARQC-Validierung

UnionPay Die ARQC-Validierung verwendet den Schlüsselableitungsmodus EMV_OPTION_A. Der Sitzungsschlüssel wird mithilfe eines Schemas aus der PAN-Sequenznummer PrimaryAccountNumber und dem Anwendungstransaktionszähler abgeleitet. UnionPay-specific Informationen zum Transaktionsdatenformat und zur Reihenfolge der Felder finden Sie in der Schemadokumentation.

Validieren Sie den ARQC

Example Validieren Sie einen ARQC mithilfe UnionPay der Schlüsselableitung

Dieses Beispiel zeigt, wie ein ARQC validiert wird, der mithilfe der Schlüsselableitung generiert wurde. UnionPay

Wenn AWS Payment Cryptography den ARQC validiert, wird eine HTTP 200-Antwort zurückgegeben. Wenn der ARQC die Validierung nicht besteht, gibt AWS Payment Cryptography eine HTTP 400-Antwort mit einem Ursachencode zurück.

```
$ aws payment-cryptography-data verify-auth-request-cryptogram --auth-request-
cryptogram A12345B1A12345B1 \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
  --major-key-derivation-mode EMV_OPTION_A \
```

```
  --transaction-data 123456789ABCDEF80000000000000000 \
  --session-key-derivation-
attributes='UnionPay={PrimaryAccountNumber=4123412341234123,PanSequenceNumber=01,ApplicationTra
```

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

Wenn der ARQC nicht validiert werden kann, gibt AWS Payment Cryptography eine Fehlermeldung zurück:

```
An error occurred (VerificationFailedException) when calling the
 VerifyAuthRequestCryptogram operation: Auth request cryptogram verification failed.

Reason: INVALID_AUTH_REQUEST_CRYPTOGRAM
```

CVN2 (Kartenprüfnummer)

CVN2 ist das Äquivalent von CVV2 UnionPay . Es handelt sich um eine Kartenprüfnummer, die für Online-Transaktionen und für Transaktionen ohne Karte verwendet wird. Kryptografisch verwendet CVN2 denselben Algorithmus wie CVV2 (). CardVerificationValue2 Die Eingaben sind die primäre Kontonummer (PAN) und das Ablaufdatum der Karte (4 Ziffern im MMYY-Format).

Erstellen Sie den Schlüssel

Um einen CVN2-Schlüssel zu erstellen, verwenden Sie den `create-key` Befehl mit der `TR31_C0_CARD_VERIFICATION_KEY` Verwendung und `TDES_2KEY` dem Algorithmus.

```
$ 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":"CVN2"},{"Key":"CARD_BIN","Value":"47613499"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe und eines Key Check Value (KCV).

```
{
```

```
    "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"
    }
}
```

Notieren Sie sich den `KeyArn` Wert, zum Beispiel arn:aws:payment-cryptography:us-east-2:111122223333:. key/7f7g4spf3xcklhzu Das benötigen Sie in den folgenden Schritten.

Generieren Sie einen CVN2

Example Generieren Sie einen CVN2 für eine bestimmte PAN

In diesem Beispiel wird ein CVN2 für eine bestimmte PAN mit Eingaben von `PAN` und Gültigkeitsdatum der Karte generiert.

Alle verfügbaren Parameter finden Sie CardVerificationValue2 im API-Referenzhandbuch.

```
$ aws payment-cryptography-data generate-card-validation-data \
```

```
  --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu \
  --primary-account-number=4761349999991234 \
  --generation-attributes CardVerificationValue2='{CardExpiryDate=1226}'
```

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

Validieren Sie einen CVN2

Example Validieren Sie einen CVN2 für eine bestimmte PAN

In diesem Beispiel wird ein CVN2 für eine bestimmte PAN anhand der Eingaben vonPAN, dem Ablaufdatum der Karte und dem während der Transaktion angegebenen CVN2 validiert.

Alle verfügbaren Parameter finden Sie CardVerificationValue2 im API-Referenzhandbuch.

> **Note**
>
> CVN2 ist ein vom Benutzer eingegebener Wert. Regelmäßige Validierungsfehler deuten nicht unbedingt auf ein Problem hin.

```
$ aws payment-cryptography-data verify-card-validation-data \
  --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu \
  --primary-account-number=4761349999991234 \
  --verification-attributes CardVerificationValue2='{CardExpiryDate=1226}' \
  --validation-data 321
```

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

AVN (Authentifizierungsnummer)

AVN (Authentication Verification Number) ist das UnionPay 3DS-Kryptogramm. Es verwendet den CVV-Algorithmus (CardVerificationValue1) mit wiederverwendeten Eingabefeldern wie unten, aber die endgültigen Details finden Sie in der Schema-Dokumentation.

- PAN — Primäre Kontonummer (16-19 Ziffern, aus dem Bereich Q)
- CardExpiryDate— Die vier niedrigstwertigen Ziffern eines Werts, der von der DS-Transaktions-ID (dsTransID) abgeleitet wird
- ServiceCode— 3 Ziffern: Ergebnis der Authentifizierungstransaktion (1 Ziffer) + Füllbit '0' (1 Ziffer) + AV-Generatoranzeige (1 Ziffer)

Erstellen Sie den Schlüssel

Um einen AVN-Schlüssel zu erstellen, verwenden Sie den `create-key` Befehl mit der `TR31_C0_CARD_VERIFICATION_KEY` Verwendung und dem `TDES_2KEY` Algorithmus.

```
$ 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":"AVN"},{"Key":"CARD_BIN","Value":"47613499"}]'
```

Die Antwort gibt die Anforderungsparameter zurück, einschließlich eines ARN für nachfolgende Aufrufe und eines Key Check Value (KCV).

```
{
    "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"
    }
}
```

Notieren Sie sich den `KeyArn` Wert, zum Beispiel arn:aws:payment-cryptography:us-east-2:111122223333:. key/dnaeyrjgdjjtw6dk Das benötigen Sie in den folgenden Schritten.

Generieren Sie ein AVN

Example Generieren Sie eine AVN für eine bestimmte PAN

Dieses Beispiel generiert eine AVN für eine bestimmte PAN. Das `CardExpiryDate` Feld enthält die vier niedrigstwertigen Ziffern, die von der dsTransID abgeleitet wurden. Das `ServiceCode` Feld enthält das Authentifizierungsergebnis, das Füllbit und den Generatorindikator.

Alle verfügbaren Parameter finden Sie CardVerificationValue1 im API-Referenzhandbuch.

```
$ aws payment-cryptography-data generate-card-validation-data \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
dnaeyrjgdjjtw6dk \
  --primary-account-number=4761349999991234 \
  --generation-attributes
 CardVerificationValue1='{CardExpiryDate=9431,ServiceCode=201}'
```

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

Validieren Sie ein AVN

Example Validieren Sie eine AVN für eine bestimmte PAN

In diesem Beispiel wird eine AVN für eine bestimmte PAN anhand derselben Eingabefelder validiert, mit denen sie generiert wurde, zusammen mit dem zu überprüfenden AVN-Wert.

Alle verfügbaren Parameter finden Sie CardVerificationValue1 im API-Referenzhandbuch.

```
$ aws payment-cryptography-data verify-card-validation-data \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
dnaeyrjgdjjtw6dk \
  --primary-account-number=4761349999991234 \
  --verification-attributes
 CardVerificationValue1='{CardExpiryDate=9431,ServiceCode=201}' \
  --validation-data 491
```

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

Akquisitions- und Zahlungsvermittler

Acquirer PSPs und Zahlungsvermittler haben in der Regel andere kryptografische Anforderungen als Emittenten. Häufige Anwendungsfälle umfassen:

Entschlüsselung von Daten

Daten (insbesondere Pan-Daten) können von einem Zahlungsterminal verschlüsselt werden und müssen vom Backend entschlüsselt werden. Daten entschlüsseln und Daten verschlüsseln unterstützen eine Vielzahl von Methoden, darunter TDES-, AES- und DUKPT-Ableitungstechniken. Der AWS Payment Cryptography Service selbst ist ebenfalls PCI P2PE-konform und als PCI-P2PE-Entschlüsselungskomponente registriert.

TranslatePin

Um die PCI-PIN-Konformität zu gewährleisten, dürfen Acquiring-Systeme keine unverschlüsselten Karteninhaber-PINS haben, nachdem sie auf einem sicheren Gerät eingegeben wurden. Um die PIN vom Terminal an ein nachgeordnetes System (z. B. ein Zahlungsnetzwerk oder

einen Emittenten) weiterzuleiten, muss sie daher mit einem anderen Schlüssel als dem, den das Zahlungsterminal verwendet hat, erneut verschlüsselt werden. Translate Pin erreicht dies, indem eine verschlüsselte PIN mit dem servicebbb sicher von einem Schlüssel in einen anderen konvertiert wird. Mit diesem Befehl können Pins zwischen verschiedenen Schemata wie der TDES-, AES- und DUKPT-Ableitung und Pinblockformaten wie ISO-0, ISO-3 und ISO-4 konvertiert werden.

VerifyMac

Daten von einem Zahlungsterminal können mit einem MAC-Code versehen werden, um sicherzustellen, dass die Daten während der Übertragung nicht verändert wurden. Verify Mac und GenerateMac unterstützt eine Vielzahl von Techniken, die symmetrische Schlüssel verwenden, darunter TDES-, AES- und DUKPT-Ableitungstechniken zur Verwendung mit ISO-9797-1-Algorithmus 1, ISO-9797-1-Algorithmus 3 (Retail MAC) und CMAC-Techniken.

Weitere Themen

- Dynamische Schlüssel verwenden

Dynamische Schlüssel verwenden

Dynamic Keys ermöglicht die Verwendung von einmalig oder begrenzt verwendbaren Schlüsseln für kryptografische Operationen wie`EncryptData`. Dieser Ablauf kann genutzt werden, wenn das Schlüsselmaterial häufig rotiert (z. B. bei jeder Kartentransaktion) und der Import des Schlüsselmaterials in den Service vermieden werden soll. Kurzlebige Schlüssel können als Teil von SoftPoS/MPoC oder anderen Lösungen verwendet werden.

> **Note**
>
> Dies kann anstelle des typischen Ablaufs der AWS Zahlungskryptografie verwendet werden, bei dem kryptografische Schlüssel entweder erstellt oder in den Dienst importiert werden und Schlüssel mit einem Schlüsselalias oder einem Schlüssel-ARN spezifiziert werden.

Die folgenden Operationen unterstützen dynamische Schlüssel:

- EncryptData
- DecryptData
- ReEncryptData

- TranslatePin

Entschlüsseln von Daten

Das folgende Beispiel zeigt die Verwendung dynamischer Schlüssel zusammen mit dem Befehl decrypt. Die Schlüssel-ID ist in diesem Fall der Wrapping Key (KEK), der den Entschlüsselungsschlüssel sichert (der im Parameter wrapped-key im TR-31-Format bereitgestellt wird). Der verpackte Schlüssel muss der Hauptzweck von D0 sein und zusammen mit dem Befehl decrypt zusammen mit einem Verwendungsmodus von B oder D verwendet werden.

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"
}
```

Einen Pin übersetzen

Das folgende Beispiel zeigt die Verwendung von Dynamic Keys zusammen mit dem Befehl translate pin, um von einem dynamischen Schlüssel in einen semistatischen Acquirer-Working Key (AWK) zu übersetzen. Die Kennung des eingehenden Schlüssels ist in diesem Fall der Wrapping Key (KEK), der den dynamischen PIN-Verschlüsselungsschlüssel (PEK) schützt, der im TR-31-Format bereitgestellt wird. Der umhüllte Schlüssel muss `P0` zusammen mit dem Verwendungsmodus B oder D für den Hauptzweck dienen. Die Kennung des ausgehenden Schlüssels ist ein Schlüssel vom Typ `TR31_P0_PIN_ENCRYPTION_KEY` und der Verwendungsmodus Encrypt=True, Wrap=True

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"
}
```

Regionsspezifische Funktionen für AWS Kryptografie im Zahlungsverkehr

Bestimmte Funktionen können regionsspezifisch sein und nicht anderweitig verwendet werden. Diese Funktionen werden in diesem Abschnitt ausführlicher beschrieben.

ALS 2805

Der australische Standard 2805 (AS2805) ist ein Standard für elektronische Geldtransfers, der hauptsächlich für kartengestützte Zahlungsvorgänge verwendet wird. Er wird von Standards Australia verwaltet. https://www.standards.org.au/ Der Standard besteht aus 6 Büchern, die zahlreiche Themen vom Nachrichtenformat bis hin zu Verschlüsselungsstandards behandeln.

Teil 6 enthält Anleitungen zur Schlüsselverwaltung, einschließlich der Host-zu-Host-Kommunikation (von Knoten zu Knoten) und zu den relevanten kryptografischen Anforderungen. Andere Aspekte werden in anderen Teilen behandelt. Die gesamte Kryptografie in diesem Standard basiert derzeit auf TDES.

AS2805 weist im Vergleich zu anderen Implementierungen eine Reihe von Unterschieden auf, die im Folgenden zusammengefasst werden.

Wichtiger Schutz

: Verlässt sich auf wichtige Varianten statt auf Tastenblöcke wie in TR-31/X9.143. AWS Payment Cryptography speichert alle Schlüssel intern als Schlüsselblöcke, ermöglicht jedoch Import, Export und Berechnung mithilfe von AS2805-definierten Varianten.

Unidirektionale Schlüssel

: AS2805 schreibt die Verwendung unidirektionaler Schlüssel vor. Wenn beide Knoten Nachrichtenauthentifizierungscodes (MAC) generieren müssen, verwenden sie zwei Schlüssel.

Blöcke anheften

: AS2805 definiert eine Technik zur Schlüsselableitung für eindeutige PIN-Verschlüsselungsschlüssel pro Transaktion. Dies kann anstelle von DUKPT verwendet werden. Das AS2805-Schema stützt sich auf Transaktionsdaten (Trace-Nummer und Transaktionsbetrag), während DUKPT den Transaktionszähler verwendet.

Validierung des Schlüsselaustauschs

Definiert einen Prozess zur Validierung von KEK, bevor mit dem Austausch von Arbeitsschlüsseln wie PIN-Schlüsseln begonnen wird. In anderen Schemata werden KEK selten ausgetauscht und mit KCV validiert.

AS2805 verwendet das Konzept der Schlüsselvarianten anstelle von Schlüsselblöcken, um sicherzustellen, dass Schlüssel nur für den vorgesehenen (und einzigen) Zweck verwendet werden. Im Folgenden wird beschrieben, wie AWS Payment Cryptography Varianten und Schlüsselblöcken zuordnet, wenn Schlüssel importiert, exportiert oder andere kryptografische Funktionen ausgeführt werden.

AS2805-Schlüsseltyp	AWS Schlüsseltyp für Kryptografie im Zahlungsverkehr
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
DATENVERSCHLÜSSELUNG_SCHLÜS SEL_VARIANT_22	TR31_D0_SYMMETRISCHER_DATEN VERSCHLÜSSELUNGSSCHLÜSSEL
VARIANT_MASK_82, VARIANT_MASK_82C0	Optionen, die als Teil des KEK-Validierungspr ozesses verfügbar sind. Diese Schlüsseltypen sind kurzlebig und werden vom Dienst nicht gespeichert.

Bei zwei Knoten, node1 und node2, beziehen sich die folgenden Beispiele auf die Perspektive von node1. AWS Payment Cryptography unterstützt APIs von beiden Seiten des Prozesses.

Themen

- Austausch des Anfangsschlüssels (KEK)
- Validierung von KEK
- Erstellung und Übertragung von Arbeitstasten

- Funktionstüchtige Schlüssel exportieren
- Pin-Übersetzung
- Mac-Generierung und Validierung

Austausch des Anfangsschlüssels (KEK)

In AS2805 hat jede Seite ihren eigenen KEK. KEK (s) bezieht sich auf den Schlüssel der sendenden Seite, der immer dann verwendet wird, wenn die sendende Seite protect/wrap Schlüssel benötigt und sie an Node2 sendet. KEK (r) ist der Schlüssel, der von der Gegenseite (Node2) erzeugt wird.

> **Note**
>
> Diese Begriffe sind relativ — eine Seite erzeugt einen Schlüssel (sendende Seite) und die andere Seite empfängt ihn. Wenn also KEY1 gegeben ist, wird es auf Node1 als KEK (s) und auf Node2 als KEK (r) bezeichnet.

KEK für AS2805 haben immer den Schlüsseltyp = TR31_K0_KEY_ENCRYPTION_KEY, da sie zum Schutz von Kryptogrammen und nicht von Schlüsselblöcken verwendet werden. Dies entspricht TERMINAL_MAJOR_KEY_VARIANT_00, wie in AS2805 6.1 definiert

Schritte:

1.Erstellen Sie einen Schlüssel

Erstellen Sie einen Schlüssel mithilfe der CreateKey API. Sie erstellen einen Schlüssel vom Typ TR31_K0_KEY_ENCRYPTION_KEY

2.Bestimmen Sie die Methode zum Austausch von Schlüsseln mit node2

Ermitteln Sie, wie KEK mit der Gegenpartei ausgetauscht werden soll. Für AS2805 ist RSA Wrap die gebräuchlichste und interoperabelste Methode.

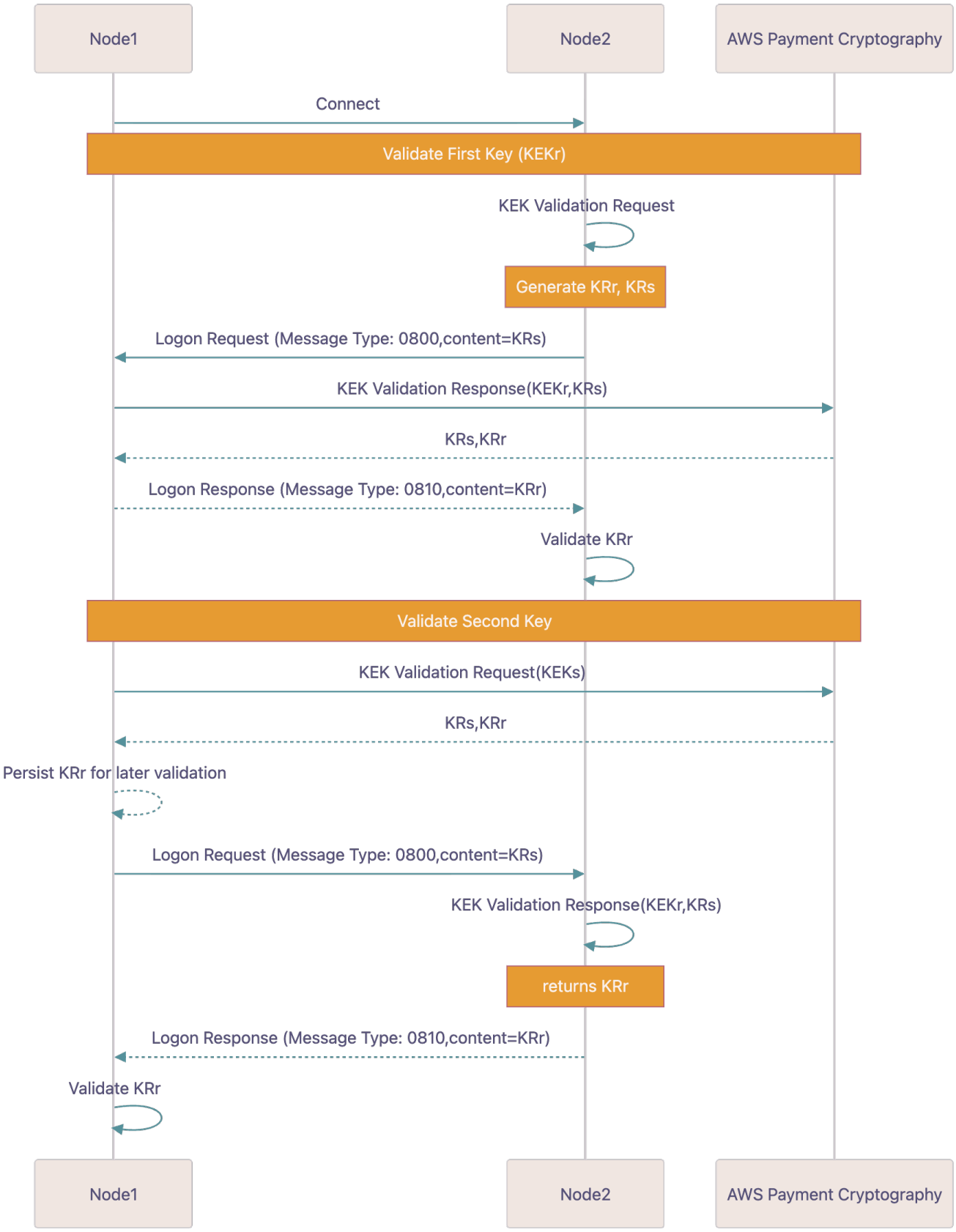
3. KEKs exportieren

Basierend auf Ihrer obigen Auswahl erhalten Sie von node2 ein Public-Key-Zertifikat. Sie führen den Export mit diesem Zertifikat aus, um den Schlüssel zu schützen (oder leiten einen Schlüssel ab, wenn Sie ECDH verwenden).

4. Importiere KeKr

Basierend auf Ihrer obigen Auswahl senden Sie ein Public-Key-Zertifikat an node2. Sie führen den Import mit diesem Zertifikat aus, um KeKR von Knoten 2 in den Dienst zu laden.

Validierung von KEK



Wenn Ihr Dienst (Node1) eine Verbindung zu Node2 herstellt, stellt jede Seite mithilfe eines Prozesses namens KEK-Validierung sicher, dass sie denselben KEK für nachfolgende Operationen verwenden.

1. Schritte zur Validierung des ersten Schlüssels

1.1 Empfangen Sie KRs

Node2 generiert ein KRs und sendet es Ihnen im Rahmen des Anmeldevorgangs. Sie können AWS Payment Cryptography verwenden, um diesen Wert oder eine andere Lösung zu generieren.

1.2 Generieren Sie eine KEK-Validierungsantwort

Ihr Knoten generiert eine KEK-Validierungsantwort mit Eingaben wie KEK (r) und KRs, die in Schritt 1 bereitgestellt wurden.

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 Gibt den berechneten KrR zurück

Gibt den berechneten KrR an Node2 zurück. Dieser Knoten vergleicht ihn mit dem berechneten Wert aus Schritt 1.

2.Schritte zur Validierung des zweiten Schlüssels

2.1 Generieren Sie KrR und KRs

Ihr Node generiert mithilfe von Payment Cryptography einen Zufallswert und eine invertierte (umgekehrte) Kopie dieses Werts. AWS Der Service gibt diese beiden Werte aus, die in den KEK (s) eingeschlossen sind. Diese sind als KR (s) und KR (r) bekannt.

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 Senden Sie KRs an Knoten2

Senden Sie die KRs an Knoten2. Bewahren Sie den KRR für eine spätere Validierung auf.

2.3 Node2 generiert eine KEK-Validierungsantwort

Node2 verwendet KeKr und KRs, generiert den KrR und sendet ihn an Ihren Dienst zurück.

2.4 Antwort validieren

Vergleichen Sie KrR aus Schritt 1 und den Wert, der aus Schritt 3 zurückgegeben wurde. Wenn sie übereinstimmen, fahren Sie fort.

Erstellung und Übertragung von Arbeitstasten

Zu den typischen Arbeitstasten, die in AS2805 verwendet werden, gehören zwei Tastensätze:

Schlüssel zwischen Knoten wie: Zonen-PIN-Schlüssel (ZPK), Zonenverschlüsselungsschlüssel (ZEK) und Zonenauthentifizierungsschlüssel (ZAK).

Schlüssel zwischen Terminals und Knoten wie: Terminal-Hauptschlüssel (TMK) und Terminal-PIN-Schlüssel (TPK), wenn DUKPT nicht verwendet wird.

> **Note**
>
> Wir empfehlen, die Anzahl der Schlüssel pro Terminal so gering wie möglich zu halten und Techniken wie TR-34 DUKPT zu nutzen, bei denen eine geringere Anzahl von Schlüsseln verwendet wird.

Example

In diesem Beispiel haben wir optionale Tags verwendet, um den Zweck und die Verwendung dieses Schlüssels nachzuverfolgen. Tags werden nicht als Teil der kryptografischen Funktion des Systems verwendet, können aber zur Kategorisierung, Finanzverfolgung und zur Anwendung von IAM-Richtlinien verwendet werden.

```
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 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"
 }
}
```

Funktionstüchtige Schlüssel exportieren

Um die Kompatibilität mit anderen Parteien aufrechtzuerhalten, unterstützt AWS Payment Cryptography die symmetrischen Schlüsselumschließungstechniken nach AS2805, bei denen Schlüsselvarianten anstelle von Keyblocks wie diesen verwendet werden. TR-31 Wenn mehrere Schlüssel von den Parteien gemeinsam genutzt werden, sollte jeder einzeln exportiert werden. Wenn Daten bidirektional gesendet werden, gibt es möglicherweise zwei Schlüssel zwischen Parteien desselben Typs, wie ZAK (s) und ZAK (r), die von beiden Seiten zur Generierung von Nachrichtenauthentifizierungscodes verwendet werden.

Die zusätzlichen Parameter für den Import und Export in diesen Formaten sind in den Befehlen angegeben.

```
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 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"
    }
}
```

Pin-Übersetzung

AS2805 beschreibt in Abschnitt 6.4 einen sitzungsspezifischen Modus zur Schlüsselableitung. Es dient einem ähnlichen Zweck wie DUKPT und jeder Algorithmus kann verwendet werden, da DUKPT in Abschnitt 6.7 behandelt wird. In diesem Schema wird ein Sitzungs-Pin-Schlüssel (bekannt als KPE) aus dem Terminal-Pin-Schlüssel abgeleitet, wobei SystemTraceAuditNumber (STAN) und TransactionAmount als Ableitungsdaten verwendet werden.

Die Funktion „PIN übersetzen“ ist eine gängige Funktion, mit der to/from eine Vielzahl von Formaten übersetzt werden können. In diesem Beispiel übersetzen wir eine PIN von einem KPE in einen PIN-Verschlüsselungsschlüssel (PEK), z. B. beim Senden einer PIN an ein Zahlungsnetzwerk.

```
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-Generierung und Validierung

Die MAC-Befehle zum Generieren und Überprüfen unterstützen eine Vielzahl von MACs, darunter HMAC, CMAC, EMV MAC usw. Für AS2805 ist eine zusätzliche Variante unter definiert. AS2805.4.1 In AS2805 werden eingehende Nachrichten in der Regel mit diesem MAC verifiziert, und ausgehende Nachrichten enthalten ebenfalls einen MAC.

```
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"
}
```

Sicherheit bei AWS Kryptografie im Zahlungsverkehr

Cloud-Sicherheit hat AWS höchste Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die auf die Anforderungen der sicherheitsrelevantesten Organisationen zugeschnitten sind.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der übergreifenden Verantwortlichkeit beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud —AWS ist für den Schutz der Infrastruktur verantwortlich, auf der AWS Dienste in der AWS Cloud ausgeführt werden. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Third-party Auditoren testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsvorkehrungen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für AWS Payment Cryptography gelten, finden Sie unter AWS-Services im Umfang nach Compliance-Programm .
- Sicherheit in der Cloud — Ihre Verantwortung hängt von dem AWS Dienst ab, den Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

In diesem Thema erfahren Sie, wie Sie das Modell der gemeinsamen Verantwortung bei der Verwendung von AWS Payment Cryptography anwenden können. Es zeigt Ihnen, wie Sie AWS Payment Cryptography so konfigurieren, dass Ihre Sicherheits- und Compliance-Ziele erreicht werden. Außerdem erfahren Sie, wie Sie andere AWS Dienste nutzen können, mit denen Sie Ihre AWS Payment Cryptography-Ressourcen überwachen und sichern können.

Topics

- Datenschutz in AWS Kryptografie im Zahlungsverkehr
- Widerstandsfähigkeit in AWS Kryptografie im Zahlungsverkehr
- Sicherheit der Infrastruktur in AWS Payment Cryptography
- Verbindung herstellen mit AWS Kryptografie für Zahlungen über einen VPC-Endpunkt
- Verwenden von Hybrid-Post-Quantum-TLS
- Bewährte Sicherheitsmethoden für AWS Kryptografie im Zahlungsverkehr

Datenschutz in AWS Kryptografie im Zahlungsverkehr

Das Modell der AWS geteilten Verantwortung (Modell der gilt für den Datenschutz in der AWS Zahlungskryptografie. Wie in diesem Modell beschrieben, AWS ist es für den Schutz der globalen Infrastruktur verantwortlich, auf der das AWS Cloud gesamte System läuft. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie in den häufig gestellten Fragen zum Datenschutz in der Region . Weitere Informationen zum Datenschutz in Europa finden Sie im Zentrum für die Datenschutz-Grundverordnung (DSGVO).

Aus Datenschutzgründen empfehlen wir, Ihre AWS-Konto Anmeldeinformationen zu schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einzurichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten findest du unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie AWS-Services mithilfe der Konsole, der API oder der SDKs mit AWS Payment Cryptography oder anderen Programmen arbeiten. AWS CLI AWS Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet

werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

AWS Payment Cryptography speichert und schützt Ihre Zahlungsverschlüsselungsschlüssel, um sie hochverfügbar zu machen und Ihnen gleichzeitig eine starke und flexible Zugriffskontrolle zu bieten.

Themen

- Schutz von Schlüsselmaterial
- Datenverschlüsselung
- Verschlüsselung im Ruhezustand
- Verschlüsselung während der Übertragung
- Richtlinie für den Datenverkehr zwischen Netzwerken

Schutz von Schlüsselmaterial

Standardmäßig schützt AWS Payment Cryptography das kryptografische Schlüsselmaterial für Zahlungsschlüssel, die vom Service verwaltet werden. Darüber hinaus bietet AWS Payment Cryptography Optionen für den Import von Schlüsselmaterial, das außerhalb des Dienstes erstellt wurde. Technische Details zu Zahlungsschlüsseln und Schlüsselmaterial finden Sie unter Kryptografische Details zu AWS Payment Cryptography.

Datenverschlüsselung

Die Daten in AWS Payment Cryptography bestehen aus den Schlüsseln von AWS Payment Cryptography, dem darin enthaltenen Verschlüsselungsschlüsselmaterial und ihren Nutzungsattributen. Schlüsselmaterial ist nur in den Hardware-Sicherheitsmodulen (HSMs) von AWS Payment Cryptography im Klartext vorhanden und nur, wenn es verwendet wird. Andernfalls werden das Schlüsselmaterial und die Attribute verschlüsselt und in einem dauerhaften, persistenten Speicher gespeichert.

Das Schlüsselmaterial, das AWS Payment Cryptography für Zahlungsschlüssel generiert oder lädt, verlässt niemals unverschlüsselt die Grenzen der HSMs von AWS Payment Cryptography. Es kann durch API-Operationen von AWS Payment Cryptography verschlüsselt exportiert werden.

Verschlüsselung im Ruhezustand

AWS Payment Cryptography generiert Schlüsselmaterial für Zahlungsschlüssel in HSM-listed PCI-PTS-HSMs. Bei Nichtgebrauch wird Schlüsselmaterial durch einen HSM-Schlüssel verschlüsselt und in dauerhaftem persistenten Speicher geschrieben. Das Schlüsselmaterial für Payment Cryptography-Schlüssel und die Verschlüsselungsschlüssel, die das Schlüsselmaterial schützen, verlassen die HSMs niemals im Klartext.

Die Verschlüsselung und Verwaltung des Schlüsselmaterials für Payment Cryptography-Schlüssel erfolgt vollständig durch den Dienst.

Weitere Informationen finden Sie unter Kryptografische Details zum AWS Key Management Service.

Verschlüsselung während der Übertragung

Schlüsselmaterial, das AWS Payment Cryptography für Zahlungsschlüssel generiert oder lädt, wird bei Vorgängen der AWS Payment Cryptography API niemals im Klartext exportiert oder übertragen. AWS Payment Cryptography verwendet Schlüsselkennungen, um die Schlüssel in API-Vorgängen darzustellen.

Bei einigen API-Vorgängen werden jedoch Schlüssel exportiert, die mit einem zuvor gemeinsam genutzten oder asymmetrischen Schlüssel für den Schlüsselaustausch verschlüsselt wurden. Außerdem können Kunden API-Operationen verwenden, um verschlüsseltes Schlüsselmaterial für Zahlungsschlüssel zu importieren.

Alle API-Aufrufe der AWS Payment Cryptography müssen signiert und mithilfe von Transport Layer Security (TLS) übertragen werden. AWS Payment Cryptography erfordert TLS-Versionen und Verschlüsselungssammlungen, die von PCI als „starke Kryptografie“ definiert werden. Alle Serviceendpunkte unterstützen TLS 1.2—1.3 und hybrides Post-Quantum-TLS.

Weitere Informationen finden Sie unter Kryptografische Details zum AWS Key Management Service.

Richtlinie für den Datenverkehr zwischen Netzwerken

AWS Payment Cryptography unterstützt eine AWS-Managementkonsole und eine Reihe von API-Vorgängen, mit denen Sie Zahlungsschlüssel erstellen und verwalten und für kryptografische Operationen verwenden können.

AWS Payment Cryptography unterstützt zwei Netzwerkverbindungsoptionen von Ihrem privaten Netzwerk zu AWS.

- Eine IPSec-VPN-Verbindung über das Internet.
- AWS Direct Connect, das Ihr internes Netzwerk über ein Standard-Ethernet-Glasfaserkabel mit einem AWS Direct Connect-Standort verbindet.

Alle Payment Cryptography API-Aufrufe müssen signiert und mithilfe von Transport Layer Security (TLS) übertragen werden. Die Anrufe erfordern auch eine moderne Verschlüsselungssammlung, die Perfect Forward Secrecy unterstützt. Der Datenverkehr zu den Hardware-Sicherheitsmodulen (HSMs), in denen Schlüsselmaterial für Zahlungsschlüssel gespeichert ist, ist nur von bekannten AWS Payment Cryptography API-Hosts über das interne AWS-Netzwerk zulässig.

Verwenden Sie VPC-Endpunkte, die von AWS bereitgestellt werden, um von Ihrer Virtual Private Cloud (VPC) aus eine direkte Verbindung zu AWS Payment Cryptography herzustellen, ohne Datenverkehr über das öffentliche Internet zu senden. PrivateLink Weitere Informationen finden Sie unter Herstellen einer Verbindung zu AWS Payment Cryptography über einen VPC-Endpunkt.

AWS Payment Cryptography unterstützt auch eine hybride Post-Quantum-Schlüsselaustauschoption für das Netzwerkverschlüsselungsprotokoll Transport Layer Security (TLS). Sie können diese Option mit TLS verwenden, wenn Sie eine Verbindung zu den API-Endpunkten der AWS Payment Cryptography herstellen.

Widerstandsfähigkeit in AWS Kryptografie im Zahlungsverkehr

AWS Die globale Infrastruktur basiert auf AWS Regionen und Verfügbarkeitszonen. Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, die über hoch redundante Netzwerke mit niedriger Latenz und hohen Durchsätzen verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Regionale Isolierung

AWS Payment Cryptography ist ein regionaler Service, der in mehreren Regionen verfügbar ist.

Das regional isolierte Design von AWS Payment Cryptography stellt sicher, dass ein Verfügbarkeitsproblem in einer AWS-Region den Betrieb von AWS Payment Cryptography in einer

anderen Region nicht beeinträchtigen kann. AWS Payment Cryptography ist so konzipiert, dass keine geplanten Ausfallzeiten entstehen, da alle Software-Updates und Skalierungsvorgänge nahtlos und unmerklich durchgeführt werden.

Das AWS Payment Cryptography Service Level Agreement (SLA) beinhaltet eine Serviceverpflichtung von 99,99% für alle Payment Cryptography APIs. Um dieser Verpflichtung nachzukommen, stellt AWS Payment Cryptography sicher, dass alle Daten und Autorisierungsinformationen, die für die Ausführung einer API-Anfrage erforderlich sind, auf allen regionalen Hosts verfügbar sind, die die Anfrage erhalten.

Die Infrastruktur von AWS Payment Cryptography wird in mindestens drei Availability Zones (AZs) in jeder Region repliziert. Um sicherzustellen, dass der Ausfall mehrerer Hosts die Leistung von AWS Payment Cryptography nicht beeinträchtigt, ist AWS Payment Cryptography so konzipiert, dass der Kundenverkehr von allen AZs in einer Region verarbeitet wird.

Änderungen, die Sie an den Eigenschaften oder Berechtigungen eines Zahlungsschlüssels vornehmen, werden auf alle Hosts in der Region repliziert, um sicherzustellen, dass nachfolgende Anfragen von jedem Host in der Region korrekt verarbeitet werden können. Anfragen für kryptografische Operationen unter Verwendung Ihres Zahlungsschlüssels werden an eine Flotte von Hardware-Sicherheitsmodulen (HSMs) von AWS Payment Cryptography weitergeleitet, von denen jedes den Vorgang mit dem Zahlungsschlüssel ausführen kann.

Multi-tenant entwerfen

Das mehrinstanzenfähige Design von AWS Payment Cryptography ermöglicht es, die Verfügbarkeits-SLA zu erfüllen und hohe Anforderungsraten aufrechtzuerhalten, während gleichzeitig die Vertraulichkeit Ihrer Schlüssel und Daten geschützt wird.

Es werden mehrere Mechanismen zur Durchsetzung der Integrität eingesetzt, um sicherzustellen, dass der Zahlungsschlüssel, den Sie für den kryptografischen Vorgang angegeben haben, immer der ist, der verwendet wird.

Das Klartext-Schlüsselmaterial für Ihre Payment Cryptography Keys ist umfassend geschützt. Das Schlüsselmaterial wird im HSM verschlüsselt, sobald es erstellt wurde, und das verschlüsselte Schlüsselmaterial wird sofort in einen sicheren Speicher verschoben. Der verschlüsselte Schlüssel wird im HSM abgerufen und entschlüsselt, sobald er benutzt wird. Der Klartextschlüssel verbleibt nur so lange im HSM-Speicher, wie er für den Abschluss der kryptografischen Operation benötigt wird. Das Klartext-Schlüsselmaterial verlässt die HSMs nie; es wird nie in den permanenten Speicher geschrieben.

Weitere Informationen zu den Mechanismen, die AWS Payment Cryptography zur Sicherung Ihrer Schlüssel verwendet, finden Sie unter Kryptografische Details von AWS Payment Cryptography.

Sicherheit der Infrastruktur in AWS Payment Cryptography

Als verwalteter Service AWS Payment Cryptography ist er durch die AWS globalen Netzwerksicherheitsverfahren geschützt, die im Whitepaper Amazon Web Services: Sicherheitsprozesse im Überblick beschrieben sind.

Für den Zugriff AWS Payment Cryptography über das Netzwerk verwenden Sie AWS veröffentlichte API-Aufrufe. Clients müssen Transport Layer Security (TLS) 1.2 oder höher unterstützen. Clients müssen außerdem Verschlüsselungssammlungen mit Perfect Forward Secrecy (PFS) wie Ephemeral (DHE) oder Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) unterstützen. Diffie-Hellman Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS -Security-Token-Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Isolierung physischer Hosts

Die Sicherheit der physischen Infrastruktur, die AWS Payment Cryptography verwendet, unterliegt den Kontrollen, die im Abschnitt Physische Sicherheit und Umgebungssicherheit von Amazon Web Services: Überblick über Sicherheitsprozesse beschrieben sind. Weitere Details finden Sie in Compliance-Berichten und Prüfungsergebnissen von Drittanbietern, die im vorherigen Abschnitt aufgeführt sind.

AWS Payment Cryptography wird von speziellen handelsüblichen HSM-listed PCI-PTS-Hardwaresicherheitsmodulen (HSMs) unterstützt. Das Schlüsselmaterial für die Schlüssel von AWS Payment Cryptography wird nur im flüchtigen Speicher der HSMs gespeichert, und zwar nur, solange der Payment Cryptography-Schlüssel verwendet wird. HSMs befinden sich in zugriffskontrollierten Racks in Amazon-Rechenzentren, die eine doppelte Kontrolle für jeden physischen Zugriff vorschreiben. Ausführliche Informationen zum Betrieb von AWS Payment Cryptography HSMs finden Sie unter Kryptografische Details von AWS Payment Cryptography.

Verbindung herstellen mit AWS Kryptografie für Zahlungen über einen VPC-Endpunkt

Sie können über einen privaten Schnittstellenendpunkt in Ihrer Virtual Private Cloud (VPC) eine direkte Verbindung zu AWS Payment Cryptography herstellen. Wenn Sie einen VPC-Schnittstellen-Endpunkt verwenden, erfolgt die Kommunikation zwischen Ihrer VPC und AWS Payment Cryptography vollständig innerhalb des Netzwerks. AWS

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

Der VPC-Endpunkt der Schnittstelle verbindet Ihre VPC direkt mit AWS Payment Cryptography, ohne dass ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder eine Verbindung erforderlich ist. AWS Direct Connect Die Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen, um mit Payment Cryptography zu kommunizieren. AWS

Regionen

AWS Payment Cryptography unterstützt VPC-Endpunkte und VPC-Endpunktrichtlinien in allen AWS-Regionen Fällen, in denen Payment Cryptography unterstützt wird. AWS

Themen

- Überlegungen für AWS VPC-Endpunkte für Zahlungskryptografie
- Erstellen eines VPC-Endpunkts für AWS Kryptografie im Zahlungsverkehr
- Verbindung zu einem AWS VPC-Endpunkt für Zahlungskryptografie
- Steuern des Zugriffs auf einen VPC-Endpunkt
- Verwenden eines VPC-Endpunkts in einer Richtlinienanweisung
- Protokollieren des VPC-Endpunkts

Überlegungen für AWS VPC-Endpunkte für Zahlungskryptografie

> **Note**
>
> Mit VPC-Endpunkten können Sie zwar in nur einer Availability Zone (AZ) eine Verbindung zum Service herstellen, wir empfehlen jedoch, aus Hochverfügbarkeits- und Redundanzgründen eine Verbindung zu drei Verfügbarkeitszonen herzustellen.

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

AWS Die Unterstützung von Payment Cryptography für einen VPC-Endpunkt umfasst Folgendes.

- Sie können Ihren VPC-Endpunkt verwenden, um alle Operationen auf der Ebene „AWS Payment Cryptography Control“ und „AWS Payment Cryptography Data Plane“ von einer VPC aus aufzurufen.
- Sie können einen VPC-Schnittstellen-Endpunkt erstellen, der eine Verbindung zu einem AWS Endpunkt der Payment Cryptography-Region herstellt.
- AWS Payment Cryptography besteht aus einer Steuerungsebene und einer Datenebene. Sie können wählen, ob Sie einen oder beide Unterdienste einrichten möchten, AWS PrivateLink aber jeder wird separat konfiguriert.
- Sie können AWS CloudTrail Protokolle verwenden, um Ihre Verwendung von AWS Payment Cryptography-Schlüsseln über den VPC-Endpunkt zu überprüfen. Details hierzu finden Sie unter Protokollieren des VPC-Endpunkts.

Erstellen eines VPC-Endpunkts für AWS Kryptografie im Zahlungsverkehr

Sie können einen VPC-Endpunkt für AWS Zahlungskryptografie mithilfe der Amazon VPC-Konsole oder der Amazon VPC-API erstellen. Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im AWS PrivateLink -Leitfaden.

- Verwenden Sie die folgenden Servicenamen, um einen VPC-Endpunkt für AWS Payment Cryptography zu erstellen:

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

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

In der Region USA West (Oregon) () würden die Servicenamen beispielsweise wie folgt lauten: `us-west-2`

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

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

Um die Verwendung des VPC-Endpunkts zu vereinfachen, können Sie einen privaten DNS-Namen für Ihren VPC-Endpunkt aktivieren. Wenn Sie die Option „DNS-Namen aktivieren“ auswählen, wird der Standard-DNS-Hostname für AWS Payment Cryptography in Ihren VPC-Endpunkt aufgelöst. `https://controlplane.payment-cryptography.us-west-2.amazonaws.com` würde beispielsweise in einen VPC-Endpunkt aufgelöst, der mit dem Servicenamen `com.amazonaws.us-west-2.payment-cryptography.controlplane` verbunden ist.

Diese Option vereinfacht die Verwendung des VPC-Endpunkts. Die AWS SDKs AWS CLI verwenden standardmäßig den Standard-DNS-Hostnamen von AWS Payment Cryptography, sodass Sie die VPC-Endpunkt-URL nicht in Anwendungen und Befehlen angeben müssen.

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

Verbindung zu einem AWS VPC-Endpunkt für Zahlungskryptografie

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

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

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

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

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

Steuern des Zugriffs auf einen VPC-Endpunkt

Um den Zugriff auf Ihren VPC-Endpunkt für AWS Payment Cryptography zu kontrollieren, fügen Sie Ihrem VPC-Endpunkt eine VPC-Endpunktrichtlinie hinzu. Die Endpunktrichtlinie bestimmt, ob Principals den VPC-Endpunkt verwenden können, um Payment Cryptography-Operationen mit bestimmten AWS Payment Cryptography-Ressourcen aufzurufen. AWS

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

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

Themen

- Weitere Informationen über VPC-Endpunktrichtlinien
- Standard-VPC-Endpunktrichtlinie
- Erstellen einer VPC-Endpunktrichtlinie
- Anzeigen einer VPC-Endpunktrichtlinie

Weitere Informationen über VPC-Endpunktrichtlinien

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

- Eine identitätsbasierte Richtlinie muss dem Prinzipal die Erlaubnis geben, den Vorgang für die Ressource (AWS Payment Cryptography Keys oder Alias) aufzurufen.
- Eine VPC-Endpunktrichtlinie muss dem Prinzipal die Berechtigung erteilen, den Endpunkt für die Anforderung zu verwenden.

Beispielsweise könnte eine Schlüsselrichtlinie einem Prinzipal die Erlaubnis geben, Decrypt für bestimmte AWS Payment Cryptography -Schlüssel aufzurufen. Die VPC-Endpunktrichtlinie erlaubt es diesem Principal jedoch möglicherweise nicht, diese AWS Payment `Decrypt` Cryptography-Schlüssel mithilfe des Endpunkts abzurufen.

Oder eine VPC-Endpunktrichtlinie könnte es einem Principal ermöglichen, den Endpunkt zu verwenden, um bestimmte AWS Payment StopKeyUsage Cryptography-Schlüssel abzurufen. Wenn der Prinzipal jedoch nicht über diese Berechtigungen aus einer IAM-Richtlinie verfügt, schlägt die Anfrage fehl.

Standard-VPC-Endpunktrichtlinie

Jeder VPC-Endpunkt verfügt über eine VPC-Endpunktrichtlinie. Sie müssen die Richtlinie jedoch nicht angeben. Wenn Sie keine Richtlinie angeben, erlaubt die standardmäßige Endpunktrichtlinie alle Operationen aller Prinzipale auf allen Ressourcen über den Endpunkt.

Für AWS Payment Cryptography-Ressourcen muss der Principal jedoch auch über die Berechtigung verfügen, den Vorgang von einer IAM-Richtlinie aus aufzurufen. Daher besagt die Standardrichtlinie in der Praxis, dass, wenn ein Prinzipal über die Berechtigung zum Aufrufen einer Operation für eine Ressource verfügt, diese auch mithilfe des Endpunkts aufrufen kann.

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

```
}
```

Damit Prinzipale den VPC-Endpunkt nur für eine Teilmenge ihrer zulässigen Operationen verwenden können, erstellen oder aktualisieren Sie die VPC-Endpunktrichtlinie.

Erstellen einer VPC-Endpunktrichtlinie

Eine VPC-Endpunktrichtlinie bestimmt, ob ein Prinzipal die Berechtigung hat, den VPC-Endpunkt zum Ausführen von Operationen auf einer Ressource zu verwenden. Für AWS Payment Cryptography-Ressourcen muss der Auftraggeber außerdem über die Berechtigung verfügen, die Operationen aus einer IAM-Richtlinie heraus auszuführen.

Für jede VPC-Endpunktrichtlinie sind die folgenden Elemente erforderlich:

- Der Prinzipal, der die Aktionen ausführen kann
- Aktionen, die ausgeführt werden können
- Ressourcen, für die Aktionen ausgeführt werden können

Die Richtlinienanweisung gibt den VPC-Endpunkt nicht an. Stattdessen gilt sie für jeden VPC-Endpunkt, dem die Richtlinie angefügt ist. Weitere Informationen finden Sie unter Steuerung des Zugriffs auf Services mit VPC-Endpunkten im Amazon-VPC-Benutzerhandbuch.

Im Folgenden finden Sie ein Beispiel für eine VPC-Endpunktrichtlinie für AWS Payment Cryptography. Wenn diese Richtlinie an einen VPC-Endpunkt angehängt ist, ermöglicht sie die Verwendung des VPC-Endpunkts `ExampleUser` zum Aufrufen der angegebenen Operationen mit den angegebenen Payment Cryptography-Schlüsseln. AWS Bevor Sie eine Richtlinie wie diese verwenden, ersetzen Sie den Beispielprinzipal und die Schlüssel-ID durch gültige Werte aus Ihrem Konto.

```
{
   "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 protokolliert alle Vorgänge, die den VPC-Endpunkt verwenden. Ihre CloudTrail Protokolle enthalten jedoch keine Operationen, die von Auftraggebern in anderen Konten angefordert wurden, oder Operationen für AWS Payment Cryptography-Schlüssel in anderen Konten.

Daher sollten Sie möglicherweise eine VPC-Endpunktrichtlinie erstellen, die verhindert, dass Prinzipale in externen Konten den VPC-Endpunkt verwenden, um AWS Payment Cryptography-Operationen für beliebige Schlüssel im lokalen Konto aufzurufen.

Im folgenden Beispiel wird der PrincipalAccount globale Bedingungsschlüssel aws: verwendet, um allen Prinzipalen den Zugriff auf alle Operationen mit allen AWS Payment Cryptography-Schlüsseln zu verweigern, sofern sich der Prinzipal nicht im lokalen Konto befindet. Bevor Sie eine Richtlinie wie diese verwenden, ersetzen Sie die Beispiel-Konto-ID durch eine gültige.

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

Anzeigen einer VPC-Endpunktrichtlinie

Verwenden Sie die VPC-Managementkonsole oder den Vorgang, um die VPC-Endpunktrichtlinie für einen Endpunkt anzuzeigen. DescribeVpcEndpoints

Mit dem folgenden AWS CLI Befehl wird die Richtlinie für den Endpunkt mit der angegebenen VPC-Endpunkt-ID abgerufen.

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

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==`vpce-1234abcdf5678c90a`].[PolicyDocument]'
--output text
```

Verwenden eines VPC-Endpunkts in einer Richtlinienanweisung

Sie können den Zugriff auf AWS Payment Cryptography-Ressourcen und -Operationen kontrollieren, wenn die Anfrage von einer VPC kommt oder einen VPC-Endpunkt verwendet. Verwenden Sie dazu eine IAM-Richtlinie ???

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

> **Note**
>
> Der `aws:sourceIP` Bedingungsschlüssel ist nicht wirksam, wenn die Anfrage von einem Amazon VPC-Endpunkt kommt. Um die Anforderungen an einen VPC-Endpunkt zu beschränken, verwenden Sie die Bedingungsschlüssel `aws:sourceVpce` oder `aws:sourceVpc`. Weitere Informationen finden Sie unter Identity and Access Management für VPC-Endpunkte und VPC-Endpunkt-Services im AWS PrivateLink -Leitfaden.

Sie können diese globalen Bedingungsschlüssel verwenden, um den Zugriff auf Schlüssel und Aliase von AWS Payment Cryptography zu kontrollieren, und zwar auf Operationen, CreateKey die nicht von einer bestimmten Ressource abhängen.

Die folgende Beispiel-Schlüsselrichtlinie ermöglicht es einem Benutzer beispielsweise, bestimmte kryptografische Operationen mit einem AWS Payment Cryptography-Schlüssel nur durchzuführen, wenn die Anforderung den angegebenen VPC-Endpunkt verwendet. Dadurch wird der Zugriff sowohl

über das Internet als auch über AWS PrivateLink Verbindungen (sofern eingerichtet) blockiert. Wenn ein Benutzer eine Anfrage an AWS Payment Cryptography stellt, wird die VPC-Endpunkt-ID in der Anfrage mit dem `aws:sourceVpce` Bedingungsschlüsselwert in der Richtlinie verglichen. Wenn sie nicht übereinstimmen, wird die Anforderung abgelehnt.

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

JSON

```
{
    "Id": "example-key-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableIAMPolicies",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "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"
                }
            }
```

```
        }

    ]
}
```

Sie können den `aws:sourceVpc` Bedingungsschlüssel auch verwenden, um den Zugriff auf Ihre AWS Payment Cryptography-Schlüssel auf der Grundlage der VPC einzuschränken, in der sich der VPC-Endpunkt befindet.

Die folgende Beispiel-Schlüsselrichtlinie erlaubt Befehle, die die AWS Payment Cryptography-Schlüssel verwalten, nur dann, wenn sie von dort stammen. `vpc-12345678` Außerdem sind Befehle, die die AWS Payment Cryptography-Schlüssel für kryptografische Operationen verwenden, nur zulässig, wenn sie von dort stammen. `vpc-2b2b2b2b` Sie verwenden eine solche Richtlinie wie diese möglicherweise, wenn eine Anwendung in einer VPC ausgeführt wird, aber eine zweite, isolierte VPC für die Verwaltungsfunktionen genutzt wird.

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

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": "*"
        }
    ]
}
```

Protokollieren des VPC-Endpunkts

AWS CloudTrail protokolliert alle Vorgänge, die den VPC-Endpunkt verwenden. Wenn eine Anfrage an AWS Payment Cryptography einen VPC-Endpunkt verwendet, erscheint die VPC-Endpunkt-ID

in dem AWS CloudTrail Protokolleintrag, der die Anfrage aufzeichnet. Sie können die Endpunkt-ID verwenden, um die Nutzung Ihres AWS Payment Cryptography VPC-Endpunkts zu überprüfen.

Um Ihre VPC zu schützen, werden Anfragen, die durch eine VPC-Endpunktrichtlinie abgelehnt wurden, aber andernfalls zulässig gewesen wären, nicht aufgezeichnet. AWS CloudTrail

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

```
{
      "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"
```

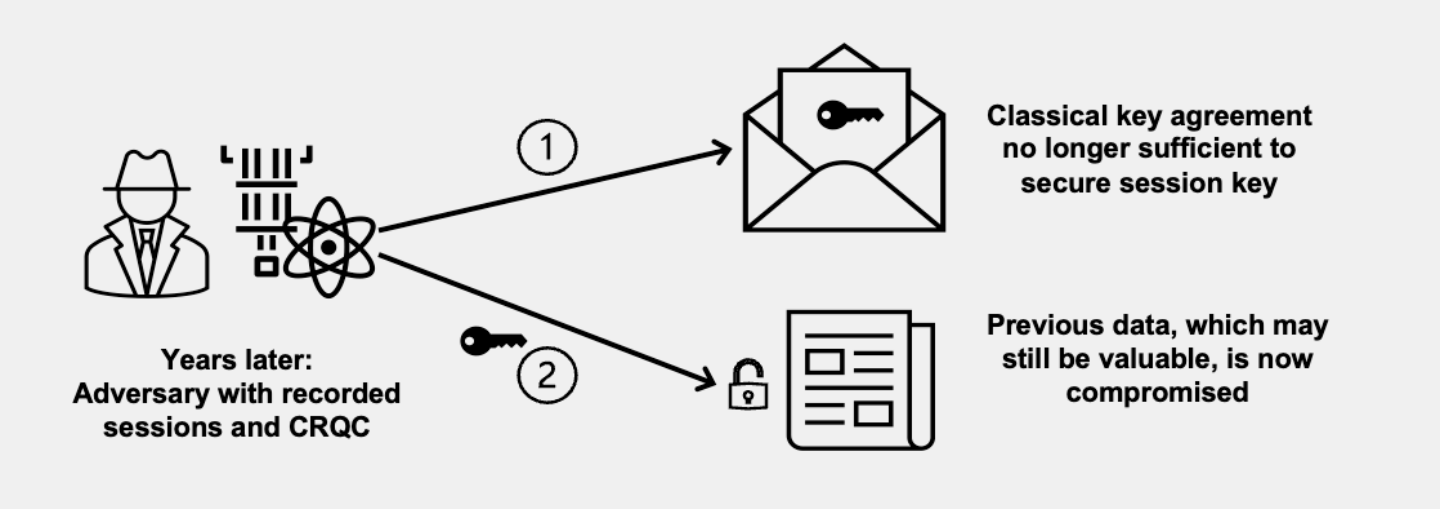
```
          }
      },
      "requestID": "f3020b3c-4e86-47f5-808f-14c7a4a99161",
      "eventID": "b87c3d30-f3ab-4131-87e8-bc54cfef9d29",
      "readOnly": false,
      "eventType": "AwsApiCall",
      "managementEvent": true,
      "recipientAccountId": "111122223333",
      "vpcEndpointId": "vpce-1234abcdf5678c90a",
      "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"
      }
  }
```

Verwenden von Hybrid-Post-Quantum-TLS

AWS Payment Cryptography und viele andere Dienste unterstützen eine hybride Post-Quantum-Schlüsselaustauschoption für das Netzwerkverschlüsselungsprotokoll Transport Layer Security (TLS). Sie können diese TLS-Option verwenden, wenn Sie eine Verbindung zu API-Endpunkten herstellen oder wenn Sie die SDKs verwenden. AWS Diese optionalen Hybrid-Post-Quantum-Schlüsselaustauschfunktionen sind mindestens so sicher wie die heute verwendete TLS-Verschlüsselung und bieten wahrscheinlich zusätzliche langfristige Sicherheitsvorteile.

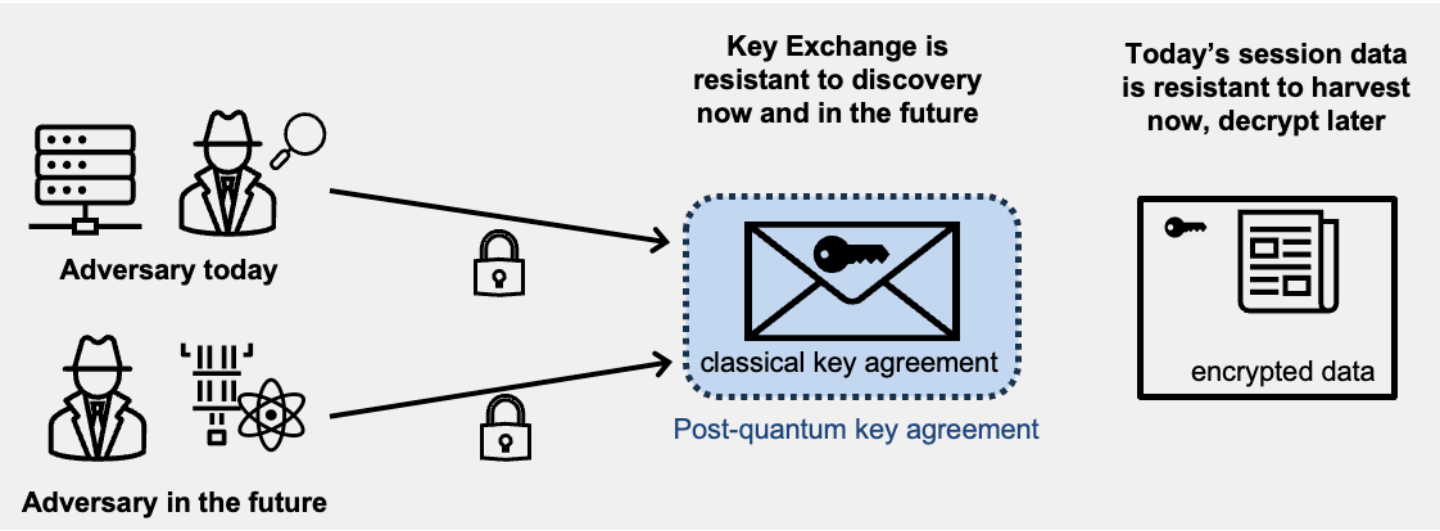
Die Daten, die Sie an aktivierte Dienste senden, sind bei der Übertragung durch die Verschlüsselung geschützt, die über eine Transport Layer Security (TLS) -Verbindung bereitgestellt wird. Die klassischen Verschlüsselungssammlungen, die auf RSA und ECC basieren und die AWS Payment Cryptography für TLS-Sitzungen unterstützt, machen Brute-Force-Angriffe auf die Schlüsselaustauschmechanismen mit der aktuellen Technologie unmöglich. Wenn jedoch in Zukunft große oder kryptografisch relevante Quantencomputer (CRQC) praktikabel werden, werden die bestehenden TLS-Schlüsselaustauschmechanismen anfällig für diese Angriffe sein. Es ist möglich, dass Gegner jetzt damit beginnen, verschlüsselte Daten zu sammeln, in der Hoffnung, sie in Zukunft entschlüsseln zu können (jetzt sammeln, später entschlüsseln). Wenn Sie Anwendungen entwickeln, die auf die langfristige Vertraulichkeit der Daten angewiesen sind, die über eine TLS-Verbindung übertragen werden, sollten Sie einen Plan zur Umstellung auf Post-Quanten-Kryptografie in Betracht

ziehen, bevor große Quantencomputer zur Verfügung stehen. AWS arbeitet daran, sich auf diese Zukunft vorzubereiten, und wir möchten, dass auch Sie gut vorbereitet sind.



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

Diese Hybrid-Cipher-Suites können für Ihre Produktions-Workloads verwendet werden, wenn Sie aktuelle Versionen der SDKs verwenden. AWS Weitere Informationen zu enable/disable diesem Verhalten finden Sie unter Hybrid-Post-Quantum-TLS aktivieren.



Über den Hybrid-Post-Quantum-Schlüsselaustausch in TLS

Bei den verwendeten Algorithmen handelt es sich AWS um einen Hybrid, der Elliptic Curve Diffie-Hellman (ECDH), einen klassischen Algorithmus für den Schlüsselaustausch, der heute in TLS verwendet wird, mit Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) kombiniert, einem Algorithmus für Verschlüsselung und Schlüsselfestlegung mit öffentlichen Schlüsseln, den das National Institute for Standards and Technology (NIST) als seinen ersten Standardalgorithmus für die Nachquantenschlüsselvereinbarung bezeichnet hat. Dieser Hybrid verwendet jeden der Algorithmen unabhängig, um einen Schlüssel zu generieren. Dann kombiniert es die beiden Schlüssel kryptografisch.

Verwendung

Informationen zur Aktivierung von hybridem Post-Quantum-TLS in den SDKs finden Sie unter Hybrid-Post-Quantum-TLS aktivieren. AWS https://docs.aws.amazon.com/sdkref/latest/guide/pqtls-details.html

Erfahren Sie mehr über PQC

Informationen zum Post-Quanten-Kryptografie-Projekt am National Institute for Standards and Technology (NIST) finden Sie unter Kryptografie. Post-Quantum

Informationen zur NIST-Standardisierung nach der Quantenkryptografie finden Sie unter Kryptografie-Standardisierung. Post-Quantum

Aktivierung von hybridem Post-Quantum-TLS

Dieses Thema wurde in das Referenzhandbuch für AWS SDKs und Tools verschoben.

Bewährte Sicherheitsmethoden für AWS Kryptografie im Zahlungsverkehr

AWS Payment Cryptography unterstützt viele Sicherheitsfunktionen, die entweder integriert sind oder die Sie optional implementieren können, um den Schutz Ihrer Verschlüsselungsschlüssel zu verbessern und sicherzustellen, dass sie für den vorgesehenen Zweck verwendet werden. Dazu gehören IAM-Richtlinien, eine umfangreiche Reihe von Richtlinien-Bedingungsschlüsseln zur Verfeinerung Ihrer Schlüsselrichtlinien und IAM-Richtlinien sowie die integrierte Durchsetzung von PCI-PIN-Regeln in Bezug auf Schlüsselblöcke.

> **Important**
>
> Die bereitgestellten allgemeinen Richtlinien stellen keine vollständige Sicherheitslösung dar. Da nicht alle bewährten Methoden für alle Situationen geeignet sind, sollten diese nicht vorschriftig sein.

- Verwendung und Verwendungsarten von Schlüsseln: AWS Payment Cryptography folgt und setzt die in der ANSI X9 TR 31-2018 Interoperable Secure Key Exchange Key Block Specification beschriebenen Einschränkungen für die Verwendung und Nutzungsweise von Schlüsseln durch und entspricht den PCI-PIN-Sicherheitsanforderungen 18-3. Dadurch wird die Möglichkeit eingeschränkt, einen einzelnen Schlüssel für mehrere Zwecke zu verwenden, und die Schlüsselmetadaten (z. B. zulässige Operationen) werden kryptografisch an das Schlüsselmaterial selbst gebunden. AWS Payment Cryptography setzt diese Einschränkungen automatisch durch, sodass beispielsweise ein Schlüsselverschlüsselungsschlüssel (TR31_K0_KEY_ENCRYPTION_KEY) nicht auch für die Datenentschlüsselung verwendet werden kann. Weitere Details finden Sie unter Grundlegendes zu Schlüsselattributen für AWS Kryptografie-Schlüssel für Zahlungen.
- Beschränken Sie die gemeinsame Nutzung von symmetrischem Schlüsselmaterial: Teilen Sie nur symmetrisches Schlüsselmaterial (wie PIN-Verschlüsselungsschlüssel oder Schlüsselverschlüsselungsschlüssel) mit höchstens einer anderen Entität. Wenn vertrauliches Material an mehrere Entitäten oder Partner übertragen werden muss, erstellen Sie zusätzliche Schlüssel. AWS Bei der Zahlungskryptografie wird symmetrisches Schlüsselmaterial oder asymmetrisches privates Schlüsselmaterial niemals im Klartext offengelegt.
- Verwenden Sie Aliase oder Tags, um Schlüssel bestimmten Anwendungsfällen oder Partnern zuzuordnen: Aliase können verwendet werden, um den einem Schlüssel zugewiesenen Anwendungsfall einfach zu kennzeichnen, z. B. um einen Kartenprüfschlüssel alias/BIN_12345_CVK zu kennzeichnen, der mit BIN 12345 verknüpft ist. Um mehr Flexibilität zu bieten, sollten Sie in Betracht ziehen, Tags wie bin=12345, use_case=acquiring, country=us, partner=foo zu erstellen. Aliase und Tags können auch verwendet werden, um den Zugriff einzuschränken, z. B. um die Durchsetzung von Zugriffskontrollen zwischen der Ausstellung und dem Erwerb von Anwendungsfällen.
- Praktizieren Sie den Zugriff mit den geringsten Rechten: IAM kann verwendet werden, um den Produktionszugriff auf Systeme und nicht auf Einzelpersonen zu beschränken. So können beispielsweise einzelne Benutzer daran gehindert werden, Schlüssel zu erstellen oder kryptografische Operationen auszuführen. IAM kann auch verwendet werden, um den

Zugriff sowohl auf Befehle als auch auf Schlüssel einzuschränken, die für Ihren Anwendungsfall möglicherweise nicht geeignet sind, z. B. um die Möglichkeit einzuschränken, Pins für einen Acquirer zu generieren oder zu validieren. Eine weitere Möglichkeit, den Zugriff mit den geringsten Rechten zu nutzen, besteht darin, vertrauliche Operationen (wie den Schlüsselimport) auf bestimmte Dienstkonten zu beschränken. Beispiele finden Sie unter AWS Beispiele für identitätsbasierte Richtlinien zur Zahlungskryptografie.

Informationen finden Sie auch unter:

- Identitäts- und Zugriffsmanagement für AWS Kryptografie im Zahlungsverkehr
- Bewährte Methoden für die Sicherheit im IAM-Benutzerhandbuch.

Konformitätsprüfung für AWS Zahlungskryptografie

Wie bei anderen AWS Diensten benötigen Kunden ein klares Verständnis des Modells der gemeinsamen Verantwortung für Sicherheit und Compliance. Da es sich um einen Service handelt, der speziell Zahlungen unterstützt, ist es für Kunden von AWS Payment Cryptography besonders wichtig, die Einhaltung der geltenden PCI-Standards zu verstehen. AWS Zu den PCI DSS- und PCI 3DS-Assessments gehört auch AWS Zahlungskryptografie. In den Leitfäden zur gemeinsamen Verantwortung für diese Berichte, die unter verfügbar sind, können Hinweise auf den Service AWS Artifact enthalten sein. Die Bewertungen der PCI-PIN-Sicherheit und Point-to-Point Verschlüsselung (P2PE) beziehen sich speziell auf die AWS Zahlungskryptografie.

Dieser Abschnitt enthält Informationen zum Status und Umfang der Einhaltung der Vorschriften durch den Service sowie Informationen, die bei der Planung der PCI-PIN-Sicherheit und der PCI P2PE-Bewertung Ihrer Anwendungen hilfreich sein können.

Themen

- Konformität des Dienstes
- Planung der Einhaltung der PIN-Vorschriften
- Verwendung der AWS Payment Cryptography Decryption Component in P2PE-Lösungen

Konformität des Dienstes

Externe Prüfer bewerten die Sicherheit und Konformität der AWS Zahlungskryptografie im Rahmen mehrerer AWS Compliance-Programme. Dazu gehören SOC, PCI und andere.

AWS Die Zahlungskryptografie wurde zusätzlich zu PCI DSS und PCI 3DS nach mehreren PCI-Standards bewertet. Dazu gehören PCI-PIN-Sicherheit (PCI-PIN) und PCI-Verschlüsselung Point-to-Point (P2PE). Verfügbare Bescheinigungen und AWS Artifact Richtlinien zur Einhaltung der Vorschriften finden Sie unter.

Eine Liste der AWS Services im Rahmen bestimmter Compliance-Programme finden Sie unter AWS-Services in Umfang nach Compliance-Programm . Allgemeine Informationen finden Sie unter AWS - Compliance-Programme.

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Verwendung von AWS Payment Cryptography hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung von Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance Schnellstartanleitungen zu — In diesen Bereitstellungshandbüchern werden architektonische Überlegungen erörtert und Schritte für die Implementierung von sicherheits- und Compliance-orientierten Basisumgebungen beschrieben. AWS
- AWS Ressourcen zur Einhaltung von — Diese Sammlung von Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- Bewertung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — AWS Config; bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub CSPM—Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus, sodass Sie überprüfen können AWS , ob Sie die Sicherheitsstandards und Best Practices der Branche einhalten.

Planung der Einhaltung der PIN-Vorschriften

In diesem Leitfaden werden die Unterlagen und Nachweise beschrieben, die Sie benötigen, um sich auf eine PCI-PIN-Bewertung Ihrer PIN-Verarbeitungsanwendung vorzubereiten, die AWS Zahlungskryptografie verwendet.

Wie bei anderen Standards AWS-Services und Compliance-Standards liegt es in Ihrer Verantwortung, den Service sicher zu nutzen, die Zugriffskontrolle zu konfigurieren und Sicherheitsparameter gemäß den PCI-PIN-Anforderungen zu verwenden. In diesem Leitfaden werden diese Konfigurationen erörtert, sofern sie für die Erfüllung einer Anforderung angemessen sind.

Themen

- Allgemeine Themen
- Umfang der Bewertung
- Vorgänge zur Transaktionsverarbeitung

Allgemeine Themen

Die Migration von Anwendungen von einer Verbindung zu HSM zu einem verwalteten Dienst wie AWS Payment Cryptography wirft häufig auftretende Probleme und Konzepte für Kunden und deren Prüfer auf. Dieser Abschnitt enthält Informationen, um zu verdeutlichen, wie die sichere Nutzung des Dienstes diese Situationen behebt.

Themen

- Gemeinsame Verantwortlichkeit
- Minimale HSM-Konfiguration
- Schlüsselaustausch zwischen Kunde und APC

Gemeinsame Verantwortlichkeit

Kunden, die die vollständige Sicherheits- und Compliance-Verantwortung für Anwendungen übernommen haben, werden ihre Compliance-Anforderungen neu strukturieren, um die Vorteile der Schlüsselverwaltung, der Sicherheitskontrollen und der verwalteten HSM-Funktionen von AWS Payment Cryptography („der Service“) zu nutzen. Dadurch werden sich einige Anforderungen vollständig darauf verlagern AWS, wie die Bewertungen von AWS Payment Cryptography durch Dritte belegen. Einige Anforderungen werden zwischen der Anwendung des Kunden und dem Service gemeinsam gelten. Eine Anwendung ist verantwortlich für:

- Bereitstellung genauer Informationen für den Dienst
- Verwendung von Sicherheitskontrollen gemäß den Empfehlungen des Dienstes und den PCI-PIN-Sicherheitsanforderungen
- Implementierung der erforderlichen Sicherheitskontrollen mithilfe der vom Service bereitgestellten Tools

Kunden und ihre Prüfer verwenden Leitfäden zur gemeinsamen Verantwortung und zur Implementierung, die zusammen mit Konformitätsbescheinigungen veröffentlicht werden, AWS Artifact um die Kontrollen und die Kontrollüberwachung zu implementieren und anschließend die Bewertungen zu planen und abzuschließen.

Minimale HSM-Konfiguration

Der PCI-Datensicherheitsstandard, der grundlegende Standard für andere PCI-Standards, verlangt, dass alle Systeme mit den für ihre Funktion erforderlichen Mindestfunktionen konfiguriert werden. PCI

PIN, P2PE und andere Lösungsstandards wenden diese Anforderung HSMs in der Lösung an. HSMs darf nur Funktionen aktivieren, die für die Lösung benötigt werden.

AWS Dienste sollten wie Systeme behandelt und für die erforderliche Mindestfunktionalität konfiguriert werden. Payment Card Industry Data Security Standard (PCI DSS) v4.0 auf AWS empfiehlt die Verwendung von IAM, um die Mindestfunktionalität für jeden von der Lösung verwendeten AWS-Service zu konfigurieren. Dies gilt auch für Zahlungskryptografie. AWS IAM-Richtlinien ermöglichen detaillierte Berechtigungen, um kryptografische Funktionen nur auf die Anwendungskomponenten zu beschränken, die darauf angewiesen sind.

Schlüsselaustausch zwischen Kunde und APC

PIN Die Sicherheitsanforderungen 8-4 und 15-2 erfordern, dass öffentliche Schlüssel für den Austausch und das Laden von Schlüsseln authentifiziert und die Integrität geschützt sind. Für das Laden von POI-Schlüsseln per Fernzugriff, dessen Funktionsweise in ANSI/ASC X9 TR-34 beschrieben ist und durch die PCI-PIN Anhang A geregelt wird, werden öffentliche Schlüssel meistens in Zertifikaten übertragen, die von einer nach Anhang A2 konformen Zertifizierungsstelle signiert wurden. Für den Austausch zwischen Organisationen nutzen öffentliche Schlüssel andere Authentizitäts- und Integritätsmechanismen.

Alle Interaktionen zwischen dem Kunden und AWS erfolgen über AWS APIs, das jeden API-Aufruf gegenseitig authentifiziert und die Integrität von Aufrufen und Antworten mithilfe von TLS sicherstellt. Die Authentifizierung der Kundenanwendung wird von AWS Identity and Access Management mit Mechanismen wie Sicherheitstoken und SigV4 verwaltet. AWS-API-Endpunkte werden vom Kunden mithilfe der TLS-Serverauthentifizierung authentifiziert, die in AWS integriert ist. SDKs Dann gewährleistet TLS die Vertraulichkeit und Integrität aller Daten, die zwischen dem Kunden und jeder AWS-API übertragen werden.

APC APIs GetParametersForImport und ImportKey implementieren Sie eine Schlüsselübertragung vom Kunden zum Service. Die von GetParametersForImport bereitgestellte Zertifizierungsstelle (CA) entspricht zwar nicht Anhang A2, ist aber sicher und nur für das Konto bestimmt. Zwar kann man sich nicht darauf verlassen, dass diese Zertifizierungsstelle die Anforderungen 8-4 und 15-2 erfüllt, sie bietet jedoch eine Integritätsprüfung des importierten Schlüssels. Sie können auch Ihre eigene CA verwenden, indem Sie die API nutzen. GetCertificateSigningRequest

Die Mechanismen, die die Authentifizierung mit öffentlichen Schlüsseln und die Integritätssicherung ermöglichen, sind:

- Authentifizierung durch AWS-API-Authentifizierung

- Die Integrität des Schlüssels wird durch die MAC-Funktion des von bereitgestellten Zertifikats gewährleistet GetParametersForImport, auch wenn die Identitätsinformationen im Zertifikat nicht vertrauenswürdig sind. Die Integrität des Schlüssels wird auch durch den von TLS verwendeten MAC gewährleistet, der die Sitzung zwischen dem Kunden und AWS schützt.

Die von APC bereitgestellten Zertifikate und Schlüsselblöcke entsprechen Anhang A1, der die Anforderungen an Zertifikate und den Schutz von Schlüsseln durch asymmetrische Methoden festlegt.

Umfang der Bewertung

Der erste Schritt bei der Planung einer Bewertung ist die Dokumentation des Umfangs. Bei PCI-PIN geht es um Systeme und Prozesse zum Schutz von PINs, einschließlich des Schutzes der kryptografischen Schlüssel und der Geräte, die sie schützen, also Zahlungsterminals, auch Points of Interaction (POI) genannt, HSMs und andere sichere kryptografische Geräte (SCD).

Wir werden uns nicht mit Anforderungen befassen, für die Sie die volle Verantwortung tragen, da diese Bereiche betreffen, die nicht zum Leistungsumfang gehören. Zum Beispiel die Konfiguration und Bereitstellung von Zahlungsterminals. Informationen zur PCI-PIN finden Sie im AWS Payment Cryptography Shared Responsibility Guide, verfügbar unter AWS Artifact

Themen

- Gemeinsame Verantwortlichkeit
- High-Level Netzwerkdiagramme
- Schlüsseltabelle
- Verweise auf Dokumente

Gemeinsame Verantwortlichkeit

AWS Payment Cryptography ist eine Encryption and Support Organization (ESO) und ein PIN-Acquiring Third-Party Servicer (TPS), wie im Visa PIN Security Program definiert und im Visa Global Service Provider Registry unter „Amazon Web Services, LLC“ aufgeführt. Dies bedeutet, dass Visa die Nutzung des Dienstes durch den PIN-Acquiring Third-Party VisaNet Prozessor (VNP), den als Dienstleister agierenden PIN-Acquiring VisaNet Kundenprozessor und andere TPS- und ESO-Anbieter gestattet, ohne dass eine weitere Prüfung durch Kunden-PIN-Assessoren (PCI Qualified PIN Assessors oder PCI QPA) erforderlich ist.

Andere Kartenmarken oder Anbieter von Zahlungsnetzwerken verlassen sich möglicherweise auf das Visa PIN Security Program oder haben ihre eigenen Programme. Wenden Sie sich an, wenn Sie Fragen AWS Support zur Einhaltung der Datenschutzbestimmungen für andere Zahlungsnetzwerkprogramme haben.

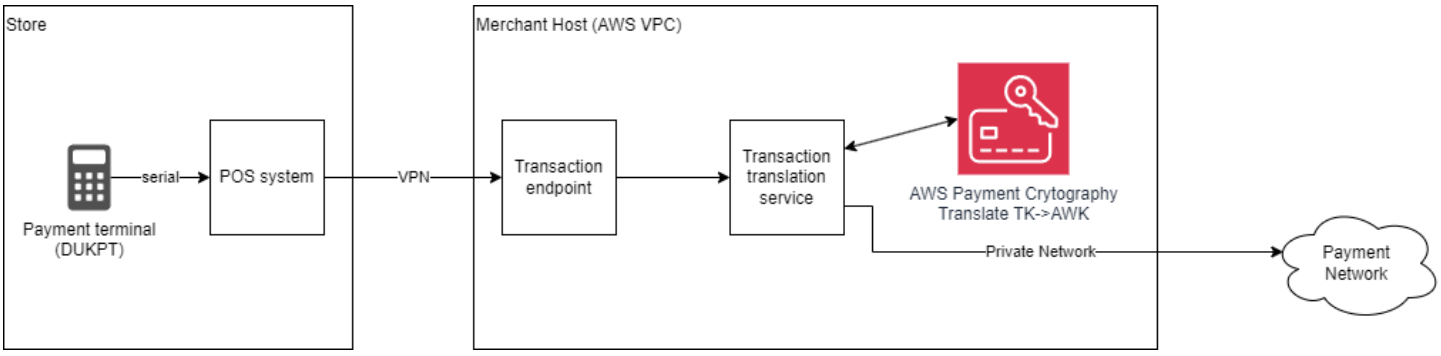
AWS bietet die PCI-PIN-Konformitätsbescheinigung (AOC) und den Leitfaden zur gemeinsamen Verantwortung für AWS Zahlungskryptografie in. AWS Artifact Der Einsatz von Dienstanbietern bei der PIN-Verarbeitung ist seit vielen Jahren üblich, der PCI-PIN-Sicherheitsstandard (bis Version 3.1) bezieht sich jedoch nicht auf die Verwaltung durch Drittanbieter. Das Visa-PIN-Sicherheitsprogramm auch nicht. Der Kunde QPA hat sich an das mit dem PCI DSS AOC und dem Shared Responsibility Guide etablierte Modell gehalten, wonach die Einhaltung der geltenden Anforderungen als „erfolgreich AWS“ bezeichnet wird.

High-Level Netzwerkdiagramme

Die PCI-PIN-Berichtsvorlage erfordert: „Stellen Sie für Unternehmen, die an der Verarbeitung von PIN-basierten Transaktionen beteiligt sind, ein Netzwerkschema zur Verfügung, in dem die PIN-basierten Transaktionsabläufe mit der zugehörigen Schlüsseltypverwendung beschrieben werden. Darüber hinaus sollten KIF und Unternehmen, die Schlüssel mithilfe asymmetrischer Techniken aus der Ferne verteilen, wichtige Materialflüsse bereitstellen.“

AWS Payment Cryptography hat die interne Servicestruktur für unsere PCI-PIN-Bewertung gemeldet. Ihre Diagramme veranschaulichen den Aufruf der Service-APIs für die PIN-Verarbeitung.

Beispiel für ein übergeordnetes Netzwerkdiagramm für PIN-Anwendungen, die AWS Zahlungskryptografie verwenden:



Schlüsseltabelle

In dem Bericht müssen alle Schlüssel, die PINs direkt oder indirekt schützen, aufgeführt werden. Alle Schlüssel, die im Dienst vorhanden sind, können in der ListKeysAPI aufgeführt werden.

Stellen Sie sicher, dass Sie die Schlüsselliste für alle Regionen und Konten angeben, die Schlüssel für Ihre Anwendung besitzen.

Verweise auf Dokumente

Die Herstellerdokumentation und die Empfehlungen für den sicheren Einsatz von AWS Payment Cryptography finden Sie im Benutzerhandbuch und in der API-Referenz. Diese sind gegebenenfalls in dieser Anleitung verlinkt.

Vorgänge zur Transaktionsverarbeitung

Die PCI-PIN-Anforderungen sind in Kontrollzielen organisiert. In jedem Kontrollziel sind die Anforderungen zur Sicherung eines bestimmten Sicherheitsaspekts für PINs zusammengefasst.

Themen

- Kontrollziel 1: PINs, die bei Transaktionen verwendet werden, die diesen Anforderungen unterliegen, werden mit Geräten und Methoden verarbeitet, die ihre Sicherheit gewährleisten.
- Kontrollziel 2: Kryptografische Schlüssel, die für die PIN encryption/decryption und die zugehörige Schlüsselverwaltung verwendet werden, werden mithilfe von Verfahren erstellt, die sicherstellen, dass es nicht möglich ist, einen Schlüssel vorherzusagen oder festzustellen, dass bestimmte Schlüssel wahrscheinlicher sind als andere Schlüssel.
- Kontrollziel 3: Schlüssel werden auf sichere Weise übermittelt oder übertragen.
- Kontrollziel 4: Der Umgang Key-loading mit HSMs und PIN-acceptance POI-Geräten erfolgt auf sichere Weise.
- Kontrollziel 5: Schlüssel werden so verwendet, dass ihre unbefugte Verwendung verhindert oder erkannt wird.
- Kontrollziel 6: Schlüssel werden auf sichere Weise verwaltet.
- Kontrollziel 7: Geräte, die zur Verarbeitung von PINs und Schlüsseln verwendet werden, werden auf sichere Weise verwaltet.

Kontrollziel 1: PINs, die bei Transaktionen verwendet werden, die diesen Anforderungen unterliegen, werden mit Geräten und Methoden verarbeitet, die ihre Sicherheit gewährleisten.

Anforderung 1: HSMs, die von AWS Payment Cryptography verwendet werden, wurden im Rahmen unserer PCI-PIN-Bewertung bewertet. Für Kunden, die den Service nutzen, gelten die Anforderungen

1—3 und 1—4 in Bezug auf das vom Service verwaltete HSM als „An Ort und Stelle“. Aus den Ergebnissen für HSM geht hervor, dass die Tests von der QPA bestätigt wurden. AWS Die PIN-Konformitätsbescheinigung kann abgerufen werden. AWS Artifact Andere SCDs, wie POI, in Ihrer Lösung müssen inventarisiert und referenziert werden.

Anforderung 2: In der Dokumentation Ihrer Verfahren muss angegeben werden, wie die PINs von Karteninhabern geschützt werden, was die Weitergabe an Ihr Personal angeht, welche PIN-Übersetzungsprotokolle implementiert wurden und wie sie bei der Online- und Offline-Verarbeitung geschützt werden. Darüber hinaus sollte Ihre Dokumentation eine Zusammenfassung der in jeder Zone verwendeten Methoden zur Verwaltung kryptografischer Schlüssel enthalten.

Anforderung 3: Der POI muss für eine sichere PIN-Verschlüsselung und -Übertragung konfiguriert sein. AWS Die Zahlungskryptografie unterstützt nur PIN-Block-Übersetzungen, die in Anforderung 3-3 angegeben sind.

Anforderung 4: Die Anwendung darf keine PIN-Blöcke speichern. Die PIN-Blöcke, auch wenn sie verschlüsselt sind, dürfen nicht in Transaktionsjournalen oder -protokollen aufbewahrt werden. Der Dienst speichert keine PIN-Blöcke und die PIN-Bewertung stellt sicher, dass sie nicht in den Protokollen enthalten sind.

Beachten Sie, dass der PCI-PIN-Sicherheitsstandard für die Erfassung „der sicheren Verwaltung, Verarbeitung und Übertragung von persönlichen Identifikationsnummern (PIN) bei der Verarbeitung von Online- und Offline-Zahlungskartentransaktionen an Geldautomaten und POS-Terminals“ gilt, wie in der Norm angegeben. Der Standard wird jedoch häufig zur Bewertung der Verwaltung kryptografischer Schlüssel für Zahlungen außerhalb des vorgesehenen Bereichs verwendet. Dies kann Anwendungsfälle von Emittenten einschließen, in denen PINs gespeichert werden. Ausnahmen von den Anforderungen für diese Fälle sollten mit der für die Bewertung vorgesehenen Zielgruppe vereinbart werden.

Kontrollziel 2: Kryptografische Schlüssel, die für die PIN encryption/decryption und die zugehörige Schlüsselverwaltung verwendet werden, werden mithilfe von Verfahren erstellt, die sicherstellen, dass es nicht möglich ist, einen Schlüssel vorherzusagen oder festzustellen, dass bestimmte Schlüssel wahrscheinlicher sind als andere Schlüssel.

Anforderung 5: Die Schlüsselgenerierung durch AWS Zahlungskryptografie wurde im Rahmen unserer PCI-PIN-Bewertung bewertet. Dies kann in der Schlüsseltabelle in der Spalte „Generiert von“ angegeben werden.

Anforderung 6: Die Sicherheitskontrollen für Schlüssel, die in AWS Zahlungskryptografie gespeichert sind, wurden im Rahmen der PCI-PIN-Bewertung des Dienstes bewertet. Fügen Sie Beschreibungen der Sicherheitskontrollen bei, die sich auf die Schlüsselgenerierung in Ihrer Anwendung und bei allen anderen Dienstanbietern beziehen.

Anforderung 7: Sie benötigen eine Richtliniendokumentation zur Schlüsselgenerierung, in der angegeben ist, wie Schlüssel generiert werden, und alle betroffenen Parteien müssen sich dieser Tatsache bewusst sein. procedures/policies Die Verfahren für die Schlüsselerstellung mithilfe der APC-API sollten die Verwendung von Rollen mit Berechtigungen zur Schlüsselerstellung und Genehmigungen für die Ausführung von Skripten oder anderem Code zur Schlüsselerstellung beinhalten. AWS CloudTrail Protokolle enthalten alle CreateKeyEreignisse mit Datum und Uhrzeit, Schlüssel-ARN und Benutzer-IDs. HSM-Seriennummern und Protokolle für den Zugriff auf physische Medien wurden im Rahmen der PIN-Bewertung des Dienstes bewertet.

Kontrollziel 3: Schlüssel werden auf sichere Weise übermittelt oder übertragen.

Anforderung 8: Die Schlüsselübertragung mit AWS Zahlungskryptografie wurde im Rahmen unserer PCI-PIN-Bewertung bewertet. Sie müssen die wichtigsten Schutzmechanismen für Übertragungen vor dem Import in und nach dem Export aus AWS Payment Cryptography dokumentieren. Der Service stellt wichtige Prüfwerte für alle Schlüssel bereit, um die korrekte Übertragung zu überprüfen.

Anforderung 8-4 verlangt, dass öffentliche Schlüssel auf eine Weise übertragen werden, die ihre Integrität und Authentizität schützt. Die Übertragung zwischen Ihrer Anwendung und AWS wird durch die Authentifizierung der Anwendung zur AWS API-Endpunktauthentifizierung der Anwendung mithilfe von AWS Identity and Access Management Methoden über TLS-Serverzertifikate gesteuert. AWS Darüber hinaus verfügen öffentliche Schlüssel, die aus Payment Cryptography exportiert oder nach AWS Payment Cryptography importiert wurden, über Zertifikate, die von kurzlebigen, kundenspezifischen Zertifizierungsstellen signiert sind (siehe, und). GetPublicKeyCertificateGetParametersForImportGetParametersForExport Diese Zertifizierungsstellen können nicht als alleinige Authentifizierungsmethode verwendet werden, da sie nicht dem PCI-PIN-Sicherheitsanhang A2 entsprechen. Die Zertifikate bieten jedoch weiterhin eine Integritätsgarantie für öffentliche Schlüssel, wobei IAM die Authentifizierung ermöglicht.

Wenn Sie öffentliche Schlüssel mit Ihren Geschäftspartnern mithilfe asymmetrischer Methoden austauschen, müssen Sie für die Authentifizierung des Unternehmens über den Kommunikationskanal sorgen, beispielsweise über eine sichere Website für den Dateiaustausch.

Anforderung 9: Der Dienst verwendet keine Schlüsselkomponenten im Klartext-Format und unterstützt diese auch nicht direkt.

Anforderung 10: Der Service setzt die relative Stärke des Schutzes von Schlüsseln für die Übertragung von Schlüsseln durch. Sie sind für die Schlüsselübertragung vor dem Import und nach dem Export aus AWS Payment Cryptography verantwortlich und verwenden API und TR-31 Parameter, die für den Import, Export und Generierung von Schlüsseln korrekt sind. Sie sollten über dokumentierte Verfahren zur Beschreibung der wichtigsten Übertragungsmechanismen und der Liste der für die Übertragung verwendeten kryptografischen Schlüssel verfügen.

Anforderung 11: In der Dokumentation Ihrer Verfahren muss angegeben sein, wie die Schlüssel übertragen werden. Die Verfahren für die Schlüsselübertragung mithilfe der AWS Payment Cryptography API sollten die Verwendung von Rollen mit Schlüsselimport- und -exportberechtigungen sowie Genehmigungen für die Ausführung von Skripten oder anderem Code zur Schlüsselerstellung beinhalten. AWS CloudTrail Protokolle enthalten alle Ereignisse und Ereignisse. ImportKeyExportKey

Kontrollziel 4: Der Umgang Key-loading mit HSMs und PIN-acceptance POI-Geräten erfolgt auf sichere Weise.

Anforderung 12: Sie sind dafür verantwortlich, Schlüssel aus Komponenten oder Shares zu laden. Die Verwaltung der HSM-Hauptschlüssel wurde im Rahmen der PIN-Bewertung des Dienstes bewertet. AWS Bei der Zahlungskryptografie werden keine Schlüssel aus einzelnen Aktien oder Komponenten geladen. Siehe Abschnitt Kryptografische Details.

Anforderungen 13 und 14: Sie müssen den Schlüsselschutz für Übertragungen vor dem Import in den Service und nach dem Export aus dem Service beschreiben.

Anforderung 15: AWS Die Zahlungskryptografie bietet wichtige Prüfwerte für alle Schlüssel im Service und gewährleistet die Integrität öffentlicher Schlüssel. Ihre Anwendung ist dafür verantwortlich, diese Prüfungen zur Validierung von Schlüsseln nach dem Import oder Export aus dem Service zu verwenden. Sie sollten die Verfahren dokumentieren, um sicherzustellen, dass ein Validierungsmechanismus vorhanden ist.

Anforderung 15-2 erfordert, dass öffentliche Schlüssel auf eine Weise geladen werden, die ihre Integrität und Authentizität schützt. ImportKeysorgt zusammen mit für GetParametersForImportdie Validierung der bereitgestellten Signaturzertifikate. Wenn die bereitgestellten Zertifikate selbstsigniert sind, muss die Authentifizierung durch einen separaten Mechanismus erfolgen, z. B. durch sicheren Dateiaustausch.

Anforderung 16: In der Dokumentation Ihrer Verfahren muss angegeben werden, wie Schlüssel in den Dienst geladen werden. Die Verfahren für den Schlüsselimport mithilfe der API sollten

die Verwendung von Rollen mit Schlüsselimportberechtigungen und Genehmigungen für die Ausführung von Skripts oder anderem Code zum Laden von Schlüsseln beinhalten. AWS CloudTrail Protokolle enthalten alle ImportKeyEreignisse. Sie sollten die Protokollierungsmechanismen in die Dokumentation aufnehmen. Der Dienst stellt wichtige Prüfwerte für alle Schlüssel bereit, um das korrekte Laden der Schlüssel zu überprüfen.

Kontrollziel 5: Schlüssel werden so verwendet, dass ihre unbefugte Verwendung verhindert oder erkannt wird.

Anforderung 17: Der Dienst stellt Mechanismen wie Tags und Aliase für Schlüssel bereit, mit denen die Beziehungen zwischen Schlüsseln nachverfolgt werden können. Darüber hinaus sollten Schlüsselprüfwerte getrennt aufbewahrt werden, um nachzuweisen, dass bekannte oder standardmäßige Schlüsselwerte nicht verwendet werden, wenn Schlüssel gemeinsam genutzt werden.

Anforderung 18: Der Dienst bietet Schlüsselintegritätsprüfungen über GetKeyund und ListKeysSchlüsselverwaltungsereignisse über AWS CloudTrail, mit deren Hilfe unbefugte Ersetzungen erkannt oder die Synchronisation von Schlüsseln zwischen den Parteien überwacht werden können. Der Dienst speichert Schlüssel ausschließlich in Schlüsselblöcken. Sie sind für die Speicherung und Verwendung der Schlüssel vor dem Import und nach dem Export aus AWS Payment Cryptography verantwortlich.

Sie sollten über Verfahren verfügen, die eine sofortige Untersuchung ermöglichen, falls es bei der Verarbeitung von PIN-basierten Transaktionen oder bei unerwarteten Schlüsselverwaltungsereignissen zu Unstimmigkeiten kommen sollte.

Anforderung 19: Der Dienst verwendet Schlüssel ausschließlich in Schlüsselblöcken KeyUsage, KeyModeOfUse Erzwingungs- und anderen Schlüsselattributen für alle Operationen. Dazu gehört auch die Einschränkung von Operationen mit privaten Schlüsseln. Sie sollten Ihre öffentlichen Schlüssel für einen einzigen Zweck verwenden, z. B. für Verschlüsselung oder Überprüfung digitaler Signaturen, aber nicht für beides. Sie sollten separate Konten für Produktion und test/development Systeme verwenden.

Anforderung 20: Sie behalten die Verantwortung für diese Anforderung.

Kontrollziel 6: Schlüssel werden auf sichere Weise verwaltet.

Anforderung 21: Die Speicherung und Verwendung von Schlüsseln mit AWS Zahlungskryptografie wurde im Rahmen der PCI-PIN-Bewertung des Dienstes bewertet. Für Speicheranforderungen im

Zusammenhang mit wichtigen Komponenten sind Sie dafür verantwortlich, diese wie unter 21-2 und 21-3 beschrieben aufzubewahren. Vor dem Import in den Service und nach dem Export aus dem Service müssen Sie die wichtigsten Schutzmechanismen in Ihrer Richtliniendokumentation beschreiben.

Anforderung 22: Die wichtigsten Kompromissverfahren für AWS Zahlungskryptografie wurden im Rahmen der PCI-PIN-Bewertung des Dienstes bewertet. Sie müssen die wichtigsten Verfahren zur Erkennung und Reaktion auf Sicherheitslücken beschreiben, einschließlich der Überwachung und Reaktion auf die Meldung von AWS.

Anforderung 23: AWS Zahlungskryptografie unterstützt keine Varianten oder andere Methoden zur Berechnung reversibler Schlüssel. APC-Hauptschlüssel oder damit verschlüsselte Schlüssel sind für Kunden niemals verfügbar. Die Verwendung der reversiblen Schlüsselberechnung wurde im Rahmen der PCI-PIN-Bewertung des Dienstes bewertet.

Anforderung 24: Verfahren zur Zerstörung interner geheimer und privater Schlüssel Die AWS Zahlungskryptografie wurde im Rahmen der PCI-PIN-Bewertung des Dienstes bewertet. Sie müssen das Verfahren zur Schlüsselvernichtung von Schlüsseln vor dem Import in und nach dem Export aus APC beschreiben. Die wichtigsten Anforderungen an die Vernichtung von Komponenten (24-2.2 und 24-2.3) liegen weiterhin in Ihrer Verantwortung.

Anforderung 25: Der Zugriff auf geheime und private Schlüssel innerhalb der AWS Zahlungskryptografie wurde im Rahmen der PCI-PIN-Bewertung des Dienstes bewertet. Sie benötigen ein Verfahren und eine Dokumentation für die Zugriffskontrollen für Schlüssel vor dem Import in und nach dem Export aus AWS Payment Cryptography.

Anforderung 26: Sie müssen die Protokollierung für jeden Zugriff auf Schlüssel, Schlüsselkomponenten oder zugehörige Materialien beschreiben, die außerhalb des Dienstes verwendet werden. Protokolle für alle wichtigen Verwaltungsaktivitäten, die Ihre Anwendung mit dem Service durchführt, sind unter verfügbar AWS CloudTrail.

Anforderung 27: Sie müssen die Sicherungsverfahren für Schlüssel, Schlüsselkomponenten oder zugehörige Materialien beschreiben, die außerhalb des Dienstes verwendet werden.

Anforderung 28: Die Verfahren für die gesamte Schlüsselverwaltung, die die API verwenden, sollten die Verwendung von Rollen mit Schlüsselverwaltungsberechtigungen und Genehmigungen für die Ausführung von Skripten oder anderem Code zur Schlüsselverwaltung beinhalten. AWS CloudTrail Protokolle enthalten alle wichtigen Verwaltungsereignisse

Kontrollziel 7: Geräte, die zur Verarbeitung von PINs und Schlüsseln verwendet werden, werden auf sichere Weise verwaltet.

Anforderung 29: Ihre Anforderungen an den physischen und logischen Schutz von HSMs werden durch den Einsatz von AWS Zahlungskryptografie erfüllt.

Anforderung 30: Ihre Anwendung behält die Verantwortung für alle Anforderungen an den physischen und logischen Schutz von POI-Geräten.

Anforderung 31: Der Schutz sicherer kryptografischer Geräte (SCD), die von AWS Payment Cryptography verwendet werden, wurde im Rahmen der PCI-PIN-Bewertung des Dienstes bewertet. Sie müssen den Schutz aller anderen von Ihrer Anwendung verwendeten SCDs nachweisen.

Anforderung 32: Die Verwendung von SCDs, die von AWS Payment Cryptography verwendet werden, wurde im Rahmen der PCI-PIN-Bewertung des Dienstes bewertet. Sie müssen die Zugriffskontrolle und den Schutz aller anderen von Ihrer Anwendung verwendeten SCDs nachweisen.

Anforderung 33: Sie müssen den Schutz aller PIN-processing Geräte beschreiben, die sich unter Ihrer Kontrolle befinden.

Verwendung der AWS Payment Cryptography Decryption Component in P2PE-Lösungen

PCI P2PE-Lösungen können die Komponente zur Entschlüsselung der AWS Zahlungskryptografie verwenden. Dies ist im Abschnitt Point-to-Point PCI-Verschlüsselung: Sicherheitsanforderungen und Testverfahren, Abschnitt P2PE-Lösungen und Einsatz von and/or P2PE-Komponentenanbietern von Drittanbietern dokumentiert: „Ein Lösungsanbieter (oder ein Händler als Lösungsanbieter) kann bestimmte P2PE-Funktionen an PCI-gelistete P2PE-Komponentenanbieter auslagern und die Verwendung der PCI-gelisteten P2PE-Komponente (n) in seinem P2PE-Validierungsbericht (P-ROV) melden“, der auf der PCI-Website verfügbar ist.

Wie bei anderen AWS-Services und Compliance-Standards liegt es in Ihrer Verantwortung, den Service sicher zu nutzen, die Zugriffskontrolle zu konfigurieren und Sicherheitsparameter gemäß den PCI P2PE-Anforderungen zu verwenden. Das Benutzerhandbuch für die P2PE-Entschlüsselungskomponente von AWS Payment Cryptography, das unter verfügbar ist AWS Artifact, enthält detaillierte Anweisungen zur Integration von AWS Payment Cryptography in Ihre PCI-P2PE-Lösung sowie den jährlichen Bericht über die Entschlüsselungskomponenten, der für die Compliance-Berichterstattung erforderlich ist.

Identitäts- und Zugriffsmanagement für AWS Kryptografie im Zahlungsverkehr

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu kontrollieren. AWS IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um AWS Zahlungskryptografie-Ressourcen zu verwenden. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- Wie AWS Zahlungskryptografie funktioniert mit IAM
- AWS Beispiele für identitätsbasierte Richtlinien zur Zahlungskryptografie
- Resource-based Richtlinien für AWS Kryptografie im Zahlungsverkehr
- Multi-party Zulassung für AWS Zahlungskryptografie
- Fehlerbehebung AWS Zahlungskryptografie, Identität und Zugriff

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Fehlerbehebung AWS Zahlungskryptografie, Identität und Zugriff).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe Wie AWS Zahlungskryptografie funktioniert mit IAM).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe AWS Beispiele für identitätsbasierte Richtlinien zur Zahlungskryptografie).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer für den Zugriff AWS mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer-

zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identity-based Richtlinien

Identity-based Richtlinien sind Richtliniendokumente für JSON-Berechtigungen, die Sie an eine Identität (Benutzer, Gruppe oder Rolle) anhängen. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identity-based Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mehreren Identitäten zugeordnet sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Resource-based Richtlinien

Resource-based Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anhängen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Resource-based Richtlinien sind Inline-Richtlinien, die sich in diesem Dienst befinden. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs legen die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in AWS Organizations fest. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Ressourcen-Kontrollrichtlinien (RCPs) – RCPs definieren die maximale Anzahl an Berechtigungen, die Ressourcen in Ihren Konten zur Verfügung stehen. Weitere Informationen finden Sie unter Ressourcen-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere Arten von Richtlinien für eine Anfrage gelten, sind die sich daraus ergebenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

Wie AWS Zahlungskryptografie funktioniert mit IAM

Bevor Sie IAM verwenden, um den Zugriff auf AWS Payment Cryptography zu verwalten, sollten Sie wissen, welche IAM-Funktionen für die Verwendung mit Payment Cryptography verfügbar sind. AWS Einen allgemeinen Überblick darüber, wie AWS Zahlungskryptografie und andere AWS Dienste mit IAM funktionieren, finden Sie im IAM-Benutzerhandbuch unter AWS Dienste, die mit IAM funktionieren.

Themen

- AWS Richtlinien für Zahlungskryptografie Identity-based
- Autorisierung basiert auf AWS Kryptografie-Tags für Zahlungen

AWS Richtlinien für Zahlungskryptografie Identity-based

Mit identitätsbasierten IAM-Richtlinien können Sie zulässige oder verweigerte Aktionen und Ressourcen sowie die Bedingungen angeben, unter denen Aktionen zulässig oder verweigert werden. AWS Die Zahlungskryptografie unterstützt bestimmte Aktionen, Ressourcen und Bedingungsschlüssel. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Aktionen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu erteilen.

Richtlinienaktionen in der AWS Zahlungskryptografie verwenden vor der Aktion das folgende Präfix:`payment-cryptography:`. Um beispielsweise jemandem die Erlaubnis zu erteilen, einen AWS Payment Cryptography `VerifyCardData` API-Vorgang auszuführen, nehmen Sie die `payment-cryptography:VerifyCardData` Aktion in seine Richtlinie auf. Richtlinienanweisungen müssen entweder ein – `Action`oder ein `NotAction`-Element enthalten. AWS Payment Cryptography definiert eigene Aktionen, die Aufgaben beschreiben, die Sie mit diesem Dienst ausführen können.

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie wie folgt durch Kommata:

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

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Um beispielsweise alle Aktionen anzugeben, die mit dem Wort beginnen `List` (wie `ListKeys` und`ListAliases`), schließen Sie die folgende Aktion ein:

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

Eine Liste der AWS Zahlungskryptografie-Aktionen finden Sie im IAM-Benutzerhandbuch unter Durch AWS Zahlungskryptografie definierte Aktionen.

Ressourcen

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

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

Die Schlüsselressource für Zahlungskryptografie hat den folgenden ARN:

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

Weitere Informationen zum Format von ARNs finden Sie unter Amazon Resource Names (ARNs) und AWS Service Namespaces.

Wenn Sie beispielsweise die `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`-Instance in Ihrer Anweisung angeben möchten, verwenden Sie den folgenden ARN:

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

Um alle Schlüssel anzugeben, die zu einem bestimmten Konto gehören, verwenden Sie den Platzhalter (*):

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

Einige Kryptografie-Aktionen für AWS Zahlungen, z. B. zum Erstellen von Schlüsseln, können nicht für eine bestimmte Ressource ausgeführt werden. In diesen Fällen müssen Sie den Platzhalter (*) verwenden.

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

Um mehrere Ressourcen in einer einzigen Anweisung anzugeben, verwenden Sie ein Komma wie unten dargestellt:

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

Beispiele

Beispiele für identitätsbasierte AWS Zahlungskryptografie-Richtlinien finden Sie unter. AWS Beispiele für identitätsbasierte Richtlinien zur Zahlungskryptografie

Autorisierung basiert auf AWS Kryptografie-Tags für Zahlungen

Sie können Tags an AWS Zahlungskryptografie-Ressourcen anhängen oder Tags in einer Anfrage an AWS Zahlungskryptografie weitergeben. Um den Zugriff auf der Grundlage von Tags zu steuern,

geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `payment-cryptography:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

AWS Beispiele für identitätsbasierte Richtlinien zur Zahlungskryptografie

Standardmäßig sind IAM-Benutzer und -Rollen nicht berechtigt, AWS Payment Cryptography-Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mit der AWS-Managementkonsole AWS CLI, oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern und Rollen die Berechtigung zum Ausführen bestimmter API-Operationen für die angegebenen Ressourcen gewähren, die diese benötigen. Der Administrator muss diese Richtlinien anschließend den IAM-Benutzern oder -Gruppen anfügen, die diese Berechtigungen benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von Richtlinien auf der JSON-Registerkarte im IAM-Benutzerhandbuch.

Themen

- Best Practices für Richtlinien
- Verwendung der AWS Konsole für Zahlungen, Kryptografie
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer
- Möglichkeit, auf alle Aspekte von zuzugreifen AWS Kryptografie für Zahlungen
- Möglichkeit, APIs mit bestimmten Schlüsseln aufzurufen
- Fähigkeit, eine Ressource gezielt abzulehnen

Best Practices für Richtlinien

Identity-based Richtlinien legen fest, ob jemand AWS Zahlungskryptografie-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Wenn Sie identitätsbasierte Richtlinien erstellen oder bearbeiten, befolgen Sie diese Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für

viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.

- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwendung der AWS Konsole für Zahlungen, Kryptografie

Um auf die AWS Payment Cryptography Console zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, die AWS Zahlungskryptografie-Ressourcen in Ihrem AWS Konto aufzulisten und einzusehen. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (IAM-Benutzer oder -Rollen) mit dieser Richtlinie.

Um sicherzustellen, dass diese Entitäten weiterhin die AWS Payment Cryptography Console verwenden können, fügen Sie den Entitäten außerdem die folgende AWS verwaltete Richtlinie bei. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die den API-Operation entsprechen, die Sie ausführen möchten.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder AWS .

```
{
    "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": "*"
        }
    ]
}
```

Möglichkeit, auf alle Aspekte von zuzugreifen AWS Kryptografie für Zahlungen

> **Warning**
>
> Dieses Beispiel bietet umfangreiche Berechtigungen und wird nicht empfohlen. Ziehen Sie stattdessen Modelle mit den geringsten Zugriffsrechten in Betracht.

In diesem Beispiel möchten Sie einem IAM-Benutzer in Ihrem AWS Konto Zugriff auf alle Ihre AWS Payment Cryptography Keys gewähren und die Möglichkeit geben, alle Payment Cryptography APIs aufzurufen, AWS einschließlich der beiden Operationen. `ControlPlane` `DataPlane`

JSON

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

```
                "payment-cryptography:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Möglichkeit, APIs mit bestimmten Schlüsseln aufzurufen

In diesem Beispiel möchten Sie einem IAM-Benutzer in Ihrem AWS Konto Zugriff auf einen Ihrer AWS Payment Cryptography Keys gewähren `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h` und diese Ressource dann in zwei APIs verwenden, `GenerateCardValidationData` und. `VerifyCardValidationData` Umgekehrt hat der IAM-Benutzer keinen Zugriff darauf, diesen Schlüssel für andere Operationen wie oder zu verwenden `DeleteKey` `ExportKey`

Bei Ressourcen kann es sich entweder um Schlüssel mit Präfix `key` oder um Aliase mit Präfix handeln. `alias`

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"
            ]
        }
    ]
}
```

Fähigkeit, eine Ressource gezielt abzulehnen

> **Warning**
>
> Überlegen Sie sorgfältig, welche Auswirkungen die Gewährung von Wildcard-Zugriff hat. Ziehen Sie stattdessen ein Modell mit den geringsten Rechten in Betracht.

In diesem Beispiel möchten Sie einem IAM-Benutzer in Ihrem AWS Konto Zugriff auf einen beliebigen Ihrer AWS Payment Cryptography-Schlüssel gewähren, aber Sie möchten die Berechtigungen für einen bestimmten Schlüssel verweigern. Der Benutzer hat Zugriff auf `VerifyCardData` und `GenerateCardData` mit allen Schlüsseln, mit Ausnahme des Schlüssels, der in der Ablehnungsanweisung angegeben ist.

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"
              ]
```

```
            }
        ]
    }
```

Resource-based Richtlinien für AWS Kryptografie im Zahlungsverkehr

Resource-based Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anhängen, z. B. einen AWS Payment Cryptography Key. In einer ressourcenbasierten Richtlinie geben Sie an, wer auf den Schlüssel zugreifen kann und welche Aktionen mit ihm ausgeführt werden können. Sie können ressourcenbasierte Richtlinien für Folgendes verwenden:

- Gewähren Sie mehreren Benutzern und Rollen Zugriff auf einen einzelnen Schlüssel.
- Gewähren Sie Benutzern oder Rollen in anderen AWS Konten Zugriff.

Themen

- Überlegungen
- Verwaltung ressourcenbasierter Richtlinien
- Resource-based Beispiele für Richtlinien

Wenn Sie einem AWS Payment Cryptography Key eine ressourcenbasierte Richtlinie zuordnen, AWS verwendet Payment Cryptography die Bewertungslogik der IAM-Richtlinie, um zu ermitteln, ob ein bestimmter Principal berechtigt ist, die angeforderte Aktion auszuführen. Um den kontoübergreifenden Zugriff zu ermöglichen, können Sie in einer ressourcenbasierten Richtlinie ein ganzes Konto oder IAM-Entitäten in einem anderen Konto als Principal angeben. Cross-account Für den Zugriff sind zwei Richtlinien erforderlich:

1. Resource-based Richtlinie (Konto des Schlüsselbesitzers) — Der Schlüsselinhaber gewährt `PutResourcePolicy` damit Zugriff auf das Konto oder den IAM-Principal des Anrufers.
2. Identity-based Richtlinie (Konto des Anrufers) — Der IAM-Administrator des Anrufers muss in der IAM-Richtlinie des Anrufers auch die Aktion „ AWS Zahlungskryptografie“ zulassen (z. B.`payment-cryptography:EncryptData`).

Beide Richtlinien müssen die Aktion zulassen. Fehlt eine dieser Optionen, wird die kontoübergreifende Anfrage mit `AccessDeniedException` abgelehnt.

Wenn eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Unterschiede zwischen IAM-Rollen und Resource-based Richtlinien.

⚠ Betrieb der Kontrollebene für Ressourcenrichtlinien

Resource-based Richtlinien gelten nicht für Operationen auf der Kontrollebene für Ressourcenrichtlinien wie `PutResourcePolicy` `GetResourcePolicy`, und `DeleteResourcePolicy`. Dadurch werden mögliche Aussperrungsszenarien vermieden, in denen eine Ressourcenrichtlinie die Möglichkeit verweigern könnte, die Richtlinie selbst zu ändern oder zu entfernen. Der Zugriff auf diese Vorgänge auf der Kontrollebene wird ausschließlich durch identitätsbasierte IAM-Richtlinien geregelt.

Überlegungen

Beachten Sie Folgendes, wenn Sie ressourcenbasierte Richtlinien mit Payment Cryptography verwenden. AWS

- AWS Zahlungskryptografie erzwingt automatisch, dass kein öffentlicher Zugriff auf Schlüssel möglich ist. Sie können keine ressourcenbasierte Richtlinie erstellen, die anonymen oder öffentlichen Prinzipalen Zugriff gewährt. Jeglicher Zugriff auf AWS Payment Cryptography Keys erfordert authentifizierte AWS Principals, und der öffentliche Zugriff ist immer gesperrt.
- Resource-based Richtlinien werden pro Schlüssel angewendet. Jedem AWS Payment Cryptography Key kann höchstens eine ressourcenbasierte Richtlinie zugeordnet werden.
- Resource-based Richtlinien gelten nicht für Aliase. Wenn Sie anhand seines Alias auf einen Schlüssel verweisen, wird die dem zugrunde liegenden Schlüssel zugeordnete Ressourcenrichtlinie ausgewertet.
- Resource-based Richtlinien gelten derzeit nicht für schreibgeschützte Replica-Regionsschlüssel, die mithilfe der Multi-Region Schlüsselreplikation erstellt wurden. Ressourcenrichtlinien können nur an den Schlüssel für die primäre Region angehängt werden.
- Das `Resource` Element in einer ressourcenbasierten Richtlinie muss dem ARN des Schlüssels entsprechen, an den die Richtlinie angehängt ist, `"*"` oder genau diesem entsprechen. Die

Verwendung `"*"` wird empfohlen, da damit dasselbe Richtliniendokument für mehrere Schlüssel wiederverwendet werden kann.

- Die APIs (`PutResourcePolicy`GetResourcePolicy, und`DeleteResourcePolicy`) zur Verwaltung von Ressourcenrichtlinien sind auf die Person beschränkt AWS-Konto , die den Schlüssel besitzt. Nur Prinzipale innerhalb des Kontos des Schlüsselbesitzers können Ressourcenrichtlinien verwalten.

Verwaltung ressourcenbasierter Richtlinien

Sie können ressourcenbasierte Richtlinien für AWS Payment Cryptography Keys mithilfe der API oder verwalten. AWS CLI AWS Um diesen Befehl zu verwenden, ersetzen Sie den Befehl *`italicized placeholder text`* im Beispiel durch Ihre eigenen Informationen.

Hängen Sie eine ressourcenbasierte Richtlinie an

Verwenden Sie die `PutResourcePolicy`API-Aktion oder den put-resource-policyCLI-Befehl, um eine ressourcenbasierte Richtlinie an einen Schlüssel anzuhängen. Wenn eine Richtlinie bereits vorhanden ist, wird sie durch den Befehl ersetzt.

Im folgenden Beispiel wird eine ressourcenbasierte Richtlinie aus einer JSON-Datei an einen Schlüssel angehängt.

```
aws payment-cryptography put-resource-policy \
    --resource-arn arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h \
    --policy file://policy.json
```

Rufen Sie eine ressourcenbasierte Richtlinie ab

Verwenden Sie die `GetResourcePolicy`API-Aktion oder den get-resource-policyCLI-Befehl, um die ressourcenbasierte Richtlinie abzurufen, die einem Schlüssel zugeordnet ist.

Im folgenden Beispiel wird die ressourcenbasierte Richtlinie abgerufen, die einem Schlüssel zugeordnet ist.

```
aws payment-cryptography get-resource-policy \
    --resource-arn arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h
```

Die Antwort gibt das Richtliniendokument zurück:

```
{
    "Policy": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
                },
                "Action": [
                    "payment-cryptography:EncryptData",
                    "payment-cryptography:DecryptData"
                ],
                "Resource": "*"
            }
        ]
    }
}
```

Löschen Sie eine ressourcenbasierte Richtlinie

Verwenden Sie die DeleteResourcePolicyAPI-Aktion oder den delete-resource-policyCLI-Befehl, um die ressourcenbasierte Richtlinie aus einem Schlüssel zu entfernen.

Im folgenden Beispiel wird die ressourcenbasierte Richtlinie gelöscht, die einem Schlüssel zugeordnet ist.

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

Resource-based Beispiele für Richtlinien

Gewähren Sie kontoübergreifenden Zugriff auf einen Schlüssel

Die folgende ressourcenbasierte Richtlinie gewährt einer Rolle in einem anderen AWS Konto die Erlaubnis, einen AWS Zahlungskryptografie-Schlüssel für kryptografische Operationen zu verwenden.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
            },
            "Action": [
                "payment-cryptography:GenerateCardValidationData",
                "payment-cryptography:VerifyCardValidationData"
            ],
            "Resource": "*"
        }
    ]
}
```

Erteilen Sie verschiedenen Konten unterschiedliche Berechtigungen

Die folgende ressourcenbasierte Richtlinie zeigt, wie Prinzipalen in separaten Konten unterschiedliche Berechtigungen erteilt werden können. In diesem Beispiel kann ein 3DS Access Control Server (ACS) in einem Konto Kartenvalidierungsdaten generieren, während ein Zahlungsautorisierungsdienst in einem anderen Konto nur 3DS-Kryptogramme validieren kann.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow3DSACSToGenerate",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/3dsAcsRole"
            },
            "Action": [
                "payment-cryptography:GenerateCardValidationData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowPaymentAuthToVerify",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:role/PaymentAuthRole"
            },
            "Action": [
                "payment-cryptography:VerifyAuthRequestCryptogram"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Multi-party Zulassung für AWS Zahlungskryptografie

AWS Payment Cryptography ist in Multi-party Approval (MPA), eine Funktion von Amazon Web Services Organizations, integriert, um kritische Abläufe durch einen verteilten Genehmigungsprozess zu schützen. Mit MPA können Sie verlangen, dass mehrere vertrauenswürdige Personen bestimmte AWS Zahlungskryptografievorgänge genehmigen, bevor sie ausgeführt werden.

Themen

- -Übersicht
- Geschützte Operationen
- Voraussetzungen
- MPA aktivieren und deaktivieren
- Erste Schritte
- Beispiel: Importieren Sie ein Stammzertifikat mit aktiviertem MPA
- AWS CloudTrail Protokollierung von MPA-Ereignissen
- Überprüfen des Anforderungsstatus und Behandlung von Fehlern

-Übersicht

Multi-party Die Genehmigung fügt vertraulichen AWS Zahlungskryptografieoperationen eine zusätzliche Sicherheitsebene hinzu, da die Genehmigung einer Gruppe vertrauenswürdiger Personen erforderlich ist, bevor der Vorgang fortgesetzt werden kann. Dies schützt vor unbefugten Änderungen, falls ein einziger Satz von Anmeldeinformationen kompromittiert wird, und verhindert, dass eine einzelne Person einseitig eine Änderung vornimmt.

Ein Genehmigungsteam ist eine Gruppe von Genehmigungsberechtigten innerhalb Ihrer Organisation, die Sie für die Genehmigung oder Ablehnung geschützter Vorgangsanfragen benennen. Der Genehmigungsprozess wird vollständig von den Genehmigungsberechtigten Ihrer Organisation verwaltet. An der Genehmigung oder Ablehnung von Anfragen ist kein AWS Personal beteiligt.

Wenn MPA für einen geschützten Vorgang aktiviert ist, passiert Folgendes:

1. Ein Anforderer initiiert den geschützten Vorgang.
2. MPA erstellt eine Genehmigungssitzung und benachrichtigt die Mitglieder des Genehmigungsteams.
3. Die Mitglieder des Genehmigungsteams prüfen die Anfrage und genehmigen oder lehnen sie über das MPA-Portal ab.
4. Sobald die erforderliche Mindestanzahl an Genehmigungen erreicht ist, wird der Vorgang fortgesetzt. Wenn die Anfrage vom Genehmigungsteam abgelehnt wird oder die zulässige Sitzungszeit abläuft, bevor der Genehmigungsschwellenwert erreicht ist, wird der Vorgang nicht ausgeführt. In beiden Fällen muss der Antragsteller eine neue Anfrage einreichen, um den Vorgang erneut zu versuchen.

> **Note**
>
> Beim Import eines Root-CA-Zertifikats mit aktiviertem MPA ist der `RequesterComment` Parameter obligatorisch. Dieser Kommentar ist in der Genehmigungsbenachrichtigung enthalten, die an das Genehmigungsteam gesendet wird, und stellt den Kontext für die Anfrage bereit.

Geschützte Operationen

AWS Payment Cryptography unterstützt MPA für den folgenden Vorgang:

- `ImportKey`mit `RootCertificatePublicKey` Schlüsselmaterial — Das Importieren eines Public-Key-Stammzertifikats ist ein kritischer Vorgang, da Stammzertifikate den Vertrauensanker für alle nachfolgenden Schlüsselimporte und -exporte mithilfe eines asymmetrischen Schlüsselaustauschs bilden, z. B. TR-34 Wenn für diesen Vorgang eine Genehmigung durch mehrere Parteien erforderlich ist, wird sichergestellt, dass keine einzelne Person die Vertrauensbasis für Ihre AWS Payment Cryptography Keys einseitig einrichten oder ändern kann.

Voraussetzungen

Bevor Sie MPA mit AWS Zahlungskryptografie verwenden können, müssen Sie die folgenden Voraussetzungen erfüllen:

- Richten Sie MPA in Ihrer Amazon Web Services Organizations-Umgebung ein. Anweisungen finden Sie unter Was ist Multi-party Genehmigung? im Benutzerhandbuch Multi-party zur Genehmigung.
- Stellen Sie mindestens ein Genehmigungsteam mit den erforderlichen Genehmigern zusammen.
- Teilen Sie das Genehmigungsteam mit dem Team AWS-Konto , das Ihre AWS Zahlungskryptografie-Schlüssel enthält, mit. AWS Resource Access Manager
- Für die Multi-party Genehmigung muss das Verwaltungskonto in Ihrer Organisation aktiviert sein.

MPA aktivieren und deaktivieren

Nachdem Sie ein Genehmigungsteam eingerichtet haben, können Sie MPA für AWS Payment Cryptography aktivieren, indem Sie das Team Ihrem Konto zuordnen. Sie können MPA auch deaktivieren, indem Sie die Zuordnung zum Team aufheben. Für die Trennung ist jedoch die Zustimmung des aktuell zugewiesenen Genehmigungsteams erforderlich.

MPA aktivieren

Verwenden Sie die `AssociateMpaTeam` API-Aktion oder den associate-mpa-team CLI-Befehl, um Ihrem AWS Payment Cryptography-Konto ein Genehmigungsteam zuzuordnen. Sobald die geschützten Operationen verknüpft sind, müssen sie vom Team genehmigt werden, bevor sie fortgesetzt werden können.

```
aws payment-cryptography associate-mpa-team \
    --team-arn arn:aws:mpa:us-east-1:111122223333:team/my-approval-team
```

MPA deaktivieren

Verwenden Sie die `DisassociateMpaTeam` API-Aktion oder den disassociate-mpa-team CLI-Befehl, um die Zuordnung des Genehmigungsteams zu entfernen. Das Trennen eines Teams ist selbst ein geschützter Vorgang, für den die Genehmigung durch das aktuell zugeordnete Genehmigungsteam erforderlich ist.

```
aws payment-cryptography disassociate-mpa-team \
    --team-arn arn:aws:mpa:us-east-1:111122223333:team/my-approval-team
```

> **Important**
>
> Die Deaktivierung von MPA erfordert die Genehmigung durch das aktuell zugeordnete Genehmigungsteam. Dadurch wird sichergestellt, dass keine einzelne Person den Genehmigungsschutz für mehrere Parteien einseitig aufheben kann.

> **Note**
>
> Der `--requester-comment` Parameter ist optional für und. associate-mpa-team disassociate-mpa-team

Erste Schritte

Informationen zu den ersten Schritten mit MPA for AWS Payment Cryptography finden Sie im Multi-party Approval User Guide. Dort finden Sie detaillierte Anweisungen zur Einrichtung, einschließlich der Erstellung von Genehmigungsteams, der Konfiguration von Genehmigungsrichtlinien und der Verwaltung von Genehmigungssitzungen.

Beispiel: Importieren Sie ein Stammzertifikat mit aktiviertem MPA

Nachdem MPA aktiviert wurde und dem `ImportKey` Vorgang für ein Genehmigungsteam zugewiesen wurde`RootCertificatePublicKey`, muss die Importanforderung genehmigt werden, bevor sie fortgesetzt werden kann.

1. Ein Anforderer ruft `import-key` an, um ein Root-Public-Key-Zertifikat zu importieren. Um diesen Befehl zu verwenden, ersetzen Sie den Befehl *`italicized placeholder text`* im Beispiel durch Ihre eigenen Informationen.

```
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": "LS0tLS1CRUdJTi..."}}' \
```

```
    --requester-comment "Importing new root CA certificate for TR-34 key exchange
 with partner XYZ"
```

Die Antwort gibt einen Schlüssel zurück, der auf `KeyState` gesetzt ist`CREATE_IN_PROGRESS`, was darauf hinweist, dass die Anfrage auf Genehmigung wartet. Die Antwort enthält `MpaStatus` auch Einzelheiten zur Genehmigungssitzung:

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE",
            "KeyClass": "PUBLIC_KEY",
            "KeyAlgorithm": "RSA_4096"
        },
        "Enabled": true,
        "KeyState": "CREATE_IN_PROGRESS",
        "KeyOrigin": "EXTERNAL",
        "CreateTimestamp": "2026-04-27T10:15:30.000000+00:00",
        "UsageStartTimestamp": "2026-04-27T10:15:29.926000+00:00",
        "MpaStatus": {
            "MpaSessionArn": "arn:aws:mpa:us-
east-1:111122223333:session/abc123def456",
            "Status": "PENDING",
            "InitiationDate": "2026-04-27T10:15:30.000000+00:00"
        }
    }
}
```

2. Da MPA aktiviert ist, wird die Anfrage nicht sofort abgeschlossen. Stattdessen erstellt AWS Payment Cryptography eine Genehmigungssitzung und gibt eine Antwort zurück, die darauf hinweist, dass die Genehmigung aussteht.
3. Die Mitglieder des Genehmigungsteams erhalten eine Benachrichtigung und prüfen die Anfrage über das MPA-Portal. Sobald die erforderliche Anzahl von Genehmigern den Antrag genehmigt hat, wird der Importvorgang fortgesetzt und das Stammzertifikat wird importiert.

AWS CloudTrail Protokollierung von MPA-Ereignissen

Wenn MPA aktiviert ist, protokolliert AWS Payment Cryptography Serviceereignisse bis zum AWS CloudTrailAbschluss einer Genehmigungssitzung. Diese Ereignisse zeichnen das Ergebnis des Genehmigungsprozesses auf, einschließlich der Frage, ob die Anfrage genehmigt wurde oder nicht. Sie können diese Protokolle verwenden, um die MPA-Aktivitäten zu überprüfen und den Status geschützter Operationen zu verfolgen.

MPA-related CloudTrail Ereignisse umfassen die folgenden Felder in`serviceEventDetails`:

- `keyArn`— Der ARN des Schlüssels, der von der Operation betroffen ist.
- `operation`— Der geschützte Vorgang, der angefordert wurde.
- `mpaSessionArn`— Die ARN der MPA-Genehmigungssitzung.
- `sessionStatus`— Das Ergebnis der Genehmigungssitzung (`APPROVED`oder`FAILED`).

Genehmigte Anfrage

Das folgende Beispiel zeigt ein CloudTrail Ereignis für eine `ImportKey` Anfrage, die vom MPA-Team genehmigt wurde:


```
{
    "eventVersion": "1.11",
    "eventTime": "2026-04-28T18:49:51Z",
    "eventName": "ImportKey",
    "eventSource": "payment-cryptography.amazonaws.com",
    "eventType": "AwsServiceEvent",
    "eventCategory": "Management",
    "awsRegion": "us-east-1",
    "readOnly": false,
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "payment-cryptography.amazonaws.com"
    },
    "resources": [
        {
            "ARN": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/spa2dclzmsihlj4o",
            "accountId": "111122223333",
```

```
            "type": "AWS::PaymentCryptography::Key"
        }
    ],
    "serviceEventDetails": {
        "keyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/spa2dclzmsihlj4o",
        "operation": "ImportKey",
        "mpaSessionArn": "arn:aws:mpa:us-east-1:111122223333:session/my-approval-
team/44c76e07-8937-4d7d-bb9a-a646322e2a1e",
        "sessionStatus": "APPROVED"
    }
}
```

Anfrage fehlgeschlagen

Das folgende Beispiel zeigt ein CloudTrail Ereignis für eine `ImportKey` Anfrage, die abgelehnt wurde oder deren Zeitlimit überschritten wurde:

```
{
    "eventVersion": "1.11",
    "eventTime": "2026-04-28T18:50:35Z",
    "eventName": "ImportKey",
    "eventSource": "payment-cryptography.amazonaws.com",
    "eventType": "AwsServiceEvent",
    "eventCategory": "Management",
    "awsRegion": "us-east-1",
    "readOnly": false,
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "payment-cryptography.amazonaws.com"
    },
    "resources": [
        {
            "ARN": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/qj46ku4qimypxdo7",
            "accountId": "111122223333",
            "type": "AWS::PaymentCryptography::Key"
        }
    ],
    "serviceEventDetails": {
```

```
        "keyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/qj46ku4qimypxdo7",
        "operation": "ImportKey",
        "mpaSessionArn": "arn:aws:mpa:us-east-1:111122223333:session/my-approval-team/
b0ac1994-14e1-47a6-bf1a-0b6fc0b845f2",
        "sessionStatus": "FAILED"
    }
}
```

Weitere Informationen AWS CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Überprüfen des Anforderungsstatus und Behandlung von Fehlern

Sie können den Status einer ausstehenden MPA-Anfrage telefonisch überprüfen `GetKey`. Die Antwort enthält das `MpaStatus` Feld mit den Details der aktuellen Genehmigungssitzung. Um diesen Befehl zu verwenden, ersetzen Sie den Befehl *`italicized placeholder text`* im Beispiel durch Ihre eigenen Informationen.

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

Während die Genehmigung der Anfrage aussteht, wird die Antwort `KeyState` `MpaStatus.Status` wie `CREATE_IN_PROGRESS` folgt angezeigt`PENDING`:

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE",
            "KeyClass": "PUBLIC_KEY",
            "KeyAlgorithm": "RSA_4096"
        },
        "Enabled": true,
        "KeyState": "CREATE_IN_PROGRESS",
        "KeyOrigin": "EXTERNAL",
        "CreateTimestamp": "2026-04-27T10:15:30.000000+00:00",
        "UsageStartTimestamp": "2026-04-27T10:15:29.926000+00:00",
        "MpaStatus": {
            "MpaSessionArn": "arn:aws:mpa:us-east-1:111122223333:session/abc123def456",
            "Status": "PENDING",
```

```
            "InitiationDate": "2026-04-27T10:15:30.000000+00:00"
        }
    }
}
```

Sobald die erforderliche Anzahl von Genehmigern die Anfrage genehmigt hat, wird die Anfrage `KeyState` `MpaStatus.Status` verschoben `CREATE_COMPLETE` und zu`APPROVED`. Der Schlüssel ist jetzt einsatzbereit:

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE",
            "KeyClass": "PUBLIC_KEY",
            "KeyAlgorithm": "RSA_4096"
        },
        "Enabled": true,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "EXTERNAL",
        "CreateTimestamp": "2026-04-27T10:15:30.000000+00:00",
        "UsageStartTimestamp": "2026-04-27T10:15:29.926000+00:00",
        "MpaStatus": {
            "MpaSessionArn": "arn:aws:mpa:us-east-1:111122223333:session/abc123def456",
            "Status": "APPROVED",
            "InitiationDate": "2026-04-27T10:15:30.000000+00:00"
        }
    }
}
```

Wenn die Anfrage vom Genehmigungsteam abgelehnt wird oder die Sitzung abläuft, bevor der Genehmigungsschwellenwert erreicht ist, `KeyState` ändert sich die Änderung `CREATE_FAILED` und `MpaStatus.Status` ändert sich in`FAILED`:

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE",
            "KeyClass": "PUBLIC_KEY",
```

```
            "KeyAlgorithm": "RSA_4096"
        },
        "Enabled": true,
        "KeyState": "CREATE_FAILED",
        "KeyOrigin": "EXTERNAL",
        "CreateTimestamp": "2026-04-27T10:15:30.000000+00:00",
        "UsageStartTimestamp": "2026-04-27T10:15:29.926000+00:00",
        "MpaStatus": {
            "MpaSessionArn": "arn:aws:mpa:us-east-1:111122223333:session/abc123def456",
            "Status": "FAILED",
            "InitiationDate": "2026-04-27T10:15:30.000000+00:00",
            "StatusMessage": "Approval session expired or was denied"
        }
    }
}
```

Ein Schlüssel im `CREATE_FAILED` Status kann nicht für kryptografische Operationen verwendet werden. Um den Import erneut zu versuchen, müssen Sie eine neue `ImportKey` Anfrage einreichen, wodurch eine neue Genehmigungssitzung erstellt wird.

Fehlerbehebung AWS Zahlungskryptografie, Identität und Zugriff

Dieser Abschnitt wird um weitere Themen erweitert, sobald IAM-related Probleme identifiziert werden, die sich speziell auf die AWS Zahlungskryptografie beziehen. Allgemeine Informationen zur Problembehebung zu IAM-Themen finden Sie im Abschnitt zur Fehlerbehebung im IAM-Benutzerhandbuch.

Überwachung der AWS Zahlungskryptografie

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von AWS Payment Cryptography und Ihren anderen AWS-Lösungen. AWS bietet die folgenden Überwachungstools, um die AWS Zahlungskryptografie zu beobachten, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen:

- Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können Kennzahlen erfassen und verfolgen, benutzerdefinierte Dashboards erstellen und Alarme festlegen, die Sie benachrichtigen oder Maßnahmen ergreifen, wenn eine bestimmte Metrik einen von Ihnen festgelegten Schwellenwert erreicht. Sie können beispielsweise die Nutzung bestimmter Daten CloudWatch verfolgen APIs oder Sie benachrichtigen lassen, wenn Sie sich Ihren AWS Zahlungskryptografie-Kontingenten nähern. Weitere Informationen finden Sie im CloudWatch Amazon-Benutzerhandbuch.
- Mit Amazon CloudWatch Logs können Sie Ihre Protokolldateien von EC2 Amazon-Instances und anderen Quellen überwachen CloudTrail, speichern und darauf zugreifen. CloudWatch Logs kann Informationen in den Protokolldateien überwachen und Sie benachrichtigen, wenn bestimmte Schwellenwerte erreicht werden. Sie können Ihre Protokolldaten auch in einem sehr robusten Speicher archivieren. Weitere Informationen finden Sie im Amazon CloudWatch Logs-Benutzerhandbuch.
- AWS CloudTrailerfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS Kontos getätigt wurden, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket. Sie können feststellen, welche Benutzer und Konten angerufen wurden AWS, welcher Endpunkt aufgerufen wurde, welche Ressourcen (Schlüssel) verwendet wurden, von welcher Quell-IP-Adresse aus die Aufrufe getätigt wurden, und wann die Aufrufe stattfanden. Weitere Informationen finden Sie im AWS CloudTrail -Benutzerhandbuch.

Themen

- Protokollieren von API-Aufrufen für AWS Zahlungskryptografie mithilfe von AWS CloudTrail

Protokollieren von API-Aufrufen für AWS Zahlungskryptografie mithilfe von AWS CloudTrail

AWS Zahlungskryptografie ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen eines Benutzers, einer Rolle oder eines AWS Dienstes in der AWS Zahlungskryptografie bereitstellt. CloudTrail erfasst alle API-Aufrufe für AWS Zahlungskryptografie als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der -Konsole und Code-Aufrufe der -API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für AWS Payment Cryptography. Wenn Sie keinen Trail konfigurieren, können Sie trotzdem die neuesten Verwaltungsereignisse (Control Plane) in der CloudTrail Konsole im Ereignisverlauf einsehen. Anhand der von CloudTrail gesammelten Informationen können Sie die Anfrage an AWS Payment Cryptography, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Informationen ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Themen

- AWS Informationen zur Zahlungskryptografie in CloudTrail
- Ereignisse auf Kontrollebene in CloudTrail
- Datenereignisse in CloudTrail
- Grundlegendes zu den Protokolldateieinträgen der AWS Payment Cryptography Control Plane
- Grundlegendes AWS zur Zahlungskryptografie: Einträge in Protokolldateien auf Datenebene

AWS Informationen zur Zahlungskryptografie in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn in der AWS Zahlungskryptografie eine Aktivität auftritt, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS -Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS Konto, einschließlich Ereignissen für AWS Zahlungskryptografie, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS -Regionen. Der Trail protokolliert Ereignisse aus allen

Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen
- Empfangen von CloudTrail Protokolldateien von mehreren Konten

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Root- oder AWS Identity and Access Management (IAM-) Benutzeranmeldedaten gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail -Element userIdentity.

Ereignisse auf Kontrollebene in CloudTrail

CloudTrail protokolliert kryptografische AWS Zahlungsvorgänge wie, CreateKey,, ImportKeyDeleteKeyListKeysTagResource, und alle anderen Vorgänge auf der Kontrollebene.

Datenereignisse in CloudTrail

Datenereignisse liefern Informationen über die Ressourcenoperationen, die auf oder in einer Ressource ausgeführt werden, z. B. das Verschlüsseln einer Nutzlast oder das Übersetzen einer PIN. Datenereignisse sind umfangreiche Aktivitäten, die standardmäßig CloudTrail nicht protokolliert werden. Sie können die API-Aktionsprotokollierung von Datenereignissen für AWS Payment Cryptography-Datenebenenereignisse mithilfe unserer Konsole CloudTrail APIs aktivieren. Weitere Informationen finden Sie unter Protokollieren von Datenereignissen im Benutzerhandbuch für AWS CloudTrail .

Bei CloudTrail müssen Sie mithilfe erweiterter Event-Selektoren entscheiden, welche AWS Payment Cryptography API-Aktivitäten protokolliert und aufgezeichnet werden. Um Ereignisse auf der Datenebene von AWS Payment Cryptography zu protokollieren, müssen Sie den Ressourcentyp und angeben. `AWS Payment Cryptography key` `AWS Payment Cryptography alias` Sobald dies festgelegt ist, können Sie Ihre Protokollierungseinstellungen weiter verfeinern, indem Sie bestimmte Datenereignisse für die Aufzeichnung auswählen, z. B. die Verwendung des Filters `eventName` zum Nachverfolgen von `EncryptData`-Ereignissen. Weitere Informationen finden Sie unter `AdvancedEventSelector` in der AWS CloudTrail -API-Referenz.

> **Note**
>
> Um AWS Payment Cryptography-Datenereignisse zu abonnieren, müssen Sie erweiterte Event-Selektoren verwenden. Wir empfehlen, Key- und Alias-Ereignisse zu abonnieren, um sicherzustellen, dass Sie alle Ereignisse erhalten.

AWS Datenereignisse im Zusammenhang mit Zahlungskryptografie:

- `DecryptData`
- `EncryptData`
- `GenerateCardValidationData`
- `GenerateMac`
- `GeneratePinData`
- `ReEncryptData`
- `TranslatePinData`
- `VerifyAuthRequestCryptogram`
- `VerifyCardValidationData`
- `VerifyMac`
- `VerifyPinData`

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen finden Sie unter AWS CloudTrail – Preise.

Grundlegendes zu den Protokolldateieinträgen der AWS Payment Cryptography Control Plane

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `CreateKey` Aktion AWS Payment Cryptography demonstriert.

```
      {
    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={
    }
  }
}
```

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die AWS Zahlungskryptografie demonstriert, die die Schlüsselreplikation in mehreren Regionen ermöglicht.

```
{
    "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"
}
```


Grundlegendes AWS zur Zahlungskryptografie: Einträge in Protokolldateien auf Datenebene

Ereignisse auf der Datenebene können optional konfiguriert werden und funktionieren ähnlich wie Logs auf der Steuerungsebene, sind aber in der Regel viel umfangreicher. Aufgrund der sensiblen Natur einiger Ein- und Ausgaben bei Vorgängen auf der Datenebene von AWS Payment Cryptography kann es vorkommen, dass Sie in bestimmten Feldern die Meldung „*** Sensible Daten redigiert ***“ finden. Dies ist nicht konfigurierbar und soll verhindern, dass sensible Daten in Logs oder Trails erscheinen.

Das folgende Beispiel zeigt einen CloudTrail Protokolleintrag, der die `EncryptData` Aktion AWS Payment Cryptography demonstriert.


```
      {
```

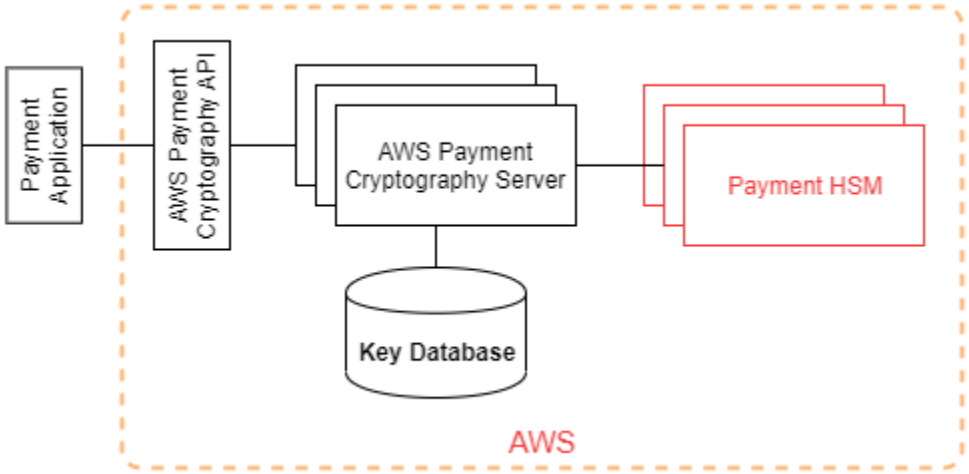
```
        "Records": [
            {
                "eventVersion": "1.09",
                "userIdentity": {
                    "type": "AssumedRole",
                    "principalId": "TESTXECZ5U2ZULLHHMJG:DataPlane-User",
                    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/DataPlane-
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"
                }
            }
        ]
    }
```

Kryptografische Details

AWS Payment Cryptography bietet eine Weboberfläche zur Generierung und Verwaltung kryptografischer Schlüssel für Zahlungstransaktionen. AWS Payment Cryptography bietet standardmäßige Schlüsselverwaltungsdienste und Kryptografie für Zahlungstransaktionen sowie Tools, die Sie für die zentrale Verwaltung und Prüfung verwenden können. Diese Dokumentation enthält eine detaillierte Beschreibung der kryptografischen Operationen, die Sie in der AWS Zahlungskryptografie verwenden können, um Sie bei der Bewertung der vom Dienst angebotenen Funktionen zu unterstützen.

AWS Payment Cryptography umfasst mehrere Schnittstellen (einschließlich einer RESTful-API über die AWS-CLI, das AWS-SDK usw. AWS-Managementkonsole), um kryptografische Operationen einer verteilten Flotte von HSM-validatedPCI-PTS-Hardwaresicherheitsmodulen anzufordern.



AWS Bei der Zahlungskryptografie handelt es sich um einen mehrstufigen Dienst, der aus webbasierten AWS Zahlungskryptografie-Hosts und einer Reihe von HSMs besteht. Die Gruppierung dieser mehrstufigen Hosts bildet den Payment Cryptography Stack. AWS Alle Anfragen an AWS Payment Cryptography müssen über das Transport Layer Security Protocol (TLS) gestellt und auf einem AWS Payment Cryptography Host beendet werden. Die Service-Hosts erlauben TLS nur mit einer Cipher Suite, die Perfect Forward Secrecy bietet. Der Dienst authentifiziert und autorisiert Ihre Anfragen mithilfe derselben Anmelde- und Richtlinienmechanismen von IAM, die für alle anderen API-Operationen verfügbar sind. AWS

AWS Kryptografie-Server für Zahlungen stellen über ein privates, nicht virtuelles Netzwerk eine Verbindung zum zugrunde liegenden HSM her. Verbindungen zwischen Servicekomponenten und HSM werden zur Authentifizierung und Verschlüsselung mit Mutual TLS (mTLS) gesichert.

Themen

- Designziele
- Grundlagen
- Interne Operationen
- Geschäftsbetrieb für Kunden

Designziele

AWS Die Zahlungskryptografie wurde entwickelt, um die folgenden Anforderungen zu erfüllen:

- Vertrauenswürdig — Die Verwendung von Schlüsseln wird durch Zugriffskontrollrichtlinien geschützt, die Sie definieren und verwalten. Es gibt keinen Mechanismus für den Export von Kryptografie-Schlüsseln im AWS Klartext-Format. Die Vertraulichkeit Ihrer kryptografischen Schlüssel ist von entscheidender Bedeutung. Für die Durchführung administrativer Aktionen am sind mehrere Amazon-Mitarbeiter mit rollenspezifischem Zugriff auf quorumbasierte Zugriffskontrollen erforderlich. HSMs Keine Amazon-Mitarbeiter haben Zugriff auf HSM-Haupt- (oder Master-) Schlüssel oder Backups. Hauptschlüssel, HSMs die nicht Teil einer AWS Payment Cryptography Region sind, können nicht synchronisiert werden. Alle anderen Schlüssel sind durch HSM-Hauptschlüssel geschützt. Daher können Kundenschlüssel für AWS Zahlungskryptografie nicht außerhalb des AWS Zahlungskryptografiedienstes verwendet werden, der innerhalb des Kundenkontos betrieben wird.
- Niedrige Latenz und hoher Durchsatz — Die AWS Zahlungskryptografie bietet kryptografische Operationen auf Latenz- und Durchsatzniveau, die für die Verwaltung kryptografischer Zahlungsschlüssel und die Verarbeitung von Zahlungstransaktionen geeignet sind.
- Haltbarkeit — Die Haltbarkeit kryptografischer Schlüssel ist so konzipiert, dass sie der Haltbarkeit der Services mit der höchsten Haltbarkeit in AWS entspricht. Ein einziger kryptografischer Schlüssel kann mit einem Zahlungsterminal, einer EMV-Chipkarte oder einem anderen sicheren kryptografischen Gerät (SCD) geteilt werden, das seit vielen Jahren verwendet wird.
- Unabhängige Regionen — AWS bietet unabhängige Regionen für Kunden, die den Datenzugriff in verschiedenen Regionen einschränken oder die Anforderungen an die Datenresidenz einhalten müssen. Die Schlüsselnutzung kann innerhalb einer AWS-Region isoliert werden.
- Sichere Quelle für Zufallszahlen — Da starke Kryptografie von einer wirklich unvorhersehbaren Zufallszahlengenerierung abhängt, bietet AWS Payment Cryptography eine hochwertige und validierte Quelle für Zufallszahlen. Die gesamte Schlüsselgenerierung für die AWS

Zahlungskryptografie verwendet HSM, das auf der PCI PTS HSM gelistet ist und im PCI-Modus arbeitet.

- Prüfung — AWS Zahlungskryptografie zeichnet die Verwendung und Verwaltung kryptografischer Schlüssel in CloudTrail Protokollen und Serviceprotokollen auf, die über Amazon verfügbar sind. CloudWatch Sie können CloudTrail Protokolle verwenden, um die Verwendung Ihrer kryptografischen Schlüssel zu überprüfen, einschließlich der Verwendung von Schlüsseln durch Konten, mit denen Sie Schlüssel geteilt haben. AWS Die Zahlungskryptografie wird von externen Prüfern anhand der geltenden PCI-, Kartenmarken- und regionalen Zahlungssicherheitsstandards geprüft. Bescheinigungen und Leitfäden zur gemeinsamen Verantwortung sind auf AWS Artifact verfügbar.
- Elastic — AWS Payment Cryptography lässt sich je nach Bedarf beliebig skalieren. Anstatt HSM-Kapazität vorherzusagen und zu reservieren, bietet Payment Cryptography AWS Zahlungskryptografie auf Abruf. AWS Payment Cryptography übernimmt die Verantwortung für die Aufrechterhaltung der Sicherheit und Konformität von HSM, um ausreichend Kapazität zur Deckung der Spitzennachfrage der Kunden bereitzustellen.

Grundlagen

Die Themen in diesem Kapitel beschreiben die kryptografischen Grundlagen der AWS Zahlungskryptografie und wo sie verwendet werden. Sie stellen auch die grundlegenden Elemente des Dienstes vor.

Themen

- Kryptografische Primitive
- Entropie und Zufallszahlengenerierung
- Symmetrische Tastenoperationen
- Asymmetrische Schlüsseloperationen
- Speicher für Schlüssel
- Schlüsselimport mit symmetrischen Schlüsseln
- Schlüsselimport mit asymmetrischen Schlüsseln
- Export von Schlüsseln
- DUKPT (Derived Unique Key Per Transaction) -Protokoll
- Schlüsselhierarchie

Kryptografische Primitive

AWS Die Zahlungskryptografie verwendet parametrierbare kryptografische Standardalgorithmen, sodass Anwendungen die für ihren Anwendungsfall erforderlichen Algorithmen implementieren können. Der Satz kryptografischer Algorithmen wird durch die Standards PCI, ANSI X9, EMVCo und ISO definiert. Die gesamte Kryptografie wird von HSMs durchgeführt, die auf der PCI-PTS-HSM-Liste stehen und im PCI-Modus ausgeführt werden.

Entropie und Zufallszahlengenerierung

AWS Die Schlüsselgenerierung für die Zahlungskryptografie erfolgt auf den Zahlungskryptografie-HSMs. AWS Die HSMs implementieren einen Zufallszahlengenerator, der die PCI-PTS-HSM-Anforderungen für alle unterstützten Schlüsseltypen und Parameter erfüllt.

Symmetrische Tastenoperationen

Symmetrische Schlüsselalgorithmen und Schlüsselstärken, die in ANSI X9 TR 31, ANSI und PCI PIN Annex C definiert sind X9.24, werden unterstützt:

- Hash-Funktionen — Algorithmen aus der SHA2- und SHA3-Familie mit einer Ausgabegröße von mehr als 2551. Mit Ausnahme der Abwärtskompatibilität mit PTS POI v3-Terminals vor PCI.
- Verschlüsselung und Entschlüsselung — AES mit einer Schlüsselgröße von mindestens 128 Bit oder TDEA mit einer Schlüsselgröße von mindestens 112 Bit (2 Schlüssel oder 3 Schlüssel).
- Message Authentication Codes (MACs) CMAC oder GMAC mit AES sowie HMAC mit einer zugelassenen Hash-Funktion und einer Schlüsselgröße größer oder gleich 128.

AWS Die Zahlungskryptografie verwendet AES 256 für HSM-Hauptschlüssel, Datenschutzschlüssel und TLS-Sitzungsschlüssel.

Hinweis: Einige der aufgelisteten Funktionen werden intern zur Unterstützung von Standardprotokollen und Datenstrukturen verwendet. In der API-Dokumentation finden Sie Informationen zu Algorithmen, die von bestimmten Aktionen unterstützt werden.

Asymmetrische Schlüsseloperationen

Asymmetrische Schlüsselalgorithmen und Schlüsselstärken, die in ANSI X9 TR 31, ANSI und PCI PIN Annex C definiert sind X9.24, werden unterstützt:

- Genehmigte zentrale Niederlassungsprogramme — wie in NIST SP800-56A (ECC/FCC2-based Key Agreement), NIST (Key Agreement) und NIST SP800-56B (IFC-based Key) beschrieben. SP800-38F AES-based encryption/wrapping

AWS Payment Cryptography Hosts erlauben nur Verbindungen zum Service über TLS mit einer Cipher Suite, die Perfect Forward Secrecy bietet.

Hinweis: Einige der aufgelisteten Funktionen werden intern zur Unterstützung von Standardprotokollen und Datenstrukturen verwendet. In der API-Dokumentation finden Sie Informationen zu Algorithmen, die von bestimmten Aktionen unterstützt werden.

Speicher für Schlüssel

AWS Schlüssel zur Zahlungskryptografie werden durch HSM AES 256-Hauptschlüssel geschützt und in ANSI X9 TR 31-Schlüsselblöcken in einer verschlüsselten Datenbank gespeichert. Die Datenbank wird in eine In-Memory-Datenbank auf Payment Cryptography-Servern repliziert. AWS

Gemäß Anhang C der PCI PIN Security Normative sind AES-256-Schlüssel genauso stark oder stärker als:

- TDEA mit 3 Tasten
- RSA 15360-Bit
- ECC 512-Bit
- DSA, DH und MQV 15360/512

Schlüsselimport mit symmetrischen Schlüsseln

AWS Die Zahlungskryptografie unterstützt den Import von Kryptogrammen und Schlüsselblöcken mit symmetrischen oder öffentlichen Schlüsseln mit einem symmetrischen Schlüssel (KEK), der genauso stark oder stärker ist als der geschützte Schlüssel für den Import.

Schlüsselimport mit asymmetrischen Schlüsseln

AWS Die Zahlungskryptografie unterstützt den Import von Kryptogrammen und Schlüsselblöcken mit symmetrischen oder öffentlichen Schlüsseln, die durch einen privaten Schlüssel (KEK) geschützt sind, der genauso stark oder stärker ist als der geschützte Schlüssel für den Import. Die Authentizität und Integrität des zur Entschlüsselung bereitgestellten öffentlichen Schlüssels muss durch ein Zertifikat einer Stelle gewährleistet werden, der der Kunde vertraut.

Öffentliche KEK, die von AWS Payment Cryptography bereitgestellt werden, verfügen über den Authentifizierungs- und Integritätsschutz einer Zertifizierungsstelle (CA), die die Einhaltung von PCI-PIN-Sicherheit und PCI P2PE Annex A bescheinigt.

Export von Schlüsseln

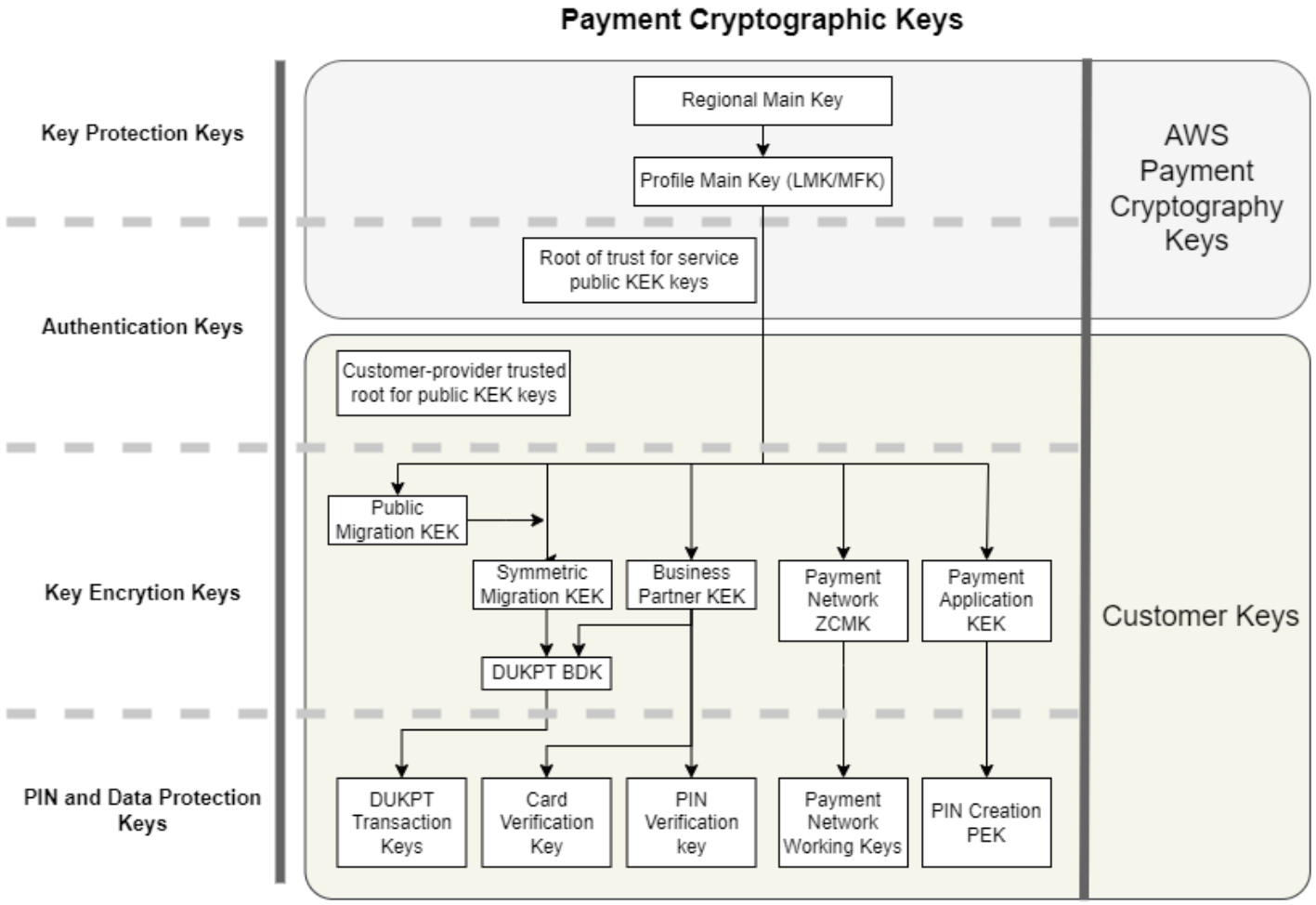
Schlüssel können exportiert und durch Schlüssel mit den entsprechenden KeyUsage Schlüsseln geschützt werden, die genauso stark oder stärker als der zu exportierende Schlüssel sind.

DUKPT (Derived Unique Key Per Transaction) -Protokoll

AWS Die Zahlungskryptografie unterstützt TDEA und AES Base Derivation Keys (BDK), wie von ANSI beschrieben. X9.24-3

Schlüsselhierarchie

Die Schlüsselhierarchie der AWS Zahlungskryptografie stellt sicher, dass Schlüssel immer durch Schlüssel geschützt werden, die genauso stark oder stärker sind als die Schlüssel, die sie schützen.



AWS Schlüssel für die Zahlungskryptografie werden für den Schlüsselschutz innerhalb des Dienstes verwendet:

Key (Schlüssel)	Description
Regionaler Hauptschlüssel	Schützt virtuelle HSM-Bilder oder Profile, die für die kryptografische Verarbeitung verwendet werden. Dieser Schlüssel ist nur in HSM- und sicheren Backups vorhanden.
Hauptschlüssel des Profils	Schlüssel zum Schutz der Kundenschlüssel auf höchster Ebene, traditionell als Local Master Key (LMK) oder Master File Key (MFK) für Kundenschlüssel bezeichnet. Dieser Schlüssel ist nur in HSM- und sicheren Backups

Key (Schlüssel)	Description
	vorhanden. Profile definieren unterschiedliche HSM-Konfigurationen, wie es die Sicherheitsstandards für Anwendungsfälle im Zahlungsverkehr erfordern.
Vertrauensbasis für KEK-Schlüssel (Public Key Encryption Key) im Bereich AWS Zahlungskryptografie	Der vertrauenswürdige öffentliche Root-Schlüssel und das Zertifikat für die Authentifizierung und Validierung von öffentlichen Schlüsseln, die von AWS Payment Cryptography für den Import und Export von Schlüsseln mithilfe asymmetrischer Schlüssel bereitgestellt werden.

Kundenschlüssel sind nach Schlüsseln gruppiert, die zum Schutz anderer Schlüssel und nach Schlüsseln zum Schutz zahlungsbezogener Daten verwendet werden. Dies sind Beispiele für Kundenschlüssel beider Typen:

Key (Schlüssel)	Description
Customer-provided vertrauenswürdiger Stamm für öffentliche KEK-Schlüssel	Von Ihnen bereitgestellter öffentlicher Schlüssel und Zertifikat als Vertrauensbasis für die Authentifizierung und Validierung von öffentlichen Schlüsseln, die Sie mit asymmetrischen Schlüsseln für den Import und Export von Schlüsseln angeben.
Verschlüsselungsschlüssel (KEK)	KEK werden ausschließlich zur Verschlüsselung anderer Schlüssel für den Austausch zwischen externen Schlüsselspeichern und AWS Zahlungskryptografie, Geschäftspartnern, Zahlungsnetzwerken oder verschiedenen Anwendungen innerhalb Ihres Unternehmens verwendet.

Key (Schlüssel)	Description
Abgeleiteter eindeutiger Schlüssel pro Transaktion (DUKPT), Basisableitungsschlüssel (BDK)	BDKs werden verwendet, um eindeutige Schlüssel für jedes Zahlungsterminal zu erstellen und Transaktionen von mehreren Terminals auf einen einzigen funktionierenden Schlüssel für die Acquirer-Bank oder den Acquirer zu übertragen. Die bewährte Methode, die für die Point-to-Point PCI-Verschlüsselung (P2PE) erforderlich ist, besteht darin, dass unterschiedliche BDKs für unterschiedliche Terminalmodelle, Schlüsselinjektions- oder -initialisierungsdienste oder andere Segmentierungen verwendet werden, um die Auswirkungen der Kompromittierung eines BDK zu begrenzen.
Hauptschlüssel für die Zonensteuerung des Zahlungsnetzwerks (ZCMK)	ZCMK, auch Zonenschlüssel oder Zonenhauptschlüssel genannt, werden von Zahlungsnetzwerken bereitgestellt, um die ersten funktionierenden Schlüssel zu erstellen.
DUKPT-Transaktionsschlüssel	Für DUKPT konfigurierte Zahlungsterminals leiten einen eindeutigen Schlüssel für das Terminal und die Transaktion ab. Das HSM, das die Transaktion empfängt, kann den Schlüssel anhand der Terminalkennung und der Transaktionssequenznummer ermitteln.

Key (Schlüssel)	Description
Schlüssel zur Vorbereitung der Kartendaten	EMV-Aussteller-Hauptschlüssel, EMV-Karte nschlüssel und Prüfwerte sowie Schlüssel zum Schutz von Kartenpersonalisierungsdate n werden verwendet, um Daten für einzelne Karten zu erstellen, die dann von einem Anbieter für Kartenpersonalisierung verwendet werden können. Diese Schlüssel und kryptogra fischen Validierungsdaten werden auch von ausstellenden Banken oder Emittenten zur Authentifizierung von Kartendaten im Rahmen der Autorisierung von Transaktionen verwendet .
Schlüssel zur Vorbereitung von Kartendaten	EMV-Aussteller-Hauptschlüssel, EMV-Karte nschlüssel und Prüfwerte sowie Schlüssel zum Schutz von Kartenpersonalisierungsdate n werden verwendet, um Daten für einzelne Karten zu erstellen, die dann von einem Anbieter für Kartenpersonalisierung verwendet werden können. Diese Schlüssel und kryptogra fischen Validierungsdaten werden auch von ausstellenden Banken oder Emittenten zur Authentifizierung von Kartendaten im Rahmen der Autorisierung von Transaktionen verwendet .
Funktionierende Schlüssel für das Zahlungsn etzwerk	Diese Schlüssel werden oft als Arbeitsschlüssel des Ausstellers oder Arbeitsschlüssel des Acquirers bezeichnet und sind die Schlüssel, mit denen Transaktionen verschlüsselt werden, die an Zahlungsnetzwerke gesendet oder von ihnen empfangen werden. Diese Schlüssel werden vom Netzwerk häufig rotiert, oft täglich oder stündlich. Dies sind PIN-Verschlüsselun gsschlüssel (PEK) für PIN/Debit Transaktionen.

Key (Schlüssel)	Description
Verschlüsselungsschlüssel (PEK) mit persönlicher Identifikationsnummer (PIN)	Anwendungen, die PIN-Blöcke erstellen oder entschlüsseln, verwenden PEK, um die Speicherung oder Übertragung von Klartext-PINs zu verhindern.

Interne Operationen

In diesem Thema werden interne Anforderungen beschrieben, die der Dienst zur Sicherung von Kundenschlüsseln und kryptografischen Vorgängen für einen weltweit verteilten und skalierbaren Zahlungskryptografie- und Schlüsselverwaltungsdienst implementiert.

Themen

- HSM-Schutz
- Allgemeine Schlüsselverwaltung
- Verwaltung von Kundenschlüsseln
- Sicherheit der Kommunikation
- Protokollierung und Überwachung

HSM-Schutz

HSM-Spezifikationen und Lebenszyklus

AWS Payment Cryptography verwendet eine Flotte von im Handel erhältlichen. HSMs Sie HSMs sind nach FIPS 140-2 Level 3 validiert und verwenden außerdem Firmware-Versionen und die Sicherheitsrichtlinien, die auf der vom PCI Security Standards Council genehmigten PCI-PTS-Geräteliste als PCI HSM v3-konform aufgeführt sind. Der PCI PTS HSM-Standard beinhaltet zusätzliche Anforderungen für die Herstellung, den Versand, die Bereitstellung, die Verwaltung und die Zerstörung von HSM-Hardware, die für die Zahlungssicherheit und die Einhaltung der Vorschriften wichtig sind, aber in FIPS 140 nicht behandelt werden.

Externe Prüfer überprüfen die Marke, das Modell, die Firmware, die Konfiguration, das physische Lebenszyklusmanagement, die Änderungskontrolle, die Bedienerzugriffskontrollen, die Verwaltung der Hauptschlüssel und alle PCI-PIN- und P2PE-Anforderungen im Zusammenhang mit dem HSM-Betrieb. HSMs

Alle HSMs werden im PCI-Modus betrieben und gemäß der PCI PTS HSM-Sicherheitsrichtlinie konfiguriert. Es sind nur Funktionen aktiviert, die zur Unterstützung von Anwendungsfällen der AWS Zahlungskryptografie erforderlich sind. AWS Bei der Zahlungskryptografie ist das Drucken, Anzeigen oder Zurückgeben von Klartext nicht vorgesehen. PINs

Physische Sicherheit von HSM-Geräten

Nur Geräteschlüssel HSMs , deren Geräteschlüssel vor der Auslieferung von einer AWS Payment Cryptography Certificate Authority (CA) vom Hersteller signiert wurden, können vom Service verwendet werden. Bei der AWS Zahlungskryptografie handelt es sich um eine Unterzertifizierungsstelle der Zertifizierungsstelle des Herstellers, die als Vertrauensbasis für HSM-Hersteller- und Gerätezertifikate dient. Die Zertifizierungsstelle des Herstellers hat die Einhaltung von PCI PIN Security Annex A und PCI P2PE Annex A bescheinigt. Der Hersteller überprüft, ob alle HSM mit Geräteschlüsseln, die von der AWS Payment Cryptography CA signiert wurden, an den von AWS angegebenen Empfänger gesendet werden.

Gemäß den Anforderungen von PCI PIN Security stellt der Hersteller eine Liste mit Seriennummern über einen anderen Kommunikationskanal als die HSM-Lieferung bereit. Diese Seriennummern werden bei jedem Schritt der HSM-Installation in AWS-Rechenzentren überprüft. Schließlich validieren die Betreiber von AWS Payment Cryptography die Liste der installierten HSM anhand der Herstellerliste, bevor sie die Seriennummer zur Liste der HSM hinzufügen, denen der Empfang AWS von Zahlungskryptografie-Schlüsseln gestattet ist.

HSMs befinden sich zu jeder Zeit in einem sicheren Speicher oder unter doppelter Kontrolle. Dazu gehören:

- Versand vom Hersteller an eine AWS-Rackmontageeinrichtung.
- Während der Rackmontage.
- Versand von der Rackmontageanlage an ein Rechenzentrum.
- Empfang und Installation in einem sicheren Verarbeitungsraum des Rechenzentrums. HSM-Racks ermöglichen eine doppelte Steuerung mit kartengesteuerten Schlössern, alarmgesteuerten Türsensoren und Kameras.
- Während des Betriebs.
- Während der Außerbetriebnahme und Zerstörung.

Für jedes chain-of-custody HSM wird ein vollständiges System mit individueller Rechenschaftspflicht geführt und überwacht.

HSM-Initialisierung

Ein HSM wird erst als Teil der AWS Payment Cryptography-Flotte initialisiert, nachdem seine Identität und Integrität anhand von Seriennummern, vom Hersteller installierten Geräteschlüsseln und Firmware-Prüfsummen überprüft wurden. Nachdem die Authentizität und Integrität eines HSM überprüft wurden, wird es konfiguriert, einschließlich der Aktivierung des PCI-Modus. Anschließend werden die Hauptschlüssel für die Region AWS Payment Cryptography und die Hauptschlüssel des Profils eingerichtet, und das HSM steht dem Dienst zur Verfügung.

Wartung und Reparatur von HSM

HSM verfügen über wartungsfähige Komponenten, für die keine Verletzung der kryptografischen Grenzen des Geräts erforderlich ist. Zu diesen Komponenten gehören Lüfter, Netzteile und Batterien. Wenn ein HSM oder ein anderes Gerät im HSM-Rack gewartet werden muss, wird die doppelte Steuerung während der gesamten Zeit, in der das Rack geöffnet ist, aufrechterhalten.

Außerbetriebnahme von HSM

Die Außerbetriebnahme erfolgt aufgrund end-of-life oder des Ausfalls eines HSM. HSM werden logisch auf Null gesetzt, bevor sie aus ihrem Rack genommen werden. Wenn sie funktionsfähig sind, werden sie dann in sicheren Verarbeitungsräumen von AWS-Rechenzentren vernichtet. Sie werden niemals zur Reparatur an den Hersteller zurückgeschickt, für einen anderen Zweck verwendet oder vor ihrer Zerstörung auf andere Weise aus einem sicheren Verarbeitungsraum entfernt.

HSM-Firmware-Update

HSM-Firmware-Updates werden bei Bedarf installiert, um die Übereinstimmung mit den auf PCI PTS HSM und FIPS 140-2 (oder FIPS 140-3) gelisteten Versionen aufrechtzuerhalten, wenn ein Update sicherheitsrelevant ist oder wenn festgestellt wird, dass Kunden von den Funktionen einer neuen Version profitieren können. AWS Für Payment Cryptography HSMs wird eine Firmware ausgeführt, die den auf PCI PTS HSM gelisteten Versionen entspricht. off-the-shelf Neue Firmware-Versionen werden anhand der PCI- oder FIPS-zertifizierten Firmware-Versionen auf Integrität geprüft und anschließend auf ihre Funktionalität getestet, bevor sie für alle verfügbar sind. HSMs

Zugriff durch den Bediener

In seltenen Fällen, in denen die während des normalen Betriebs vom HSM gesammelten Informationen nicht ausreichen, um ein Problem zu identifizieren oder eine Änderung zu planen, können Bediener zur Fehlerbehebung auch außerhalb der Konsole auf HSM zugreifen. Die folgenden Schritte werden ausgeführt:

- Aktivitäten zur Problembehandlung werden entwickelt und genehmigt, und die Sitzung außerhalb der Konsole ist geplant.
- Ein HSM wird aus dem Kundenbearbeitungsservice entfernt.
- Die Haupttasten werden gelöscht, und zwar unter doppelter Kontrolle.
- Der Bediener darf ohne Konsole auf das HSM zugreifen, um genehmigte Fehlerbehebungsaktivitäten durchzuführen, wobei die doppelte Kontrolle erfolgt.
  - Nach Beendigung der Sitzung ohne Konsole wird der erste Bereitstellungsprozess auf dem HSM durchgeführt, wobei die Standardfirmware und -konfiguration zurückgegeben und anschließend der Hauptschlüssel synchronisiert wird, bevor das HSM an die Servicekunden zurückgegeben wird.
  - Aufzeichnungen der Sitzung werden in der Änderungsnachverfolgung aufgezeichnet.
  - Die aus der Sitzung gewonnenen Informationen werden für die Planung future Änderungen verwendet.

Alle Aufzeichnungen über den Zugriff auf die Konsole werden auf Einhaltung der Prozessvorschriften und mögliche Änderungen an der HSM-Überwachung, dem non-console-access Verwaltungsprozess oder der Bedienerschulung überprüft.

Allgemeine Schlüsselverwaltung

Alle HSM in einer Region werden mit einem Region-Hauptschlüssel synchronisiert. Ein Region-Hauptschlüssel schützt mindestens einen Profil-Hauptschlüssel. Ein Profilhauptschlüssel schützt Kundenschlüssel.

Alle Hauptschlüssel werden von einem HSM generiert und durch symmetrische Schlüsselverteilung unter Verwendung asymmetrischer Techniken verteilt, die auf ANSI X9 TR 34 und PCI-PIN Anhang A abgestimmt sind.

Generation

AES-256-Bit-Hauptschlüssel werden auf einem der für die Service-HSM-Flotte bereitgestellten HSM mithilfe des PCI PTS HSM-Zufallszahlengenerators generiert.

Synchronisation der Hauptschlüssel der Region

Die Hauptschlüssel der HSM-Region werden vom Service für die gesamte regionale Flotte mithilfe von Mechanismen synchronisiert, die in ANSI X9 TR-34 definiert sind. Dazu gehören:

- Gegenseitige Authentifizierung mithilfe von Schlüsseln und Zertifikaten für den Schlüsselverteilungshost (KDH) und das Schlüsselempfangsgerät (KRD), um die Authentifizierung und Integrität von öffentlichen Schlüsseln zu gewährleisten.
- Zertifikate werden von einer Zertifizierungsstelle (CA) signiert, die die Anforderungen des PCI-PIN-Anhangs A2 erfüllt, mit Ausnahme von asymmetrischen Algorithmen und Schlüsselstärken, die für den Schutz von AES-256-Bit-Schlüsseln geeignet sind.
- Die Identifizierung und der Schlüsselschutz der verteilten symmetrischen Schlüssel entsprechen ANSI X9 TR-34 und PCI-PIN Anhang A1, mit Ausnahme von asymmetrischen Algorithmen und Schlüsselstärken, die für den Schutz von AES-256-Bit-Schlüsseln geeignet sind.

Es werden regionale Hauptschlüssel eingerichtet HSMs , die authentifiziert und für eine Region bereitgestellt wurden durch:

- Ein Hauptschlüssel wird auf einem HSM in der Region generiert. Dieses HSM ist als Host für die Schlüsselverteilung vorgesehen.
- Alle HSMs in der Region bereitgestellten Daten generieren ein KRD-Authentifizierungstoken, das den öffentlichen Schlüssel des HSM und Authentifizierungsinformationen enthält, die nicht wiedergegeben werden können.
- KRD-Token werden der KDH-Zulassungsliste hinzugefügt, nachdem das KDH die Identität und die Berechtigung des HSM zum Empfang von Schlüsseln überprüft hat.
- Das KDH erstellt für jedes HSM ein authentifizierbares Hauptschlüssel-Token. Token enthalten KDH-Authentifizierungsinformationen und einen verschlüsselten Hauptschlüssel, der nur auf ein HSM geladen werden kann, für das er erstellt wurde.
- Jedes HSM erhält das dafür erstellte Hauptschlüssel-Token. Nach der Validierung der HSM-eigenen Authentifizierungsinformationen und der KDH-Authentifizierungsinformationen wird der Hauptschlüssel mit dem privaten KRD-Schlüssel entschlüsselt und in den Hauptschlüssel geladen.

Für den Fall, dass ein einzelnes HSM mit einer Region erneut synchronisiert werden muss:

- Es wird erneut validiert und mit Firmware und Konfiguration ausgestattet.
- Wenn es neu in der Region ist:
  - Das HSM generiert ein KRD-Authentifizierungstoken.
  - Das KDH fügt das Token seiner Zulassungsliste hinzu.
  - Das KDH generiert ein Hauptschlüssel-Token für das HSM.

- Das HSM lädt den Hauptschlüssel.
- Das HSM wird dem Dienst zur Verfügung gestellt.

Dies stellt sicher, dass:

- Nur HSM, das innerhalb einer Region für die Verarbeitung von AWS Zahlungskryptografie validiert wurde, kann den Masterschlüssel dieser Region empfangen.
- Nur ein Masterschlüssel aus einem HSM für AWS Zahlungskryptografie kann an ein HSM in der Flotte verteilt werden.

Rotation der wichtigsten Schlüssel in der Region

Die Hauptschlüssel der Region werden nach Ablauf der Kryptoperiode, im unwahrscheinlichen Fall, dass ein Schlüssel kompromittiert wird, oder nach Änderungen am Dienst, die sich nachweislich auf die Sicherheit des Schlüssels auswirken, ausgetauscht.

Ein neuer Hauptschlüssel für die Region wird wie bei der ersten Bereitstellung generiert und verteilt. Die Hauptschlüssel des gespeicherten Profils müssen in den Hauptschlüssel der neuen Region übersetzt werden.

Die Rotation der Hauptschlüssel der Region hat keine Auswirkungen auf die Kundenabwicklung.

Synchronisation der Hauptschlüssel des Profils

Die Hauptschlüssel des Profils sind durch die Hauptschlüssel der Region geschützt. Dadurch wird ein Profil auf eine bestimmte Region beschränkt.

Die Hauptschlüssel des Profils werden entsprechend bereitgestellt:

- Ein Profilhauptschlüssel wird auf einem HSM generiert, bei dem der Hauptschlüssel der Region synchronisiert ist.
- Der Hauptschlüssel des Profils wird zusammen mit der Profilkonfiguration und anderem Kontext gespeichert und verschlüsselt.
- Das Profil wird von jedem HSM in der Region mit dem Hauptschlüssel der Region für kryptografische Kundenfunktionen verwendet.

Rotation der Hauptschlüssel des Profils

Die Hauptschlüssel des Profils werden nach Ablauf der Krypto-Periode, nach dem Verdacht, dass Schlüssel kompromittiert wurden, oder nach Änderungen am Dienst, die nachweislich die Sicherheit des Schlüssels beeinträchtigen, ausgetauscht.

Schritte der Rotation:

- Ein neuer Hauptschlüssel für das Profil wird generiert und wie bei der ersten Bereitstellung als ausstehender Hauptschlüssel verteilt.
- Ein Hintergrundprozess übersetzt das Kundenschlüsselmaterial vom etablierten Profilhauptschlüssel in den ausstehenden Hauptschlüssel.
- Wenn alle Kundenschlüssel mit dem ausstehenden Schlüssel verschlüsselt wurden, wird der ausstehende Schlüssel zum Hauptschlüssel des Profils heraufgestuft.
- Ein Hintergrundprozess löscht das durch den abgelaufenen Schlüssel geschützte Kundenschlüsselmaterial.

Die Rotation des Hauptschlüssels im Profil hat keine Auswirkungen auf die Kundenabwicklung.

Schutz

Schlüssel hängen nur von der zu schützenden Schlüsselhierarchie ab. Der Schutz der Hauptschlüssel ist entscheidend, um den Verlust oder die Beeinträchtigung aller Kundenschlüssel zu verhindern.

Hauptschlüssel der Region können nur aus dem Backup wiederhergestellt werden, wenn HSM authentifiziert und für den Service bereitgestellt wurde. Diese Schlüssel können nur als gegenseitig authentifizierbare, verschlüsselte Hauptschlüssel-Token von einem bestimmten KDH für ein bestimmtes HSM gespeichert werden.

Profilhauptschlüssel werden mit der Profilkonfiguration und den Kontextinformationen gespeichert, die nach Regionen verschlüsselt sind.

Kundenschlüssel werden in Schlüsselblöcken gespeichert, die durch einen Profil-Hauptschlüssel geschützt sind.

Alle Schlüssel befinden sich ausschließlich in einem HSM oder werden durch einen anderen Schlüssel mit gleicher oder höherer kryptografischer Stärke geschützt gespeichert.

Haltbarkeit

Kundenschlüssel für Transaktionskryptografie und Geschäftsfunktionen müssen auch in Extremsituationen verfügbar sein, die normalerweise zu Ausfällen führen würden. AWS Die Zahlungskryptografie verwendet ein mehrstufiges Redundanzmodell für alle Verfügbarkeitszonen und Regionen. AWS Kunden, die für kryptografische Zahlungsvorgänge eine höhere Verfügbarkeit und Beständigkeit benötigen, als sie vom Service bereitgestellt werden, sollten Architekturen mit mehreren Regionen implementieren.

Die HSM-Authentifizierung und die Hauptschlüssel-Token werden gespeichert und können zur Wiederherstellung eines Hauptschlüssels oder zur Synchronisation mit einem neuen Hauptschlüssel verwendet werden, falls ein HSM zurückgesetzt werden muss. Die Token werden archiviert und bei Bedarf nur unter doppelter Kontrolle verwendet.

Bedienerzugriff auf die HSM-Haupttasten

Hauptschlüssel existieren nur in HSM, die vom Service verwaltet und in sicheren AWS-Einrichtungen gesichert sind. Hauptschlüssel können nicht aus einem HSM exportiert oder mit einem HSM synchronisiert werden, das nicht vom Hersteller für die Verwendung im Service initialisiert wurde. AWS-Betreiber können keine Hauptschlüssel in irgendeiner Form abrufen, die in ein HSM geladen werden könnten, das nicht vom Service verwaltet wird.

Verwaltung von Kundenschlüsseln

Bei AWS hat das Vertrauen unserer Kunden oberste Priorität. Sie behalten unter Ihrem AWS-Konto die volle Kontrolle über Ihre Schlüssel, die Sie in den Service importieren oder dort erstellen. Sie behalten die Verantwortung für die Konfiguration des Zugriffs auf Schlüssel.

AWS Payment Cryptography ist ein Dienstleister, der Schlüssel im Auftrag von Kunden verwendet HSMs und verwaltet, ähnlich wie langjährige Zahlungsdienstleister. Der Service trägt die volle Verantwortung für die physische und logische Sicherheit von HSM. Der Service und die Kunden teilen sich die Verantwortung für die Schlüsselverwaltung, da der Kunde genaue Informationen zu Schlüsseln bereitstellen muss, die vom Service erstellt oder in diesen importiert wurden. Diese Informationen verwendet der Service, um die korrekte Verwendung und Verwaltung der Schlüssel durchzusetzen. Der AWS-Schutz zur Datentrennung wird verwendet, um sicherzustellen, dass die Kompromittierung von Schlüsseln, die zu einem AWS-Konto gehören, Schlüssel eines anderen nicht gefährden kann.

AWS Payment Cryptography trägt die volle Verantwortung für die physische Einhaltung der Vorschriften durch HSM und die Schlüsselverwaltung der vom Service verwalteten Schlüssel.

Dies erfordert den Besitz und die Verwaltung der HSM-Hauptschlüssel sowie den Schutz der Kundenschlüssel, die von Payment Cryptography verwaltet werden. AWS

Trennung von Kundenschlüsselraum

AWS Bei der Zahlungskryptografie werden wichtige Richtlinien für die gesamte Verwendung von Schlüsseln durchgesetzt, einschließlich der Beschränkung der Hauptbenutzer auf das Konto, dem der Schlüssel gehört, sofern ein Schlüssel nicht ausdrücklich mit einem anderen Konto geteilt wird.

AWS-Konten bieten eine vollständige Trennung der Umgebung zwischen Kunden oder Anwendungen, analog zu Implementierungen außerhalb der Cloud in verschiedenen Rechenzentren. Jedes Konto bietet isolierte Zugriffskontrolle, Netzwerke, Rechenressourcen, Datenspeicher, kryptografische Schlüssel für Datenschutz und Zahlungstransaktionen sowie alle AWS-Ressourcen. AWS-Services wie Organizations und Control Tower ermöglichen die Unternehmensverwaltung separater Anwendungskonten, analog zu Käfigen oder Räumen innerhalb eines Unternehmensrechenzentrums.

Zugriff des Betreibers auf Kundenschlüssel

Die vom Service verwalteten Kundenschlüssel werden durch Partitionshauptschlüssel geschützt gespeichert und können nur von dem Kundenkonto verwendet werden, das den Eigentümer besitzt, oder von dem Konto, das der Eigentümer speziell für die gemeinsame Nutzung von Schlüsseln konfiguriert hat. AWS-Betreiber können keine Schlüsselverwaltungs- oder kryptografischen Operationen mit Kundenschlüsseln exportieren oder ausführen, indem sie den manuellen Zugriff auf den Service verwenden, der durch manuelle AWS-Zugriffsmechanismen verwaltet wird.

Servicecode, der die Verwaltung und Verwendung von Kundenschlüsseln implementiert, unterliegt den AWS-Praktiken für sicheren Code, die gemäß der AWS PCI DSS-Bewertung bewertet wurden.

Sicherung und Wiederherstellung

Schlüssel und wichtige Informationen, die intern vom Service für eine Region gespeichert werden, werden in verschlüsselten Archiven von AWS gesichert. Für die Wiederherstellung von AWS Archiven ist eine doppelte Kontrolle erforderlich.

Schlüsselblöcke

Alle Schlüssel werden in Schlüsselblöcken im ANSI X9.143-Format gespeichert und verarbeitet.

Schlüssel können aus Kryptogrammen oder anderen Schlüsselblockformaten, die von unterstützt werden, in den Dienst importiert werden. ImportKey Ebenso können Schlüssel, sofern sie

exportierbar sind, in andere Schlüsselblockformate oder Kryptogramme exportiert werden, die von Schlüsselexportprofilen unterstützt werden.

Verwendung von Schlüsseln

Die Verwendung von Schlüsseln ist auf die KeyUsage vom Dienst konfigurierten Werte beschränkt. Der Dienst schlägt alle Anfragen fehl, bei denen der Schlüssel, der Verwendungsmodus oder der Algorithmus für den angeforderten kryptografischen Vorgang unangemessen verwendet wurden.

Beziehungen zum Schlüsselaustausch

PCI PIN Security und PCI P2PE verlangen, dass Unternehmen, die Schlüssel zur Verschlüsselung PINs oder Kartendaten gemeinsam nutzen, einschließlich der Key-Exchange-Schlüssel (KEK), die für die gemeinsame Nutzung dieser Schlüssel verwendet werden, nicht dieselben Schlüssel mit anderen Organisationen teilen. Es hat sich bewährt, dass symmetrische Schlüssel nur von zwei Parteien für einen einzigen Zweck gemeinsam genutzt werden, auch innerhalb derselben Organisation. Dadurch werden die Auswirkungen vermuteter wichtiger Sicherheitslücken minimiert, die den Austausch der betroffenen Schlüssel erzwingen könnten.

Selbst in Geschäftsfällen, bei denen Schlüssel zwischen mehr als zwei Parteien gemeinsam genutzt werden müssen, sollte die Anzahl der Parteien auf die Mindestanzahl beschränkt werden.

AWS Die Zahlungskryptografie bietet Schlüsselkennzeichnungen, anhand derer die Verwendung von Schlüsseln im Rahmen dieser Anforderungen nachverfolgt und durchgesetzt werden kann.

Beispielsweise können KEK und BDK für verschiedene Schlüsseleingabeeinrichtungen identifiziert werden, indem für alle Schlüssel, die mit diesem Dienstanbieter geteilt werden, ein „KIFPOSStation“ = „“ gesetzt wird. Ein anderes Beispiel wäre, Schlüssel, die mit Zahlungsnetzwerken geteilt werden, mit „Network“ = „PayCard“ zu kennzeichnen. Durch Tagging können Sie Zugriffskontrollen einrichten und Prüfberichte erstellen, um Ihre wichtigsten Managementpraktiken durchzusetzen und nachzuweisen.

Löschen von Schlüsseln

DeleteKey markiert Schlüssel in der Datenbank zum Löschen nach einem vom Kunden konfigurierbaren Zeitraum. Nach diesem Zeitraum wird der Schlüssel unwiederbringlich gelöscht. Dies ist ein Sicherheitsmechanismus, um das versehentliche oder böswillige Löschen eines Schlüssels zu verhindern. Schlüssel, die zum Löschen markiert sind, sind für keine anderen Aktionen verfügbar als RestoreKey.

Gelöschte Schlüssel verbleiben nach dem Löschen 7 Tage lang in Service-Backups. Sie können in diesem Zeitraum nicht wiederhergestellt werden.

Schlüssel, die zu geschlossenen AWS-Konten gehören, sind zum Löschen markiert. Wenn das Konto vor Ablauf der Löschfrist reaktiviert wird, werden alle zum Löschen markierten Schlüssel zwar wiederhergestellt, aber deaktiviert. Sie müssen von Ihnen erneut aktiviert werden, um sie für kryptografische Operationen verwenden zu können.

Sicherheit der Kommunikation

Extern

AWS API-Endpunkte für Zahlungskryptografie erfüllen AWS Sicherheitsstandards wie TLS 1.2 oder höher und Signature Version 4 für die Authentifizierung und Integrität von Anfragen.

Eingehende TLS-Verbindungen werden auf Netzwerk-Loadbalancern beendet und über interne TLS-Verbindungen an API-Handler weitergeleitet.

Intern

Die interne Kommunikation zwischen Servicekomponenten sowie zwischen Servicekomponenten und anderen AWS wird durch TLS unter Verwendung starker Kryptografie geschützt.

HSM befinden sich in einem privaten, nicht virtuellen Netzwerk, das nur von Servicekomponenten aus erreichbar ist. Alle Verbindungen zwischen HSM und Servicekomponenten sind mit Mutual TLS (mTLS) auf oder höher als TLS 1.2 gesichert. Interne Zertifikate für TLS und mTLS werden von Amazon Certificate Manager mithilfe einer privaten AWS-Zertifizierungsstelle verwaltet. Das interne Netzwerk VPCs und das HSM-Netzwerk werden auf unerwartete Aktivitäten und Konfigurationsänderungen überwacht.

Protokollierung und Überwachung

Zu den internen Serviceprotokollen gehören:

- CloudTrail Protokolle von AWS-Serviceaufrufen, die vom Service getätigt wurden
- CloudWatch Protokolle beider Ereignisse, die direkt in den CloudWatch Protokollen oder Ereignissen von HSM protokolliert wurden
- Protokolldateien von HSM- und Servicesystemen
- Log-Archive

Alle Protokollquellen überwachen und filtern nach vertraulichen Informationen, einschließlich Informationen zu Schlüsseln. Die Protokolle werden systematisch überprüft, um sicherzustellen, dass sie keine sensiblen Kundeninformationen enthalten.

Der Zugriff auf die Protokolle ist auf Personen beschränkt, die für die Erfüllung von Aufgaben benötigt werden.

Alle Protokolle werden gemäß den AWS-Richtlinien zur Aufbewahrung von Protokollen aufbewahrt.

Geschäftsbetrieb für Kunden

AWS Payment Cryptography trägt die volle Verantwortung für die physische Einhaltung der PCI-Standards durch HSM. Der Service bietet auch einen sicheren Schlüsselspeicher und stellt sicher, dass Schlüssel nur für die Zwecke verwendet werden können, die gemäß den PCI-Standards zulässig sind und von Ihnen bei der Erstellung oder beim Import angegeben wurden. Sie sind für die Konfiguration der wichtigsten Attribute und des Zugriffs verantwortlich, um die Sicherheits- und Compliance-Funktionen des Dienstes zu nutzen.

Themen

- Generieren von Schlüsseln
- Importieren von Schlüsseln
- Exportieren von Schlüsseln
- Löschen von -Schlüsseln
- Rotieren von -Schlüsseln

Generieren von Schlüsseln

Bei der Erstellung von Schlüsseln legen Sie die Attribute fest, die der Service verwendet, um die rechtskonforme Verwendung des Schlüssels zu erzwingen:

- Algorithmus und Schlüssellänge
- Usage
- Verfügbarkeit und Ablauf

Tags, die für die attributebasierte Zugriffskontrolle (ABAC) verwendet werden, um die Verwendung von Schlüsseln für bestimmte Partner oder Anwendungen einzuschränken, sollten ebenfalls bei der

Erstellung festgelegt werden. Stellen Sie sicher, dass Sie Richtlinien zur Beschränkung von Rollen angeben, die Tags löschen oder ändern dürfen.

Sie sollten sicherstellen, dass die Richtlinien, die festlegen, welche Rollen den Schlüssel verwenden und verwalten dürfen, vor der Erstellung des Schlüssels festgelegt werden.

> **Note**
>
> Die IAM-Richtlinien für die CreateKey Befehle können verwendet werden, um die doppelte Kontrolle bei der Schlüsselgenerierung durchzusetzen und nachzuweisen.

Importieren von Schlüsseln

Beim Import von Schlüsseln werden die Attribute, mit denen die gesetzeskonforme Verwendung des Schlüssels erzwungen wird, vom Dienst anhand der kryptografisch gebundenen Informationen im Schlüsselblock festgelegt. Der Mechanismus zur Festlegung des grundlegenden Schlüsselkontextes besteht in der Verwendung von Schlüsselblöcken, die mit dem Quell-HSM erstellt und durch einen gemeinsamen oder asymmetrischen KEK geschützt sind. Dies entspricht den PCI-PIN-Anforderungen und behält die Verwendung, den Algorithmus und die Schlüsselstärke der Quellanwendung bei.

Beim Import müssen zusätzlich zu den Informationen im Schlüsselblock wichtige Schlüsselattribute, Tags und Zugriffskontrollrichtlinien festgelegt werden.

Beim Import von Schlüsseln mithilfe von Kryptogrammen werden keine Schlüsselattribute aus der Quellanwendung übertragen. Mithilfe dieses Mechanismus müssen Sie die Attribute entsprechend festlegen.

Oft werden Schlüssel mithilfe von Klartext-Komponenten ausgetauscht, von Schlüsselverwaltern übertragen und dann mit einer Zeremonie verbunden, bei der die doppelte Kontrolle in einem sicheren Raum stattfindet. Dies wird von AWS Payment Cryptography nicht direkt unterstützt. Die API exportiert einen öffentlichen Schlüssel mit einem Zertifikat, das von Ihrem eigenen HSM importiert werden kann, um einen Schlüsselblock zu exportieren, der vom Dienst importiert werden kann. Das ermöglicht die Verwendung Ihres eigenen HSM zum Laden von Klartextkomponenten.

Sie sollten Key Check Values (KCV) verwenden, um zu überprüfen, ob importierte Schlüssel mit Quellschlüsseln übereinstimmen.

IAM-Richtlinien auf der ImportKey API können verwendet werden, um die doppelte Kontrolle beim Schlüsselimport durchzusetzen und nachzuweisen.

Exportieren von Schlüsseln

Die gemeinsame Nutzung von Schlüsseln mit Partnern oder lokalen Anwendungen erfordert möglicherweise den Export von Schlüsseln. Durch die Verwendung von Schlüsselblöcken für Exporte wird der grundlegende Schlüsselkontext mit dem verschlüsselten Schlüsselmaterial aufrechterhalten.

Schlüsseltags können verwendet werden, um den Export von Schlüsseln nach KEK einzuschränken, die dasselbe Tag und denselben Wert haben.

AWS Bei der Zahlungskryptografie werden Schlüsselkomponenten nicht im Klartext bereitgestellt oder angezeigt. Dies erfordert direkten Zugriff von Schlüsselverwaltern auf PCI PTS HSM oder nach ISO 13491 getestete sichere kryptografische Geräte (SCD) zur Anzeige oder zum Drucken. Sie können mit Ihrem SCD ein asymmetrisches KEK oder ein symmetrisches KEK einrichten, um die Zeremonie zur Erstellung der Schlüsselkomponenten im Klartext unter doppelter Kontrolle durchzuführen.

Mithilfe von Schlüsselprüfwerten (KCV) sollte überprüft werden, ob die vom Ziel-HSM importierten Schlüssel mit den Quellschlüsseln übereinstimmen.

Löschen von -Schlüsseln

Sie können die API zum Löschen von Schlüsseln verwenden, um festzulegen, dass Schlüssel nach einem von Ihnen konfigurierten Zeitraum gelöscht werden. Vor diesem Zeitpunkt können Schlüssel wiederhergestellt werden. Sobald Schlüssel gelöscht wurden, werden sie dauerhaft aus dem Dienst entfernt.

Die IAM-Richtlinien auf der DeleteKey API können verwendet werden, um die doppelte Kontrolle beim Löschen von Schlüsseln durchzusetzen und nachzuweisen.

Rotieren von -Schlüsseln

Der Effekt der Schlüsselrotation kann mithilfe eines Schlüsselalias implementiert werden, indem ein neuer Schlüssel erstellt oder importiert und anschließend der Schlüsselalias so geändert wird, dass er auf den neuen Schlüssel verweist. Der alte Schlüssel würde je nach Ihren Verwaltungspraktiken gelöscht oder deaktiviert.

Kontingente für AWS Payment Cryptography

Das AWS-Konto verfügt über Standardkontingente (früher als Limits bezeichnet) für jeden AWS-Service. Sofern nicht anders angegeben, ist jedes Kontingent regionsspezifisch. Sie können Erhöhungen für einige Kontingente beantragen und andere Kontingente können nicht erhöht werden.

Name	Standard	Anpas	Description
Aliase	Jede unterstützte Region: 2.000	Ja	Die maximale Anzahl von Aliasen, die Sie in diesem Konto in der aktuellen Region haben können.
Kombinierte Rate der Anfragen auf der Kontrollebene	Jede unterstützte Region: 5 pro Sekunde	Ja	Die maximale Anzahl an Anfragen der Kontrollebene pro Sekunde, die Sie in der aktuellen Region über dieses Konto stellen können. Dieses Kontingent gilt für alle Operationen auf der Kontrollebene zusammen.
Kombinierte Rate von Anfragen auf Datenebene (asymmetrisch)	Jede unterstützte Region: 20 pro Sekunde	Ja	Die maximale Anzahl von Anfragen pro Sekunde für Datenebenenoperationen mit einem asymmetrischen Schlüssel, die Sie in diesem Konto in der aktuellen Region stellen können. Dieses Kontingent gilt für alle Operationen auf Datenebene zusammen.

Name	Standard	Anpas	Description
Kombinierte Rate von Anfragen auf Datenebene (symmetrisch)	Jede unterstützte Region: 500 pro Sekunde	Ja	Die maximale Anzahl von Anfragen pro Sekunde für Datenebenenoperationen mit einem symmetrischen Schlüssel, die Sie in diesem Konto in der aktuellen Region stellen können. Dieses Kontingent gilt für alle Operationen auf Datenebene zusammen.
Schlüssel	Jede unterstützte Region: 2.000	Ja	Die maximale Anzahl von Schlüsseln, die Sie in diesem Konto in der aktuellen Region haben können, ausgenommen gelöschte Schlüssel.

Dokumenthistorie für den AWS Benutzerhandbuch für Kryptografie im Zahlungsverkehr

In der folgenden Tabelle werden die Dokumentationsversionen für AWS Payment Cryptography beschrieben.

Änderung	Beschreibung	Datum
Unterstützung für asymmetrische Schlüssel für die Schlüsselreplikation Multi-Region	Multi-Region Die Schlüsselreplikation unterstützt jetzt zusätzlich zu symmetrischen Schlüsseln auch asymmetrische Zahlungskryptographie-Schlüssel wie RSA und ECC.	27. Juli 2026
Neue Funktion — AS2805	Unterstützung für Algorithmen und Flows zur Unterstützung der regionalen Unterstützung von AS2805	17. Dezember 2025
Neues Feature — Multi-Region Schlüsselreplikation	Mit der Multi-Region Schlüsselreplikation können Sie Ihre AWS Payment Cryptography Schlüssel auf mehrere replizieren. AWS-Regionen	10. September 2025
Neues Feature — ECDH	Mit dieser Version kann ECDH verwendet werden, um einen gemeinsamen KEK für den weiteren Schlüsselaustausch einzurichten.	30. März 2025
Neue Leitlinien für den Austausch von Schlüsselbörsen	Neue Leitlinien für wichtige Börsen. Informationen zu gängigen JCB-Befehlen wurden ebenfalls hinzugefügt.	31. Januar 2025

Einführung einer neuen Region	Endpunkte für die Einführung neuer Regionen in Europa (Frankfurt), Europa (Irland), Asien-Pazifik (Singapur) und Asien-Pazifik (Tokio) hinzugefügt	31. Juli 2024
CloudTrail für Datenebene und dynamische Schlüssel	Es wurden Informationen zur Verwendung CloudTrail für (kryptografische) Operationen auf Datenebene hinzugefügt, einschließlich Beispielen. Außerdem wurden Informationen zur Verwendung dynamischer Schlüssel für bestimmte Funktionen hinzugefügt, um Schlüssel mit einmaliger oder eingeschränkter Verwendung, die nicht in Payment Cryptography importiert werden sollten, besser zu unterstützen AWS	10. Juli 2024
Aktualisierte Beispiele	Neue Beispiele für die Ausgabe von Karten hinzugefügt	1. Juli 2024
Veröffentlichung der Funktion	Hinzufügen von Informationen zu VPC-Endpunkten (PrivateLink) und iCVV-Beispielen	30. Mai 2024
Veröffentlichung der Funktion	Es wurden Informationen zu neuen Funktionen rund um die import/export Verwendung von RSA und das Exportieren von IPEK/IK DUKPT-Schlüsseln hinzugefügt.	15. Januar 2024

Erstversion	Erste Veröffentlichung des AWS Payment Cryptography User Guide	08. Juni 2023

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.