使用者指南

AWS 付款密碼編譯



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

AWS 付款密碼編譯: 使用者指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 AWS 付款密碼編譯？ 1
概念 2
產業術語 3
常見金鑰類型 4
其他術語 6
相關服務 10
如需詳細資訊 10
端點 11
控制平面端點 11
資料平面端點 14
開始使用 17
先決條件 17
步驟 1：建立金鑰 17
步驟 2：使用 金鑰產生 CVV2 值 18
步驟 3：驗證步驟 2 中產生的值 19
步驟 4：執行陰性測試 20
步驟 5：（選用） 清除 20
管理金鑰 22
建立金鑰 22
建立 2KEY TDES 基礎衍生金鑰 23
為 CVV/CVV2 建立 2KEY TDES 金鑰 25
建立 HMAC 金鑰 26
建立 AES-256 金鑰 27
建立 PIN 加密金鑰 (PEK) 28
建立非對稱 (RSA) 金鑰 29
建立 PIN 驗證值 (PVV) 金鑰 30
建立非對稱 ECC 金鑰 31
列出金鑰 32
啟用和停用 金鑰 33
啟動金鑰用量 33
停止金鑰用量 35
複寫金鑰 37
多區域金鑰複寫的優點 37
多區域金鑰複寫的運作方式 37

限制及考量 37
啟用多區域金鑰複寫 38
停用多區域金鑰複寫 40
安全考量 41
最佳實務 41
定價 41
刪除 金鑰 41
關於等待期 42
匯入和匯出金鑰 45
匯入金鑰 47
匯出金鑰 70
進階主題 89
使用別名 107
關於別名 107
在應用程式中使用別名 110
相關 API 111
取得金鑰 111
取得與金鑰對相關聯的公有金鑰/憑證 113
標記金鑰 113
關於 AWS 付款密碼編譯中的標籤 114
在主控台中檢視金鑰標籤 115
使用 API 操作管理金鑰標籤 115
控制對標籤的存取 118
使用標籤控制對金鑰的存取 122
了解金鑰屬性 125
對稱金鑰 125
非對稱金鑰 127
資料操作 129
加密、解密和重新加密資料 129
加密資料 130
解密資料 134
產生和驗證卡片資料 138
產生卡片資料 138
驗證卡片資料 140
產生、翻譯和驗證 PIN 資料 142
翻譯 PIN 資料 143

產生 PIN 資料 145
驗證 PIN 資料 149
產生並驗證 ARQC 153
產生身分驗證請求 (ARQC) 密碼編譯 154
驗證身分驗證請求 (ARQC) 密碼編譯 156
產生和驗證 MAC 159
產生 MAC 161
驗證 MAC 164
特定資料操作的金鑰類型 166
GenerateCardData 167
VerifyCardData 168
GeneratePinData （適用於 VISA/ABA 配置） 169
GeneratePinData （適用於 IBM3624) 170
VerifyPinData （適用於 VISA/ABA 配置） 171
VerifyPinData （適用於 IBM3624) 171
解密資料 172
加密資料 173
翻譯 PIN 資料 174
產生/驗證 MAC 175
GenerateMacEmvPinChange 177
VerifyAuthRequestCryptogram 178
匯入/匯出金鑰 178
未使用的金鑰類型 179
常用案例 180
發行者和發行者處理器 180
一般函數 180
網路特定函數 210
取得和付款引導程式 241
使用動態金鑰 242
區域特定功能 244
AS2805 244
初始金鑰 (KEK) 交換 245
KEK 驗證 247
建立和傳輸工作金鑰 250
匯出工作金鑰 252
接腳轉譯 252

Mac 產生和驗證 254
安全 255
資料保護 255
保護金鑰資料 256
資料加密 256
靜態加密 257
傳輸中加密 257
網際網路流量隱私權 257
恢復能力 258
區域隔離 258
多租用戶設計 259
基礎設施安全性 259
實體主機的隔離 259
使用 Amazon VPC 和 AWS PrivateLink 260
AWS 付款密碼編譯 VPC 端點的考量事項 260
建立用於 AWS 付款密碼編譯的 VPC 端點 261
連線到 VPC 端點 262
控制對 VPC 端點的存取 262
在政策陳述式中使用 VPC 端點 265
記錄您的 VPC 端點 268
混合式後量子 TLS 271
關於後量子 TLS 272
使用方式 272
關於 PQC 272
使用方式 272
安全最佳實務 273
法規遵循驗證 274
服務的合規 274
PIN 合規 275
常見主題 275
評估範圍 277
交易處理操作 278
P2PE 合規 282
身分與存取管理 283
目標對象 283
使用身分驗證 283

AWS 帳戶 根使用者 284
IAM 使用者和群組 284
IAM 角色 284
使用政策管理存取權 284
身分型政策 285
資源型政策 285
存取控制清單 (ACL) 285
其他政策類型 285
多種政策類型 286
AWS 付款密碼編譯如何與 IAM 搭配使用 286
AWS 付款密碼編譯身分型政策 286
以 AWS 付款密碼編譯標籤為基礎的授權 288
身分型政策範例 288
政策最佳實務 289
使用主控台 289
允許使用者檢視他們自己的許可 290
能夠存取 AWS 付款密碼學的所有層面 291
能夠使用指定的金鑰呼叫 APIs 291
能夠明確拒絕資源 292
資源型政策 293
考量事項 294
管理以資源為基礎的政策 295
資源型政策範例 296
多方核准 297
概觀 298
受保護的操作 298
先決條件 299
啟用和停用 MPA 299
開始使用 300
範例：匯入已啟用 MPA 的根憑證 300
AWS CloudTrail MPA 事件的記錄 301
檢查請求狀態和處理失敗 303
疑難排解 305
監控 306
CloudTrail 日誌 306
AWS CloudTrail 中的付款密碼編譯資訊 307

CloudTrail 中的控制平面事件 307
CloudTrail 中的資料事件 308
了解 AWS 付款密碼編譯控制平面日誌檔案項目 309
了解 AWS 付款密碼編譯資料平面日誌檔案項目 312
密碼編譯詳細資訊 315
設計目標 316
基礎 316
密碼編譯基本元素 317
熵和隨機數字產生 317
對稱金鑰操作 317
非對稱金鑰操作 318
金鑰儲存 318
使用對稱金鑰匯入金鑰 318
使用非對稱金鑰匯入金鑰 318
金鑰匯出 319
每個交易衍生的唯一金鑰 (DUKPT) 通訊協定 319
金鑰階層 319
內部操作 321
HSM 保護 322
一般金鑰管理 324
客戶金鑰的管理 327
通訊安全性 329
日誌記錄和監控 329
客戶操作 329
產生金鑰 330
匯入金鑰 330
匯出金鑰 331
刪除 金鑰 331
輪換 金鑰 331
配額 332
文件歷史紀錄 333
..... cccxxxv

什麼是 AWS 付款密碼編譯？

AWS Payment Cryptography 是一項受管 AWS 服務，可根據支付卡產業 (PCI) 標準提供付款處理中使用的密碼編譯函數和金鑰管理的存取權，而不需要您購買專用付款 HSM 執行個體。 AWS Payment Cryptography 提供客戶執行付款函數，例如收單機構、 付款協調者、 網路、 切換、 處理器、 和 銀行能夠將其付款密碼編譯操作移至更接近雲端應用程式的位置，並將對包含專用付款 HSMs 的輔助資料中心或主機代管設施的相依性降至最低。

此服務旨在符合適用的產業規則，包括 PCI PIN、PCI P2PE 和 PCI DSS，而且服務會利用經 PCI PTS HSM V3 和 FIPS 140-2 第 3 級認證的硬體。它旨在支援低延遲和高層級的正常運作時間和彈性。 AWS Payment Cryptography 具有完全彈性，可免除內部部署 HSMs的許多操作需求，例如需要佈建硬體、安全地管理金鑰材料，以及在安全設施中維護緊急備份。 AWS 付款密碼編譯還提供選項，讓您以電子方式與合作夥伴共用金鑰，無需共用紙質純文字元件。

您可以使用AWS 付款密碼編譯控制平面 API 來建立和管理金鑰。

您可以使用AWS 付款密碼編譯資料平面 API，將加密金鑰用於付款相關交易處理和相關聯的密碼編譯操作。

AWS 付款密碼編譯提供可用來管理金鑰的重要功能：

- 建立和管理對稱和非對稱 AWS 付款密碼編譯金鑰，包括 TDES、AES 和 RSA 金鑰，並指定其預期用途，例如產生 CVV 或衍生 DUKPT 金鑰。
- 自動安全地存放您的 AWS 付款密碼編譯金鑰，受到硬體安全模組 (HSMs) 的保護，同時強制執行使用案例之間的金鑰分離。
- 建立、刪除、列出和更新別名，這些別名是「易記名稱」，可用於存取或控制對 AWS 付款密碼編譯金鑰的存取。
- 標記您的 AWS 付款密碼編譯金鑰，用於識別、分組、自動化、存取控制和成本追蹤。
- 遵循 TR-31（可互通安全金鑰交換金鑰區塊規格），使用金鑰加密金鑰 (KEK)，在 AWS 付款密碼編譯與 HSM （或第三方） 之間匯入和匯出對稱金鑰。
- 使用 TR-34 （使用非對稱技術分配對稱金鑰的方法） 等電子方式，在 AWS 付款密碼編譯和其他系統之間匯入和匯出對稱金鑰加密金鑰 (KEK)。

您可以在密碼編譯操作中使用 AWS 付款密碼編譯金鑰，例如：

- 使用對稱或非對稱 AWS 付款密碼編譯金鑰加密、解密和重新加密資料。

- 在加密金鑰之間安全地翻譯敏感資料（例如持卡人接腳），而不會根據 PCI PIN 規則公開純文字。
- 產生或驗證持卡人資料，例如 CVV、CVV2 或 ARQC。
- 產生和驗證持卡人接腳。
- 產生或驗證 MAC 簽章。

概念

了解 AWS 付款密碼編譯中使用的基本術語和概念，以及如何使用這些術語和概念來協助您保護資料。

別名

與 AWS 付款密碼編譯金鑰相關聯的易用名稱。在許多 AWS 付款密碼編譯 API 操作中，別名可與金鑰 ARN 互換使用。別名允許金鑰輪換或以其他方式變更，而不會影響應用程式程式碼。別名名稱是最多 256 個字元的字串。它可唯一識別帳戶和區域內相關聯的 AWS 付款密碼編譯金鑰。在 AWS 付款密碼編譯中，別名名稱一律以 開頭`alias/`。

別名的格式如下：

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

例如：

```
alias/sampleAlias2
```

金鑰 ARN

金鑰 ARN 是 AWS 付款密碼編譯中金鑰項目的 Amazon Resource Name (ARN)。它是 AWS 付款密碼編譯金鑰的唯一、完全合格的識別符。金鑰 ARN 包含 AWS 帳戶、 區域和隨機產生的 ID。ARN 與金鑰材料無關或衍生自金鑰材料。由於它們會在建立或匯入操作期間自動指派，因此這些值不具有等冪性。多次匯入相同的金鑰將導致多個金鑰 ARNs 具有自己的生命週期。

金鑰 ARN 的格式如下：

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

以下是範例金鑰 ARN：

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

金鑰識別符

金鑰識別符是金鑰的參考，其中一或多個（一或多個）是 AWS 付款密碼編譯操作的典型輸入。有效的金鑰識別符可以是金鑰 Arn a Key Alias。

AWS 付款密碼編譯金鑰

AWS 付款密碼編譯金鑰（金鑰）用於所有密碼編譯函數。金鑰是由您使用建立金鑰命令直接產生，或是透過呼叫金鑰匯入新增至系統。您可以透過檢閱屬性 KeyOrigin 來判斷金鑰的來源。 AWS 付款密碼編譯也支援在密碼編譯操作期間使用的衍生金鑰或中繼金鑰，例如 DUKPT 所使用的金鑰。

這些索引鍵在建立時會同時定義不可變和不可變的屬性。屬性，例如演算法、長度和用量會在建立時定義，且無法變更。其他，例如生效日期或過期日期，都可以修改。如需AWS 付款密碼編譯金鑰屬性的完整清單，請參閱付款密碼編譯 API 參考。 AWS

AWS 付款密碼編譯金鑰具有主要由 ANSI X9 TR 31 定義的金鑰類型，這些金鑰類型會將其使用限制為 PCI PIN v3.1 要求 19 中指定的預期用途。

在儲存、與其他帳戶共用或匯出時，屬性會繫結至使用金鑰區塊的金鑰，如 PCI PIN v3.1 要求 18-3 中所指定。

金鑰是在 AWS 付款密碼編譯平台中使用稱為金鑰 Amazon Resource Name () 的唯一值來識別ARN。

> **Note**
>
> 金鑰ARN會在最初建立或匯入 AWS 付款密碼編譯服務時產生。因此，如果使用匯入金鑰功能多次新增相同的金鑰材料，相同的金鑰材料將位於多個金鑰下，ARNS但每個金鑰生命週期都不同。

產業術語

主題

- 常見金鑰類型
- 其他術語

常見金鑰類型

AWS 付款密碼編譯金鑰

AWS 付款密碼編譯金鑰存在於單一 中 AWS 區域。它由儲存在 AWS 付款密碼編譯服務中的金鑰中繼資料和材料組成。金鑰可從外部來源匯入為 TR-31 金鑰區塊，或由 AWS 付款密碼編譯服務產生。

AWK

收單機構工作金鑰 (AWK) 是通常用來在收單機構/收單機構處理器與網路 （ 例如 Visa 或 Mastercard) 之間交換資料的金鑰。過去，AWK 會利用 3DES 進行加密，並以 TR31_P0_PIN_ENCRYPTION_KEY 表示。

BDK

基礎衍生金鑰 (BDK) 是用於衍生後續金鑰的工作金鑰，通常用於 PCI PIN 和 PCI P2PE DUKPT 程序。它表示為 TR31_B0_BASE_DERIVATION_KEY。

CMK

卡片主金鑰 (CMK) 是一或多個卡片特定的金鑰 （ 通常衍生自發行者主金鑰、PAN 和 PSN)，通常是 3DES 金鑰。這些金鑰會在個人化期間存放在 EMV Chip 上。CMKs 的範例包括 AC、SMI 和 SMC 金鑰。

CMK-AC

應用程式加密法 (AC) 金鑰做為 EMV 交易的一部分來產生交易加密法，是一種卡片主金鑰。

CMK-SMI

安全訊息完整性 (SMI) 金鑰是做為 EMV 的一部分使用，以驗證使用 MAC 傳送至卡片之承載的完整性，例如 PIN 更新指令碼。這是卡片主金鑰的類型。

CMK-SMC

安全訊息機密性 (SMC) 金鑰是做為 EMV 的一部分，用來加密傳送至卡片的資料，例如 PIN 更新。這是卡片主金鑰的類型。

CVK

卡片驗證金鑰 (CVK) 是一種金鑰，用於使用定義的演算法產生 CVV、CVV2 和類似的值，以及驗證輸入。它表示為 TR31_C0_CARD_VERIFICATION_KEY。

IMK

發行者主金鑰 (IMK) 是做為 EMV 晶片卡個人化的一部分使用的主金鑰。一般而言，會有 3 個 IMKs - AC （加密圖）、SMI （完整性/簽章的指令碼主金鑰） 和 SMC （機密性/加密的指令碼主金鑰） 金鑰各一個。

IK

初始金鑰 (IK) 是 DUKPT 程序中使用的第一個金鑰，衍生自基本衍生金鑰 (BDK)。此金鑰不會處理任何交易，但會用來衍生未來將用於交易的金鑰。用於建立 IK 的衍生方法在 X9.24-1：2017 中定義。使用 TDES BDK 時，X9.24-1：2009 是適用的標準，並以初始 PIN 加密金鑰 (IPEK) 取代 IK。

IPEK

初始 PIN 加密金鑰 (IPEK) 是 DUKPT 程序中使用的初始金鑰，衍生自基本衍生金鑰 (BDK)。此金鑰不會處理任何交易，但會用來衍生未來將用於交易的金鑰。IPEK 是一種誤判，因為此金鑰也可以用來衍生資料加密和 mac 金鑰。用於建立 IPEK 的衍生方法在 X9.24-1：2009 中定義。使用 AES BDK 時，X9.24-1：2017 是適用的標準，並以初始金鑰 (IK) 取代 IPEK。

IWK

發行者工作金鑰 (IWK) 是通常用來在發行者/發行者處理器與網路 （例如 Visa 或 Mastercard) 之間交換資料的金鑰。傳統上，IWK 會利用 3DES 進行加密，並以 TR31_P0_PIN_ENCRYPTION_KEY 表示。

KBPK

金鑰區塊保護金鑰 (KBPK) 是一種對稱金鑰，用於保護金鑰區塊，進而包裝/加密其他金鑰。KBPK 類似於 KEK，但 KEK 會直接保護金鑰材料，而在 TR-31 和類似結構描述中，KBPK 只會間接保護工作金鑰。使用 時TR-31，TR31_K1_KEY_BLOCK_PROTECTION_KEY 是正確的金鑰類型，但 TR31_K0_KEY_ENCRYPTION_KEY 可互換支援用於歷史用途。

KEK

金鑰加密金鑰 (KEK) 是用來加密其他金鑰的金鑰，用於傳輸或儲存。用於保護其他金鑰的金鑰通常根據TR-31標準具有 TR31_K0_KEY_ENCRYPTION_KEY 的 KeyUsage。

PEK

PIN 加密金鑰 (PEK) 是一種用於加密 PINs的工作金鑰，用於在雙方之間進行儲存或傳輸。IWK 和 AWK 是 PIN 加密金鑰之特定用途的兩個範例。這些金鑰以 TR31_P0_PIN_ENCRYPTION_KEY 表示。

PGK

PGK (Pin Generation Key) 是 PIN 驗證金鑰的另一個名稱。它實際上不會用來產生接腳（預設為密碼編譯隨機數字），而是用來產生驗證值，例如 PVV。

PRK

主要區域金鑰是已啟用複寫之指定付款密碼編譯金鑰的授權複寫來源。PRK 是多區域金鑰複寫組態中來源付款密碼編譯金鑰角色的參考。在付款密碼編譯金鑰上啟用複寫時，稱為該特定金鑰複寫組態的 PRK。

PVK

PIN 驗證金鑰 (PVK) 是一種工作金鑰，用於產生 PVV 等 PIN 驗證值。兩種最常見的類型是用於產生 IBM36243624 位移值的 TR31_V1_IBM3624_PIN_VERIFICATION_KEY，以及用於 Visa/ABA 驗證值的 TR31_V2_VISA_PIN_VERIFICATION_KEY。這也可以稱為 PIN 產生金鑰。

RRK

複本區域金鑰是複寫的金鑰材料和中繼資料，可安全地從 PRK 複製到設定的複本 AWS 區域。RRK 是付款密碼編譯金鑰的唯讀複本。RRK 是特定金鑰在多區域金鑰複寫組態中扮演的角色參考。任何金鑰中繼資料變更，包括複寫設定都必須套用至 PRK。

其他術語

ARQC

授權請求密碼編譯 (ARQC) 是 EMV 標準晶片卡（或同等非接觸式實作）在交易時間產生的密碼編譯。一般而言，ARQC 是由晶片卡產生，並轉送給發行者或其代理，以在交易時間進行驗證。

CVV

卡片驗證值是傳統上內嵌在磁帶條紋上的靜態秘密值，用於驗證交易的真實性。此演算法也用於其他用途，例如 iCVV、CAVV、CVV2。對於其他使用案例，它可能不會以這種方式內嵌。

CVV2

卡片驗證值 2 是傳統上列印在支付卡正面（或背面）的靜態秘密值，用於驗證卡片不存在付款的真實性（例如電話或線上）。它使用與 CVV 相同的演算法，但服務代碼設定為 000。

iCVV

iCVV 是CVV2-like 的值，但內嵌 EMV（晶片）卡上的 track2 對等資料。此值使用服務代碼 999 計算，不同於 CVV1/CVV2，以防止使用遭竊的資訊來建立新的不同類型的付款登入資料。例如，如果取得晶片交易資料，則無法使用此資料產生磁條 (CVV1) 或線上購買 (CVV2)。

它使用???金鑰

杜克普

每個交易衍生的唯一金鑰 (DUKPT) 是一種金鑰管理標準，通常用於定義在實體 POS/POI 上使用一次性加密金鑰。過去，DUKPT 會利用 3DES 進行加密。DUKPT 的業界標準在 ANSI X9.24-3-2017 中定義。

ECC

ECC （橢圓曲線密碼編譯） 是一種公有金鑰密碼編譯系統，使用橢圓曲線的數學來建立加密金鑰。ECC 提供與 RSA 等傳統方法相同的安全層級，但金鑰長度較短，以更有效率的方式提供同等的安全性。這與 RSA 不是實際解決方案 (RSA 金鑰長度 > 4096 位元） 的使用案例特別相關。AWS 付款密碼編譯支援 NIST 定義的曲線，可用於 ECDH 操作。

ECDH

ECDH (Elliptic Curve Diffie-Hellman) 是一種金鑰協議協定，允許雙方建立共用秘密 （例如 KEK 或 PEK)。在 ECDH 中，A 和 B 各自有自己的公有/私有金鑰對，並互相交換公有金鑰 （以 AWS 付款密碼編譯的憑證形式） 以及金鑰衍生中繼資料 （衍生方法、雜湊類型和共用資訊）。雙方將其私有金鑰乘以對方的公有金鑰，由於橢圓曲線屬性，雙方能夠衍生 （產生） 產生的金鑰。

EMV

EMV （最初是 Europay、Mastercard、 Visa) 是與付款利益相關者合作的技術機構，可建立互通的付款標準和技術。其中一個範例標準是晶片/非接觸式卡及其互動的付款終端機，包括所使用的密碼編譯。EMV 金鑰衍生是指根據初始金鑰集 （例如）為每個支付卡產生唯一金鑰的方法 IMK

HSM

硬體安全模組 (HSM) 是一種實體裝置，可保護密碼編譯操作 （例如加密、解密和數位簽章），以及用於這些操作的基礎金鑰。

KCAAS

Key Custodian As A Service (KCAAS) 提供各種與金鑰管理相關的服務。對於付款金鑰，它們通常可以將紙質金鑰元件轉換為 AWS 付款密碼編譯支援的電子形式，或將電子保護金鑰轉換為某些廠商可能需要的紙質元件。他們也可能為遺失會對持續操作造成負面影響的金鑰提供金鑰託管服務。KCAAS 廠商能夠協助客戶減輕在安全服務之外管理金鑰材料的操作負擔，例如付款密碼編譯，以符合 PCI DSS、PCI PIN AWS 和 PCI P2PE 標準。 AWS 付款密碼編譯提供實體金鑰交換內建的 KCAAS 功能，可將紙質型金鑰元件轉換為電子格式。

KCV

金鑰檢查值 (KCV) 是指各種主要檢查總和方法，用於互相比較金鑰，而無法存取實際金鑰材料。KCV 也用於完整性驗證 （特別是交換金鑰時），雖然此角色現在包含在 等金鑰區塊格式中TR-31。對於 TDES 金鑰，KCV 是透過加密 8 個位元組來計算，每個位元組的值為零，要檢查的金鑰並保留加密結果的 3 個最高順序位元組。對於 AES 金鑰，使用 CMAC 演算法計算 KCV，其中輸入資料為 16 個位元組零，並保留加密結果的 3 個最高順序位元組。對於 HMAC 金鑰，使用在零長度訊息上建立金鑰時選取的雜湊來計算 KCV，最多需要 3 個位元組。

KDH

金鑰分佈主機 (KDH) 是在 TR-34 等金鑰交換程序中傳送金鑰的裝置或系統。從 AWS 付款密碼編譯傳送金鑰時，會被視為 KDH。

KIF

金鑰注入設施 (KIF) 是用於初始化付款終端機的安全設施，包括使用加密金鑰載入它們。

KRD

金鑰接收裝置 (KRD) 是在 TR-34 等金鑰交換程序中接收金鑰的裝置。將金鑰傳送至 AWS 付款密碼編譯時，會被視為 KRD。

KSN

金鑰序號 (KSN) 是做為 DUKPT 加密/解密輸入的值，用於為每個交易建立唯一的加密金鑰。KSN 通常包含 BDK 識別符、半唯一終端機 ID，以及在指定付款終端機上處理的每個轉換上遞增的交易計數器。根據 X9.24，對於 TDES，10 位元組 KSN 通常包含金鑰集 ID 的 24 位元、終端機 ID 的 19 位元，以及交易計數器的 21 位元，雖然金鑰集 ID 和終端機 ID 之間的界限不會影響 AWS 付款密碼編譯的函數。對於 AES，12 位元組 KSN 通常包含 32 位元的 BDK ID、32 位元的衍生識別符 (ID)，以及 32 位元的交易計數器。

MPoC

MPoC （商業硬體上的行動銷售點） 是 PCI 標準，可解決解決方案的安全要求，讓商家能夠使用智慧型手機或其他商用off-the-shelf(COTS) 行動裝置接受持卡人 PINs 碼或感應式付款。

PAN

主要帳戶號碼 (PAN) 是帳戶的唯一識別符，例如信用卡或簽帳金融卡。長度通常為 13-19 位數。前 6-8 位數識別網路和發行銀行。

PIN 區塊

在處理或傳輸期間包含 PIN 的資料區塊，以及其他資料元素。PIN 區塊格式會將 PIN 區塊的內容標準化，以及如何處理以擷取 PIN。大多數 PIN 區塊是由 PIN、PIN 長度組成，並且經常包含部分或

全部 PAN。 AWS 付款密碼編譯支援 ISO 9564-1 格式 0、1、3 和 4。AES 金鑰需要格式 4。驗證或翻譯 PINs 時，需要指定傳入或傳出資料的 PIN 區塊。

POI

互動點 (POI) 也經常匿名與銷售點 (POS) 搭配使用，是持卡人用來呈現其付款憑證的硬體裝置。POI 的範例是商家位置中的實體終端機。如需經認證的 PCI PTS POI 終端機清單，請參閱 PCI 網站。

PSN

PAN 序號 (PSN) 是用來區分以相同 PAN 發行之多張卡片的數值。

公有金鑰

使用非對稱密碼 (RSA、ECC) 時，公有金鑰是公有私有金鑰對的公有元件。公有金鑰可以共用並分配至需要為公有-私有金鑰對擁有者加密資料的實體。對於數位簽章操作，公有金鑰用於驗證簽章。

私有金鑰

使用非對稱密碼 (RSA、ECC) 時，私有金鑰是公有/私有金鑰對的私有元件。私有金鑰用於解密資料或建立數位簽章。與對稱 AWS 付款密碼編譯金鑰類似，私有金鑰是由 HSMs 安全建立。它們只會解密到 HSM 的揮發性記憶體，而且只會在處理密碼編譯請求所需的時間內解密。

PVV

PIN 驗證值 (PVV) 是一種密碼編譯輸出，可用於驗證 PIN，而不儲存實際的 PIN。雖然這是一般術語，但在 AWS 付款密碼編譯中，PVV 是指 Visa 或 ABA PVV 方法。此 PVV 是一個四位數號碼，其輸入為卡號、平移序號、平移本身和 PIN 驗證金鑰。在驗證階段， AWS 付款密碼編譯會使用交易資料在內部重新建立 PVV，並再次比較 AWS 付款密碼編譯客戶儲存的值。透過這種方式，它在概念上類似於密碼編譯雜湊或 MAC。

RSA 包裝/取消包裝

RSA 包裝使用非對稱金鑰來包裝對稱金鑰 （例如 TDES 金鑰） 以傳輸到另一個系統。只有具有相符私有金鑰的系統才能解密承載並載入對稱金鑰。相反地，RDA 取消包裝會安全地解密使用 RSA 加密的金鑰，然後將金鑰載入 AWS 付款密碼編譯。RSA 包裝是一種低階的金鑰交換方法，不會以金鑰區塊格式傳輸金鑰，也不會利用傳送方的承載簽署。替代控制項應視為確定提供，且金鑰屬性不會變更。

TR-34 也會在內部使用 RSA，但是獨立的格式，且無法互通。

服務代碼

服務代碼是在支付卡的磁條和晶片上編碼的三位數值，用於定義卡片屬性，例如卡片可以使用的位置，以及是否需要額外的授權。計算驗證值時，會使用服務代碼做為輸入。不同的服務代碼值會產

生不同類型的驗證值，例如 CVV （使用卡片磁條中的服務代碼）、iCVV （晶片卡的服務代碼 999) 和 CVV2 （服務代碼 000)。

TR-31

TR-31 （正式定義為 ANSI X9 TR 31) 是一種金鑰區塊格式，由美國國家標準研究所 (ANSI) 定義，以支援在與金鑰資料本身相同的資料結構中定義金鑰屬性。TR-31 金鑰區塊格式會定義一組繫結至金鑰的金鑰屬性，以便它們一起保存。 AWS 付款密碼編譯會盡可能使用 TR-31 標準化詞彙，以確保適當的金鑰分離和金鑰用途。TR-31 已由 ANSI X9.143-2022 取代。

TR-34

TR-34 是 ANSI X9.24-2 的實作，描述使用非對稱技術 （例如 RSA) 安全地分發對稱金鑰 （例如 3DES 和 AES) 的通訊協定。 AWS 付款密碼編譯使用 TR-34 方法來允許安全匯入和匯出金鑰。

X9.143

X9.143 是美國國家標準協會 (ANSI) 定義的金鑰區塊格式，可支援保護相同資料結構中的金鑰和金鑰屬性。金鑰區塊格式會定義一組繫結至金鑰的金鑰屬性，以便它們一起保存。 AWS 付款密碼編譯會盡可能使用 X9.143 標準化詞彙，以確保適當的金鑰分離和金鑰用途。X9.143 取代了較早的 TR-31 提案，雖然在大多數情況下，它們是向後和向前相容的，而且這些術語通常可互換使用。

相關服務

AWS Key Management Service

AWS Key Management Service (AWS KMS) 是一種受管服務，可讓您輕鬆地建立和控制用於保護資料的加密金鑰。 AWS KMS 使用硬體安全模組 HSMs) 來保護和驗證您的 AWS KMS 金鑰。

AWS CloudHSM

AWS CloudHSM 可在 AWS 雲端為客戶提供專用的一般用途 HSM 執行個體。 AWS CloudHSM 可以提供各種密碼編譯函數，例如建立金鑰、資料簽署或加密和解密資料。

如需詳細資訊

- 若要了解 AWS 付款密碼編譯中使用的術語和概念，請參閱AWS 付款密碼編譯概念。
- 如需 AWS 付款密碼編譯控制平面 API 的相關資訊，請參閱AWS 付款密碼編譯控制平面 API 參考。
- 如需 AWS 付款密碼編譯資料平面 API 的資訊，請參閱AWS 付款密碼編譯資料平面 API 參考。

- 如需 AWS 付款密碼編譯如何使用密碼編譯和保護 AWS 付款密碼編譯金鑰的詳細資訊，請參閱密碼編譯詳細資訊。

的端點 AWS Payment Cryptography

若要以程式設計方式連接到 AWS Payment Cryptography，您可以使用端點，即服務的進入點 URL。AWS SDKs 和命令列工具 AWS 區域 會根據請求的區域內容，在 中自動使用服務的預設端點，因此通常不需要明確設定這些值。如有需要，您可以為 API 請求指定不同的端點。

控制平面端點

區域名稱	區域	端點	通訊協定	
美國東部 (俄亥俄)	us-east-2	controlplane.payment-cryptography.us-east-2.amazonaws.com controlplane.payment-cryptography.us-east-2.api.aws	HTTPS HTTPS	
美國東部 (維吉尼亞北部)	us-east-1	controlplane.payment-cryptography.us-east-1.amazonaws.com controlplane.payment-cryptography.us-east-1.api.aws	HTTPS HTTPS	
美國西部 (奧勒岡)	us-west-2	controlplane.payment-cryptography.us-west-2.amazonaws.com controlplane.payment-cryptography.us-west-2.api.aws	HTTPS HTTPS	
Africa (Cape Town)	af-south-1	controlplane.payment-cryptography.af-south-1.amazonaws.com controlplane.payment-cryptography.af-south-1.api.aws	HTTPS HTTPS	

區域名稱	區域	端點	通訊協定	
亞太區域 (海德拉巴)	ap-south-2	controlplane.payment-cryptography.ap-south-2.amazonaws.com controlplane.payment-cryptography.ap-south-2.api.aws	HTTPS HTTPS	
亞太區域 (孟買)	ap-south-1	controlplane.payment-cryptography.ap-south-1.amazonaws.com controlplane.payment-cryptography.ap-south-1.api.aws	HTTPS HTTPS	
亞太區域 (大阪)	ap-northeast-3	controlplane.payment-cryptography.ap-northeast-3.amazonaws.com controlplane.payment-cryptography.ap-northeast-3.api.aws	HTTPS HTTPS	
亞太區域 (新加坡)	ap-southeast-1	controlplane.payment-cryptography.ap-southeast-1.amazonaws.com controlplane.payment-cryptography.ap-southeast-1.api.aws	HTTPS HTTPS	
亞太地區 (悉尼)	ap-southeast-2	controlplane.payment-cryptography.ap-southeast-2.amazonaws.com controlplane.payment-cryptography.ap-southeast-2.api.aws	HTTPS HTTPS	
亞太區域 (東京)	ap-northeast-1	controlplane.payment-cryptography.ap-northeast-1.amazonaws.com controlplane.payment-cryptography.ap-northeast-1.api.aws	HTTPS HTTPS	

區域名稱	區域	端點	通訊協定
加拿大 (中部)	ca-central-1	controlplane.payment-cryptography.ca-central-1.amazonaws.com controlplane.payment-cryptography.ca-central-1.api.aws	HTTPS HTTPS
歐洲 (法蘭克福)	eu-central-1	controlplane.payment-cryptography.eu-central-1.amazonaws.com controlplane.payment-cryptography.eu-central-1.api.aws	HTTPS HTTPS
歐洲 (愛爾蘭)	eu-west-1	controlplane.payment-cryptography.eu-west-1.amazonaws.com controlplane.payment-cryptography.eu-west-1.api.aws	HTTPS HTTPS
歐洲 (倫敦)	eu-west-2	controlplane.payment-cryptography.eu-west-2.amazonaws.com controlplane.payment-cryptography.eu-west-2.api.aws	HTTPS HTTPS
歐洲 (巴黎)	eu-west-3	controlplane.payment-cryptography.eu-west-3.amazonaws.com controlplane.payment-cryptography.eu-west-3.api.aws	HTTPS HTTPS
南美洲 (聖保羅)	sa-east-1	controlplane.payment-cryptography.sa-east-1.amazonaws.com controlplane.payment-cryptography.sa-east-1.api.aws	HTTPS HTTPS

資料平面端點

區域名稱	區域	端點	通訊協定
美國東部 (俄亥俄)	us-east-2	dataplane.payment-cryptography.us-east-2.amazonaws.com dataplane.payment-cryptography.us-east-2.api.aws	HTTPS HTTPS
美國東部 (維吉尼亞北部)	us-east-1	dataplane.payment-cryptography.us-east-1.amazonaws.com dataplane.payment-cryptography.us-east-1.api.aws	HTTPS HTTPS
美國西部 (奧勒岡)	us-west-2	dataplane.payment-cryptography.us-west-2.amazonaws.com dataplane.payment-cryptography.us-west-2.api.aws	HTTPS HTTPS
Africa (Cape Town)	af-south-1	dataplane.payment-cryptography.af-south-1.amazonaws.com dataplane.payment-cryptography.af-south-1.api.aws	HTTPS HTTPS
亞太區域 (海德拉巴)	ap-south-2	dataplane.payment-cryptography.ap-south-2.amazonaws.com dataplane.payment-cryptography.ap-south-2.api.aws	HTTPS HTTPS
亞太區域 (孟買)	ap-south-1	dataplane.payment-cryptography.ap-south-1.amazonaws.com dataplane.payment-cryptography.ap-south-1.api.aws	HTTPS HTTPS

區域名稱	區域	端點	通訊協定
亞太區域 (大阪)	ap-northeast-3	dataplane.payment-cryptography.ap-northeast-3.amazonaws.com dataplane.payment-cryptography.ap-northeast-3.api.aws	HTTPS HTTPS
亞太區域 (新加坡)	ap-southeast-1	dataplane.payment-cryptography.ap-southeast-1.amazonaws.com dataplane.payment-cryptography.ap-southeast-1.api.aws	HTTPS HTTPS
亞太地區 (悉尼)	ap-southeast-2	dataplane.payment-cryptography.ap-southeast-2.amazonaws.com dataplane.payment-cryptography.ap-southeast-2.api.aws	HTTPS HTTPS
亞太區域 (東京)	ap-northeast-1	dataplane.payment-cryptography.ap-northeast-1.amazonaws.com dataplane.payment-cryptography.ap-northeast-1.api.aws	HTTPS HTTPS
加拿大 (中部)	ca-central-1	dataplane.payment-cryptography.ca-central-1.amazonaws.com dataplane.payment-cryptography.ca-central-1.api.aws	HTTPS HTTPS
歐洲 (法蘭克福)	eu-central-1	dataplane.payment-cryptography.eu-central-1.amazonaws.com dataplane.payment-cryptography.eu-central-1.api.aws	HTTPS HTTPS

區域名稱	區域	端點	通訊協定
歐洲 (愛爾蘭)	eu-west-1	dataplane.payment-cryptography.eu-west-1.amazonaws.com dataplane.payment-cryptography.eu-west-1.api.aws	HTTPS HTTPS
歐洲 (倫敦)	eu-west-2	dataplane.payment-cryptography.eu-west-2.amazonaws.com dataplane.payment-cryptography.eu-west-2.api.aws	HTTPS HTTPS
歐洲 (巴黎)	eu-west-3	dataplane.payment-cryptography.eu-west-3.amazonaws.com dataplane.payment-cryptography.eu-west-3.api.aws	HTTPS HTTPS
南美洲 (聖保羅)	sa-east-1	dataplane.payment-cryptography.sa-east-1.amazonaws.com dataplane.payment-cryptography.sa-east-1.api.aws	HTTPS HTTPS

開始使用 AWS 付款密碼編譯

若要開始使用 AWS 付款密碼編譯，您必須先建立金鑰，然後在各種密碼編譯操作中使用它們。以下教學課程提供簡單的使用案例，以產生用於產生/驗證 CVV2 值的金鑰。若要嘗試其他範例並探索 AWS 內的部署模式，請嘗試下列AWS 付款密碼編譯研討會，或探索 GitHub 上可用的範例專案

本教學課程會逐步引導您建立單一金鑰，並使用 金鑰執行密碼編譯操作。之後，如果您不想再刪除金鑰，則會刪除金鑰，這會完成金鑰生命週期。

> **Warning**
>
> 本使用者指南中的範例可能會使用範例值。我們強烈建議不要在生產環境中使用範例值，例如金鑰序號。

主題

- 先決條件
- 步驟 1：建立金鑰
- 步驟 2：使用 金鑰產生 CVV2 值
- 步驟 3：驗證步驟 2 中產生的值
- 步驟 4：執行陰性測試
- 步驟 5：（選用）清除

先決條件

在開始之前，請確認：

- 您有權存取 服務。如需詳細資訊，請參閱 IAM 政策。
- 您已AWS CLI安裝 。您也可以使用 AWS SDKs 或 AWS APIs 來存取 AWS 付款密碼編譯，但本教學中的指示會使用 AWS CLI。

步驟 1：建立金鑰

第一步是建立金鑰。在本教學課程中，您會建立 CVK 雙長度 3DES (2KEY TDES) 金鑰來產生和驗證 CVV/CVV2 值。

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

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
    }
}
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/tqv5yij6wtxx64pi。在下一個步驟中，您需要用到。

步驟 2：使用 金鑰產生 CVV2 值

在此步驟中，您會使用步驟 1 中的 金鑰，為指定PAN和過期日期產生 CVV2。

```
$  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"
}
```

請記下 cardDataValue，在此例中為 3 位數字 144。在下一個步驟中，您需要用到。

步驟 3：驗證步驟 2 中產生的值

在此範例中，您可以使用您在步驟 1 中建立的金鑰來驗證步驟 2 的 CVV2。

執行下列命令來驗證 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"
}
```

服務傳回 200 的 HTTP 回應，表示已驗證 CVV2。

步驟 4：執行陰性測試

在此步驟中，您會建立陰性測試，其中 CVV2 不正確且未驗證。您嘗試使用您在步驟 1 中建立的金鑰來驗證不正確的 CVV2。這是預期的操作，例如，如果持卡人在結帳時輸入錯誤的 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 999
```

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

服務傳回 400 的 HTTP 回應，其中包含「卡片驗證資料驗證失敗」訊息，以及 INVALID_VALIDATION_DATA 的原因。

步驟 5：（選用）清除

現在您可以刪除在步驟 1 中建立的金鑰。若要將無法復原的變更降至最低，預設金鑰刪除期間為七天。

```
$ 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"
    }
}
```

請記下輸出中的兩個欄位。預設為未來的deletePendingTimestamp七天。keyState 設定為DELETE_PENDING。您可以在排定的刪除時間之前隨時呼叫 來取消此刪除restore-key。

管理金鑰

若要開始使用 AWS 付款密碼編譯，請建立 AWS 付款密碼編譯金鑰。

本節說明如何在整個生命週期中建立和管理各種 AWS 付款密碼編譯金鑰類型。您將了解如何建立、檢視和編輯金鑰，以及如何標記金鑰、建立金鑰別名，以及啟用或停用金鑰。

AWS 付款密碼編譯金鑰是區域資源。如果您想要在多個 中使用指定的金鑰 AWS 區域，您可以啟用多區域金鑰複寫，以安全地將金鑰材料和中繼資料複製到 AWS 區域 您在相同 AWS 分割區和帳戶中指定的 。多區域金鑰複寫中的來源金鑰稱為主要區域金鑰 (PRK)，這仍然是所有金鑰管理活動的授權來源。複寫的金鑰稱為複本區域金鑰 (RRK)，這是 PRK 的唯讀複本。您應該考慮將多區域金鑰與金鑰搭配使用，以滿足可用性、災難復原和低延遲的設計目標。

主題

- 建立金鑰
- 列出金鑰
- 啟用和停用 金鑰
- 複寫 AWS 付款密碼編譯金鑰
- 刪除 金鑰
- 匯入和匯出金鑰
- 使用別名
- 取得金鑰
- 標記金鑰
- 了解 AWS 付款密碼編譯金鑰的金鑰屬性

建立金鑰

您可以使用 CreateKey API 操作建立 AWS 付款密碼編譯金鑰。當您建立金鑰時，您可以指定屬性，例如金鑰演算法、金鑰用量、允許的操作，以及是否可以匯出。您無法在建立 AWS 付款密碼編譯金鑰後變更這些屬性。

> **Note**
>
> 如果 已啟用多區域金鑰複寫， AWS 帳戶 且您建立付款密碼編譯金鑰，則此金鑰會自動成為主要區域金鑰 (PRK)。即使您未在 CreateKey命令中指定 `--replication-regions` 參數，仍會複寫 PRK。如需詳細資訊，請參閱多區域金鑰複寫的運作方式。

範例

- 建立 2KEY TDES 基礎衍生金鑰
- 為 CVV/CVV2 建立 2KEY TDES 金鑰
- 建立 HMAC 金鑰
- 建立 AES-256 金鑰
- 建立 PIN 加密金鑰 (PEK)
- 建立非對稱 (RSA) 金鑰
- 建立 PIN 驗證值 (PVV) 金鑰
- 建立非對稱 ECC 金鑰

建立 2KEY TDES 基礎衍生金鑰

> **Note**
>
> 雖然 KeyUsage = TR31_B0_BASE_DERIVATION_KEY 支援 3KEY TDES，但目前沒有操作支援此金鑰類型。ANSI X9.24 標準 (DUKPT) 僅定義對 TDES_2KEY 和 AES 金鑰類型的支援。

Example

此命令會為 DUKPT 建立 2KEY TDES 衍生金鑰，該金鑰將從美國西部（奧勒岡）複寫到美國東部（俄亥俄）區域。回應包含請求參數、後續呼叫的 Amazon Resource Name (ARN)，以及金鑰檢查值 (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
```

輸出範例：

```
{
    "Key": {
        "CreateTimestamp": "2022-10-26T16:04:11.642000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/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"
}
```

為 CVV/CVV2 建立 2KEY TDES 金鑰

Example

此命令會建立 2KEY TDES 金鑰來產生和驗證 CVV/CVV2 值。回應包含請求參數、後續呼叫的 Amazon Resource Name (ARN)，以及金鑰檢查值 (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}'
```

輸出範例：

```
{
    "Key": {
        "CreateTimestamp": "2022-10-26T16:04:11.642000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_2KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": false,
                "Encrypt": false,
                "Generate": true,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
        },
        "KeyCheckValue": "AEA5CD",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-26T16:04:11.559000-07:00"
    }
}
```

建立 HMAC 金鑰

Example

HMAC 金鑰用於產生或驗證雜湊訊息驗證碼 (HMAC)。使用 HMAC 金鑰時，雜湊類型會在建立金鑰時指派（例如 HMAC_SHA224 和 HMAC_SHA512），且無法修改。

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

輸出範例：

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

建立 AES-256 金鑰

Example

此命令會建立用於資料加密和解密的 AES-256 對稱金鑰。AES 金鑰可為敏感資料提供強式加密，並常用於付款處理，以加密持卡人資料和其他敏感資訊，但 TDES 更常用於 EMV 等發行者使用案例。

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

輸出範例：

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

建立 PIN 加密金鑰 (PEK)

Example

此命令會建立 3KEY TDES 金鑰來加密 PIN 值，但 PIN 金鑰也可以是 AES，視您對互通性的需求而定。您可以使用此金鑰在驗證期間安全地存放 PINs 或解密 PINs，例如在交易中。回應包含請求參數、後續呼叫的 ARN，以及 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}'
```

輸出範例：

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T08:27:51.795000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY"
        },
        "KeyCheckValue": "7CC9E2",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
    }
}
```

建立非對稱 (RSA) 金鑰

Example

此命令會產生新的非對稱 RSA 2048 位元金鑰對。它會建立新的私有金鑰及其相符的公有金鑰。您可以使用 getPublicCertificate API 擷取公有金鑰。

```
$ 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}'
```

輸出範例：

```
{
    "Key": {
        "CreateTimestamp": "2022-11-15T11:15:42.358000-08:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
nsq2i3mbg6sn775f",
        "KeyAttributes": {
            "KeyAlgorithm": "RSA_2048",
            "KeyClass": "ASYMMETRIC_KEY_PAIR",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION"
        },
        "KeyCheckValue": "40AD487F",
        "KeyCheckValueAlgorithm": "SHA-1",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-11-15T11:15:42.182000-08:00"
    }
}
```

建立 PIN 驗證值 (PVV) 金鑰

Example

此命令會建立 3KEY TDES 金鑰來產生 PVV 值。您可以使用此金鑰來產生 PVV，以便與後續計算的 PVV 進行比較。回應包含請求參數、後續呼叫的 ARN，以及 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}'
```

輸出範例：

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T10:22:59.668000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": false,
                "Encrypt": false,
                "Generate": true,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY"
        },
        "KeyCheckValue": "7F2363",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-27T10:22:59.614000-07:00"
    }
}
```

建立非對稱 ECC 金鑰

Example

此命令會產生 ECC 金鑰對，用於在雙方之間建立 ECDH (Elliptic Curve Diffie-Hellman) 金鑰協議。使用 ECDH，每一方都會產生自己的 ECC 金鑰對，其中包含金鑰用途 K3 和使用模式 X，並交換公有金鑰。雙方接著會使用其私有金鑰和收到的公有金鑰來建立共用衍生金鑰。

若要在付款中維護密碼編譯金鑰的單次使用原則，建議您不要將 ECC 金鑰對重複使用，以進行多種用途，例如 ECDH 金鑰衍生和簽署。

```
$ 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}'
```

輸出範例：

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

列出金鑰

使用 ListKeys操作來取得您帳戶和區域中可存取的金鑰清單。

Example

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

輸出範例：

```
{
    "Keys": [
        {
            "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
            "Enabled": false,
            "Exportable": true,
            "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
            "KeyAttributes": {
                "KeyAlgorithm": "TDES_3KEY",
                "KeyClass": "SYMMETRIC_KEY",
                "KeyModesOfUse": {
                    "Decrypt": true,
                    "DeriveKey": false,
                    "Encrypt": true,
                    "Generate": false,
                    "NoRestrictions": false,
                    "Sign": false,
                    "Unwrap": true,
                    "Verify": false,
                    "Wrap": true
                },
                "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
            },
            "KeyCheckValue": "7F2363",
            "KeyCheckValueAlgorithm": "ANSI_X9_24",
            "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
            "KeyState": "CREATE_COMPLETE",
            "UsageStopTimestamp": "2022-10-27T14:19:42.488000-07:00"
        }
    ]
}
```

啟用和停用 金鑰

您可以停用並重新啟用 AWS 付款密碼編譯金鑰。當您建立金鑰時，預設會啟用該金鑰。如果您停用金鑰，則在您重新啟用之前，無法在任何密碼編譯操作中使用它。啟動/停止用量命令會立即生效，因此建議您在進行此類變更之前檢閱用量。您也可以使用選用timestamp參數，將變更（開始或停止用量）設定為在未來生效。

由於它是暫時且容易復原的，因此停用 AWS 付款密碼編譯金鑰是刪除 AWS 付款密碼編譯金鑰更安全的替代方案，這是具有破壞性和不可復原性的動作。如果您考慮刪除 AWS 付款密碼編譯金鑰，請先將其停用，並確保未來不需要使用該金鑰來加密或解密資料。

主題

- 啟動金鑰用量
- 停止金鑰用量

啟動金鑰用量

必須啟用金鑰用量，才能將金鑰用於密碼編譯操作。如果未啟用金鑰，您可以使用此操作來使其可供使用。欄位UsageStartTimestamp將代表金鑰何時變成/將變成作用中。對於已啟用的權杖，這將是過去，如果是待啟用，則未來會是過去。

Example

在此範例中，系統會請求針對金鑰用量啟用金鑰。回應包含金鑰資訊，且啟用旗標已轉換為 true。這也會反映在 list-keys 回應物件中。

```
$ 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"
  }
}
```

停止金鑰用量

如果您不再計劃使用金鑰，您可以停止金鑰使用，以防止進一步的密碼編譯操作。此操作不是永久的，因此您可以使用啟動金鑰用量將其反轉。您也可以將金鑰設定為未來停用。欄位UsageStopTimestamp代表金鑰何時變成/將變成停用。

Example

在此範例中，系統會要求 在未來停止金鑰使用。執行之後，除非透過啟動金鑰用量重新啟用，否則此金鑰無法用於密碼編譯操作。回應包含金鑰資訊和啟用旗標已轉換為 false。這也會反映在 list-keys 回應物件中。

```
$ 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"
  }
}
```

複寫 AWS 付款密碼編譯金鑰

AWS 付款密碼編譯支援多區域金鑰複寫，可讓您將金鑰資料和中繼資料從任何指定的 AWS 付款密碼編譯金鑰安全地分發到 AWS 區域 相同 AWS 分割區和帳戶中的一或多個 。

來源金鑰稱為主要區域金鑰 (PRK)，並且仍然是所有金鑰管理活動的授權來源，而 PRK 和複本區域金鑰 (RRK) 都可以用於其各自的密碼編譯操作 AWS 區域。

多區域金鑰複寫的優點

以下概述多區域金鑰複寫的一些優點。

- 更輕鬆地設定高可用性的應用程式 - AWS 付款密碼編譯會為您處理金鑰分佈，因此您可以在多個 AWS 區域 中使用金鑰，而不需要建立指定金鑰的解耦副本。
- 高可用性和低延遲金鑰 - 透過多區域金鑰複寫，您可以在多個 中存取金鑰 AWS 區域 ，使其具有高可用性，進而降低延遲。
- 金鑰材料耐久性 - 複本區域金鑰是完整的金鑰複本，可在密碼編譯操作中獨立於其主要區域金鑰使用。當 PRK 發生災難性資料遺失時，RRK 會提供耐用的複本。

多區域金鑰複寫的運作方式

啟用多區域金鑰複寫時， AWS 付款密碼編譯服務會使用安全金鑰分佈機制，將金鑰資料和中繼資料複製到 AWS 區域 您指定的複本。主要區域金鑰中繼資料的變更，例如金鑰屬性、狀態和啟用，會自動複寫至複本區域金鑰。

限制及考量

以下是一些多區域金鑰複寫限制和考量事項。

- 付款密碼編譯金鑰必須設定為可匯出，才能使用多區域金鑰複寫。
- 您必須為 AWS 區域 或特定付款密碼編譯金鑰啟用此功能。
  - 如果為 啟用此功能 AWS 區域，則啟用後建立的所有 AWS 付款密碼編譯金鑰都會複寫到指定的 AWS 區域。在此區域中建立的金鑰將成為主要區域金鑰。此區域中的現有金鑰不會自動複寫。您可以在金鑰層級為 AWS 區域 內的現有金鑰啟用多區域金鑰複寫。
  - 每個 AWS 區域 都可以有唯一的多區域金鑰複寫設定。
  - 金鑰的多區域複寫設定優先於 AWS 區域 多區域金鑰複寫設定。

- 複本區域金鑰無法設定為複寫至其他 AWS 區域。
- 多區域金鑰複寫適用於對稱付款密碼編譯金鑰，例如三重資料加密標準 (3DES)、進階加密標準 (AES) 和雜湊型訊息驗證碼 (HMAC)，以及非對稱付款密碼編譯金鑰，例如 RSA 和橢圓曲線密碼編譯 (ECC)。
- 複本區域金鑰是唯讀金鑰。主要區域金鑰的所有變更都會套用至複本區域金鑰。
- 主要區域金鑰變更最終會與複本區域金鑰一致。
- 付款密碼編譯金鑰只能使用相同的 AWS 分割區和帳戶複寫。
- 複本區域金鑰會計入您的 AWS 帳戶 層級 AWS 付款密碼編譯限制。
- 主要區域金鑰和複本區域金鑰使用相同的金鑰識別符，可讓您透過 IAM 政策中的相同 ARN 來參考這兩個金鑰。
- 您必須在複本 AWS 區域 中具有複寫`CreateKey`許可才能成功。

啟用多區域金鑰複寫

有兩種方式可以為 AWS 付款密碼編譯金鑰啟用多區域金鑰複寫。

1. AWS 區域：啟用 AWS 區域 時，多區域金鑰複寫會套用至在該 中建立的所有新金鑰。此方法為所有金鑰提供一致的複寫。
2. 特定 AWS 付款密碼編譯金鑰：您可以管理個別金鑰的多區域金鑰複寫，以允許更精細的控制層級。

啟用多區域金鑰複寫後，您的付款密碼編譯金鑰將複寫到 AWS 區域 您指定的 。

> **Important**
>
> 多區域金鑰複寫無法暫停。啟用複寫後，您的金鑰會自動複寫到 AWS 區域 您指定的 。您可以針對特定 AWS 區域 或付款密碼編譯金鑰停用多區域金鑰複寫。您必須從主要區域金鑰中移除 AWS 區域 做為複寫區域，才能刪除複本區域金鑰。
> 或者，您可以呼叫 PRK stop-key-usage 上的 `StopKeyUsage` API 或 CLI 命令，以停止同時使用 PRK 和所有相關聯的 RRKs。您無法在密碼編譯操作中使用這些金鑰。使用 `StopKeyUsage` API 或 stop-key-usage CLI 命令不會停止為 PRK 啟用的進行中多區域金鑰複寫。

您可以呼叫 GetDefaultKeyReplicationRegions API 或 CLI get-default-key-replication-regions 命令，檢查特定 AWS 區域 中 AWS 付款密碼編譯金鑰的多區域金鑰複寫設定。您呼叫此 API 動作或命令 AWS 區域 之 中的金鑰將成為您的 PRK。

使用下列程序來啟用多區域金鑰複寫。

For AWS 區域

- 使用以下命令為您 AWS 區域 指定的 啟用多區域金鑰複寫。在此範例中，在美國東部（俄亥俄）和美國西部（奧勒岡）啟用多區域金鑰複寫。若要使用此命令，請以您自己的資訊取代範例命令中的########。

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

> **Note**
>
> 為 啟用多區域金鑰複寫 AWS 區域 不會變更任何現有 AWS 付款密碼編譯金鑰的複寫組態。您可以在金鑰層級為現有金鑰啟用此功能。只有為 啟用多區域金鑰複寫之後建立的金鑰 AWS 區域 ，才會使用區域複寫設定。

For specific AWS Payment Cryptography keys

- 使用下列命令來啟用特定付款密碼編譯金鑰的多區域金鑰複寫。在此範例中，美國東部（俄亥俄）啟用多區域金鑰複寫。若要使用此命令，請將範例命令中的########取代為您自己的資訊。

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

或者，您可以在建立金鑰請求中包含複寫，以啟用此功能建立新的付款密碼編譯金鑰。 AWS 區域

> **Note**
>
> 金鑰複寫設定優先於 AWS 區域 複寫設定。

停用多區域金鑰複寫

如果您想要停用多區域金鑰複寫，您可以呼叫 disable-default-key-replication或 remove-key-replication-regions CLI 命令，具體取決於多區域金鑰複寫的啟用方式。您需要指定金鑰的 ARN 和 AWS 區域 ，以停用多區域金鑰複寫。

考量事項

複寫區域金鑰刪除最終一致。

您可以呼叫 `GetDefaultKeyReplicationRegions` API 或 CLI get-default-key-replication-regions 命令，檢查特定 AWS 區域 中 AWS 付款密碼編譯金鑰的多區域金鑰複寫設定。

使用下列程序來停用多區域金鑰複寫。

For AWS 區域

- 使用以下命令為您指定的 停用多區域金鑰複寫 AWS 區域 。在此範例中，美國東部 （俄亥俄） 停用多區域金鑰複寫。若要使用此命令，請以您自己的資訊取代範例命令中的*#######*。

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

For specific AWS Payment Cryptography keys

- 使用下列命令來停用特定付款密碼編譯金鑰的多區域金鑰複寫。在此範例中，多區域金鑰複寫正在美國東部 （俄亥俄） 停用。若要使用此命令，請以您自己的資訊取代範例命令中的*########*。

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

安全考量

以下是對付款密碼編譯金鑰使用多區域金鑰複寫時的安全考量。如需詳細資訊，請參閱AWS 付款密碼編譯的安全最佳實務。

- 限制共用金鑰資料。
- 建立 IAM 政策時，請遵循最低權限許可的主體。
- 您無法變更複本區域金鑰，因為它是唯讀金鑰。

最佳實務

以下是將多區域金鑰複寫與 AWS 付款密碼編譯金鑰搭配使用時的一些最佳實務。

- 確保您的應用程式持續運作，即使多區域金鑰複寫到指定的 AWS 區域 不是立即的。如果您需要知道多區域金鑰複寫何時完成，您可以使用 GetKey API 動作來監控 。您可以使用 監控金鑰複寫事件AWS CloudTrail。
- 從一個 容錯移轉 AWS 區域 到另一個 區域時，測試並實作自動化部署程序。

定價

您需要為使用 AWS 付款密碼編譯建立的複本區域金鑰付費。這些金鑰按 收費 AWS 區域。如需最新的付款密碼編譯定價資訊，請參閱AWS 付款密碼編譯定價頁面。

刪除 金鑰

刪除 AWS 付款密碼編譯金鑰會刪除金鑰材料和與金鑰相關聯的所有中繼資料，除非金鑰複本可在 AWS 付款密碼編譯外部取得，否則將無法復原。刪除金鑰後，您無法再解密在該金鑰下加密的資料，這表示資料可能無法復原。只有當您確定不再需要使用金鑰，而且沒有其他方正在使用此金鑰時，才應該刪除金鑰。如果您不確定，請考慮停止金鑰用量，而不是將其刪除。如果您稍後需要再次使用已停用的金鑰，您可以重新啟用該金鑰，但除非您能夠從其他來源重新匯入，否則無法復原已刪除的 AWS 付款密碼編譯金鑰。

在刪除金鑰之前，您應該確保不再需要金鑰。 AWS 付款密碼編譯不會儲存 CVV2 等密碼編譯操作的結果，也無法判斷任何持久性密碼編譯資料是否需要金鑰。

AWS 除非您明確排定刪除金鑰，且強制等待期間過期，否則付款密碼編譯永遠不會刪除屬於作用中 AWS 帳戶的金鑰。

不過，由於下列一個或多個原因，您可以選擇刪除 AWS 付款密碼編譯金鑰：

- 為不再需要的金鑰完成金鑰生命週期
- 為了避免與維護未使用的 AWS 付款密碼編譯金鑰相關聯的管理開銷

> **Note**
>
> 如果您關閉或刪除 AWS 帳戶，則無法存取您的 AWS 付款密碼編譯金鑰。您不需要將 AWS 付款密碼編譯金鑰與關閉帳戶分開排程刪除。

AWS 當您排程刪除付款密碼編譯金鑰，以及實際刪除 AWS 付款密碼 AWS 編譯金鑰時，付款密碼編譯會在AWS CloudTrail日誌中記錄項目。

使用多區域金鑰複寫時，刪除主要區域金鑰 (PRK) 的付款密碼編譯金鑰，複本區域金鑰 (RRK) 也會自動刪除。RRK 無法像 PRK 一樣刪除。如果您想要刪除 RRK，則需要修改 PRK 的複寫區域。

關於等待期

由於刪除金鑰是不可復原的， AWS 因此付款密碼編譯需要您將等待期間設定為 3-180 天。預設等待期間為七天。

不過，實際等待期可能比您排定的等待期長最多 24 小時。若要取得要刪除 AWS 付款密碼編譯金鑰的實際日期和時間，請使用 GetKey 操作。請務必注意時區。

在等待期間， AWS 付款密碼編譯金鑰狀態和金鑰狀態為待刪除。

> **Note**
>
> 等待刪除的 AWS 付款密碼編譯金鑰不能用於任何密碼編譯操作。

等待期間結束後， AWS 付款密碼編譯會刪除 AWS 付款密碼編譯金鑰、其別名和所有相關 AWS 付款密碼編譯中繼資料。

使用等待期間來確保您現在或未來不需要 AWS 付款密碼編譯金鑰。如果您發現在等待期間確實需要金鑰，您可以在等待期間結束前取消刪除金鑰。在等待期間結束後，您無法取消刪除金鑰，且服務會刪除金鑰。

Example

在此範例中，系統會請求刪除金鑰。除了基本金鑰資訊之外，兩個相關欄位是金鑰狀態已變更為 DELETE_PENDING，而 deletePendingTimestamp 代表目前排程刪除金鑰的時間。

```
$ 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

在此範例中，取消待定刪除。一旦成功完成，就不會再根據先前的排程刪除金鑰。回應包含基本金鑰資訊；此外，兩個相關欄位已變更 - KeyState和 deletePendingTimestamp。DeletePendingTimestamp當 移除時， KeyState 會傳回 CREATE_COMPLETE 的值。

```
$ 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"
    }
}
```

匯入和匯出金鑰

您可以從其他解決方案匯入 AWS 付款密碼編譯金鑰，並將其匯出至其他解決方案，例如 HSMs。許多客戶使用匯入和匯出功能與服務供應商交換金鑰。我們設計 AWS 了付款密碼編譯，以使用現代化的電子方法來管理金鑰，協助您維持合規和控制。我們建議您使用標準型電子金鑰交換，而非紙質型金鑰元件。如果您需要繼續處理紙質金鑰元件，直到所有合作夥伴都支援電子金鑰交換，您可以使用 實體金鑰交換。

最低金鑰強度和對匯入和匯出函數的影響

PCI 需要密碼編譯操作、金鑰儲存和金鑰傳輸的特定最低金鑰強度。修改 PCI 標準時，這些要求可能會變更。規則指定用於儲存或傳輸的包裝金鑰必須至少與受保護的金鑰一樣強。我們在匯出期間自動強制執行此要求，並防止金鑰受到較弱金鑰的保護，如下表所示。

下表顯示支援的包裝金鑰、要保護的金鑰和保護方法組合。

	包裝金鑰											
要保護的金鑰	TDES	TDES	AES_	AES_	AES_	RSA_	RSA_	RSA_	ECC_	ECC_	ECC_	備註
TDES_2KEY	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	ECDH	ECDH	ECDH	
TDES_3KEY	✗ 不支援	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	TR-3	ECDH	ECDH	ECDH	
AES_128	✗ 不支援	✗ 不支援	TR-3	TR-3	TR-3	✗ 不支援	TR-3	TR-3	ECDH	ECDH	ECDH	
AES_192	✗ 不支援	✗ 不支援	✗ 不支援	TR-3	TR-3	✗ 不支援	✗ 不支援	✗ 不支援	✗ 不支援	ECDH	ECDH	

	包裝金鑰											
要保護的金鑰	TDES	TDES	AES_	AES_	AES_	RSA_	RSA_	RSA_	ECC_	ECC_	ECC_	備註
AES_256	✗ 不支援	✗ 不支援	✗ 不支援	✗ 不支援	TR-3	✗ 不支援	✗ 不支援	✗ 不支援	✗ 不支援	✗ 不支援	ECDH	

如需詳細資訊，請參閱 PCI HSM 標準中的附錄 D - 已核准演算法的最小和同等金鑰大小和強度。

金鑰加密金鑰 (KEK) 交換

建議使用 ANSI X9.24 TR-34 標準。此初始金鑰類型可以稱為金鑰加密金鑰 (KEK)、區域主金鑰 (ZMK) 或區域控制主金鑰 (ZCMK)。如果您的系統或合作夥伴尚未支援 TR-34，您可以使用 RSA Wrap/Unwrap。如果您的需求包括交換 AES-256 金鑰，您可以使用 ECDH。

> **Note**
>
> 若要匯入您自己的測試金鑰或將金鑰與現有的 HSMs 同步，請參閱 GitHub 上的 AWS 付款密碼編譯範例程式碼。

工作金鑰 (WK) 交換

我們使用業界標準 (ANSI X9.24 TR 31-2018 和 X9.143) 來交換工作金鑰。這需要您已使用 TR-34、RSA Wrap、ECDH 或類似結構描述交換 KEK。此方法符合一律以密碼編譯方式將金鑰材料繫結至其類型和用量的 PCI PIN 要求。工作金鑰包括取得者工作金鑰、發行者工作金鑰、BDK 和 IPEK。

主題

- 匯入金鑰
- 匯出金鑰
- 進階主題

匯入金鑰

> **Important**
>
> 範例需要最新版本的 AWS CLI V2。開始使用之前，請確定您已升級至最新版本。

內容

- 匯入金鑰簡介
- 匯入對稱金鑰
  - 使用非對稱技術匯入金鑰 (TR-34)
  - 使用非對稱技術 (ECDH) 匯入金鑰
  - 使用非對稱技術匯入金鑰 (RSA Unwrap)
  - 使用預先建立的金鑰交換金鑰 (TR-31) 匯入對稱金鑰
- 匯入非對稱 (RSA、ECC) 公有金鑰
  - 匯入 RSA 公有金鑰
  - 匯入 ECC 公有金鑰

匯入金鑰簡介

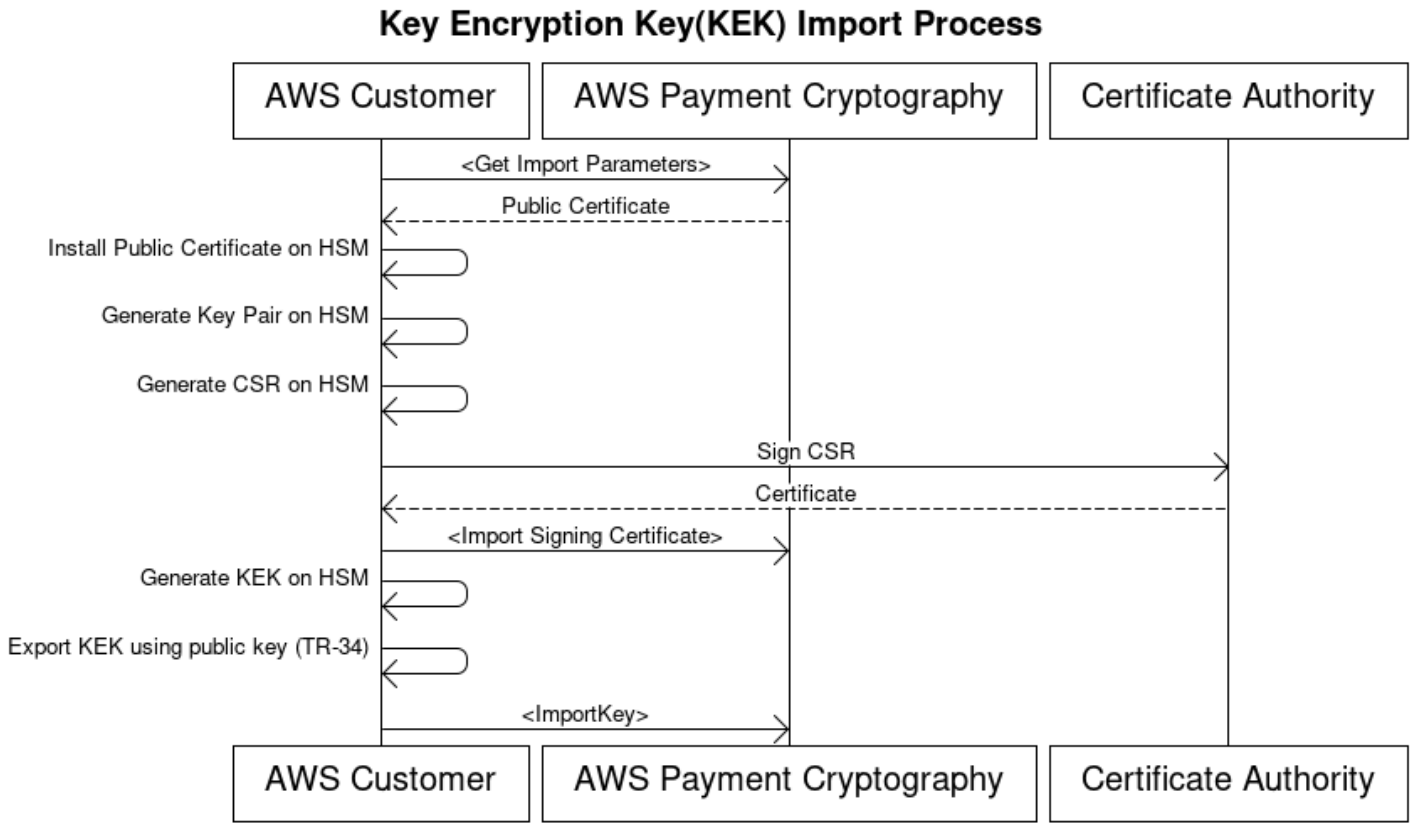
> **Note**
>
> 使用 X9.143、TR-31 或 TR-34 金鑰區塊匯入金鑰時， AWS 付款密碼編譯通常會保留 （但不使用） 任何選用的標頭。密碼編譯操作期間會使用 HM(HMAC 雜湊類型） 標頭。KP 標頭（包裝金鑰的 KCV) 專屬於匯入程序，不會保留。

與對手方交換金鑰時，通常會先交換金鑰交換金鑰 (KEK)。然後，此金鑰將用於保護後續金鑰。使用電子格式交換 KEK 可能會使用非對稱技術，例如 TR-34、ECDH 或 RSA 包裝。後續金鑰將使用對稱金鑰交換交換，例如 TR-31。此 KEK 將長期存在，並且只能根據政策及其定義的加密期間每幾年更新一次。

如果只交換一或兩個金鑰，您也可以選擇使用非對稱技術直接交換該金鑰，例如 BDK。 AWS 付款密碼編譯支援兩種金鑰交換方法。

匯入對稱金鑰

使用非對稱技術匯入金鑰 (TR-34)



TR-34 使用 RSA 非對稱密碼編譯來加密和簽署對稱金鑰以進行交換。這可確保包裝金鑰的機密性 （加密） 和完整性 （簽章）。

若要匯入您自己的金鑰，請查看 GitHub 上的 AWS 付款密碼編譯範例專案。如需有關如何從其他平台匯入/匯出金鑰的說明，範例程式碼可在 GitHub 上取得，或參閱這些平台的使用者指南。

1. 呼叫初始化匯入命令

   呼叫 `get-parameters-for-import` 以初始化匯入程序。此 API 會為金鑰匯入產生金鑰對、簽署金鑰，並傳回憑證和憑證根。使用此金鑰加密要匯出的金鑰。在 TR-34 術語中，這稱為 KRD Cert。這些憑證為 base64 編碼、短期，且僅用於此目的。儲存 `ImportToken`值。

   ```
   $ 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. 在金鑰來源系統上安裝公有憑證

   對於大多數 HSMs，您需要安裝、載入或信任步驟 1 中產生的公有憑證，才能使用它匯出金鑰。這可能包括整個憑證鏈，或僅包含步驟 1 的根憑證，視 HSM 而定。

3. 在來源系統上產生金鑰對，並將憑證鏈提供給 AWS 付款密碼編譯

   為了確保傳輸承載的完整性，傳送方 （ 金鑰分佈主機或 KDH) 會簽署它。為此目的產生公有金鑰，並建立公有金鑰憑證 AWS (X509) 以提供付款密碼編譯。

   從 HSM 傳輸金鑰時，請在該 HSM 上建立金鑰對。HSM、第三方或 等服務 AWS 私有 CA 可以產生憑證。

   使用 KeyMaterialType 為 `RootCertificatePublicKey`且 KeyUsageType 為 的 `importKey`命令，將根憑證載入 AWS 付款密碼編譯`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`。

   對於中繼憑證，請使用 `importKey`命令搭配 KeyMaterialType `TrustedCertificatePublicKey`和 KeyUsageType `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`。針對多個中繼憑證重複此程序。使用鏈結中最後一個匯入憑證`KeyArn`的 作為後續匯入命令的輸入。

   > **Note**
   >
   > 請勿匯入分葉憑證。在匯入命令期間直接提供。

4. 從來源系統匯出金鑰

   許多 HSMs和相關系統支援使用 TR-34 規範匯出金鑰。將步驟 1 的公有金鑰指定為 KRD （ 加密 ） 憑證，並將步驟 3 的金鑰指定為 KDH （ 簽署 ） 憑證。若要匯入 AWS 付款密碼編譯，請將格式指定為 TR-34.2012 非 CMS 兩種傳遞格式，也可以稱為 TR-34 Diebold 格式。

5. 呼叫匯入金鑰

使用 KeyMaterialType 呼叫 importKey APITR34_KEY_BLOCK。使用步驟 3 為 匯入的最後一個 CA 的 keyARNcertificate-authority-public-key-identifier、步驟 4 為 的包裝金鑰材料key-material，以及步驟 3 為 的分葉憑證signing-key-certificate。包含步驟 1 的 Import-token。

```
$ 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.
 \
    }'
```

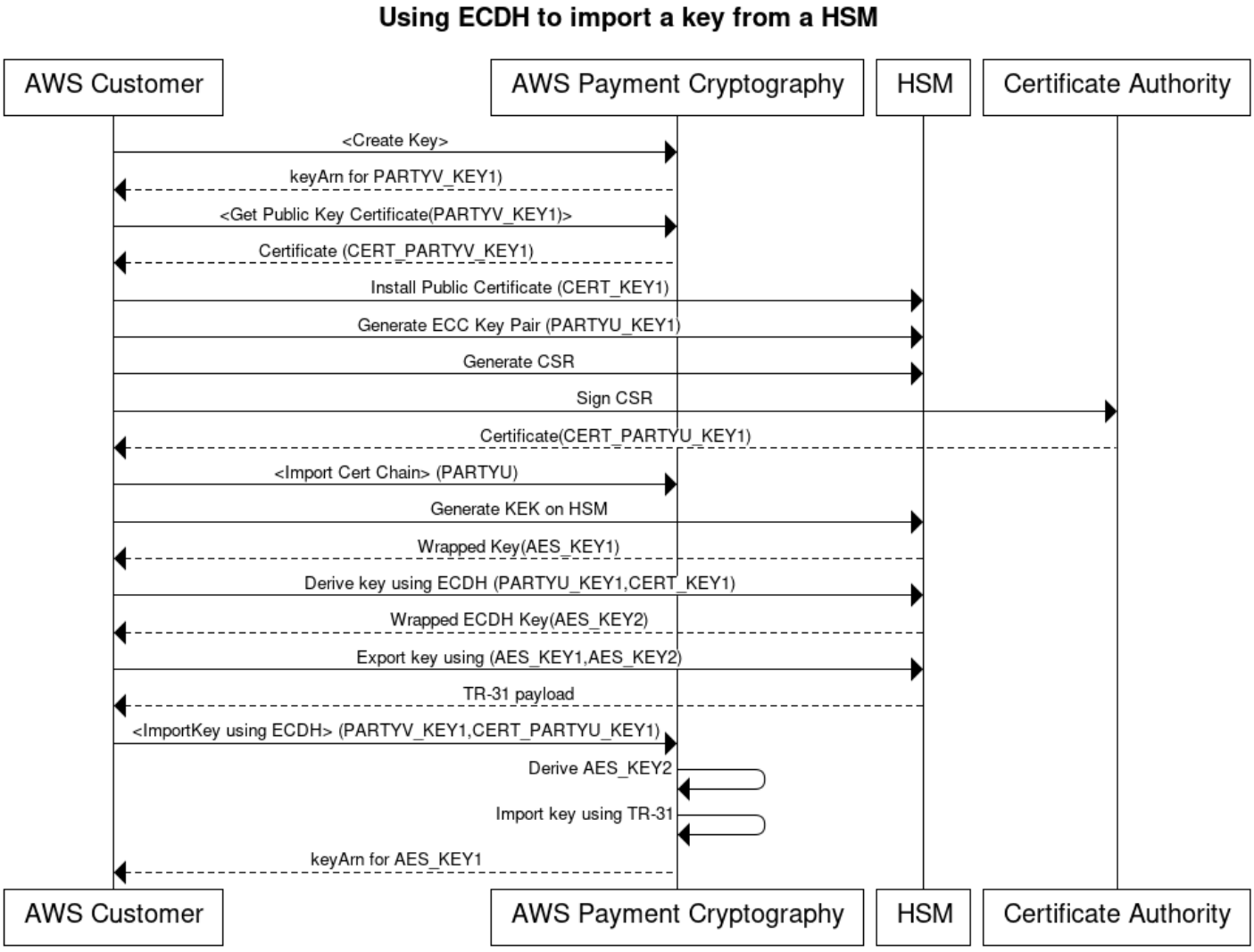
```
{
  "Key": {
    "CreateTimestamp": "2023-06-13T16:52:52.859000-04:00",
    "Enabled": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "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. 使用匯入的金鑰進行密碼編譯操作或後續匯入

   如果匯入的 KeyUsage 是 TR31_K0_KEY_ENCRYPTION_KEY，您可以使用此金鑰進行使用 TR-31 的後續金鑰匯入。對於其他金鑰類型 （例如 TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY)，您可以直接將金鑰用於密碼編譯操作。

使用非對稱技術 (ECDH) 匯入金鑰



橢圓曲線 Diffie-Hellman (ECDH) 使用 ECC 非對稱密碼編譯在兩方之間建立共用金鑰，而不需要預先交換的金鑰。ECDH 金鑰是暫時性的，因此 AWS 付款密碼編譯不會存放它們。在此程序中，使用 ECDH 衍生一次性 KBPK/KEK。該衍生金鑰會立即用於包裝您要傳輸的實際金鑰，這可能是另一個 KBPK、IPEK 金鑰或其他金鑰類型。

匯入時，傳送系統通常稱為 Party U （啟動器），而 AWS 付款密碼編譯稱為 Party V （回應者）。

> **Note**
>
> 雖然 ECDH 可用來交換任何對稱金鑰類型，但它是唯一可以安全地傳輸 AES-256 金鑰的方法。

1. 產生 ECC 金鑰對

   呼叫 `create-key` 來建立此程序的 ECC 金鑰對。此 API 會為金鑰匯入或匯出產生金鑰對。在建立時，指定可以使用此 ECC 金鑰衍生的金鑰類型。使用 ECDH 交換 （包裝） 其他金鑰時，請使用 的值`TR31_K1_KEY_BLOCK_PROTECTION_KEY`。

   > **Note**
   >
   > 雖然低階 ECDH 會產生可用於任何用途的衍生金鑰，但 AWS Payment Cryptography 會允許金鑰僅用於單一衍生金鑰類型，藉此限制金鑰意外重複使用多次用途。

```
$ 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. 取得公有金鑰憑證

   呼叫 `get-public-key-certificate` 以接收公有金鑰做為您帳戶的 CA 簽署的 X.509 憑證，該憑證是特定區域的 AWS 付款密碼編譯所特有。

   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. 在交易對手系統上安裝公有憑證 （第三方 U)

   許多 HSMs 都需要安裝、載入或信任步驟 1 中產生的公有憑證，才能使用它匯出金鑰。這可能包括整個憑證鏈，或僅包含步驟 1 的根憑證，視 HSM 而定。如需詳細資訊，請參閱您的 HSM 文件。

4. 在來源系統上產生 ECC 金鑰對，並提供 AWS 付款密碼編譯的憑證鏈

   在 ECDH 中，每一方都會產生金鑰對，並同意通用金鑰。若要讓 AWS 付款密碼編譯衍生金鑰，它需要 X.509 公有金鑰格式的對手方公有金鑰。

   從 HSM 傳輸金鑰時，請在該 HSM 上建立金鑰對。對於支援金鑰區塊HSMs，金鑰標頭看起來會與 類似`D0144K3EX00E0000`。建立憑證時，您通常會在 HSM 上產生 CSR，然後 HSM、第三方或 等服務 AWS 私有 CA 可以產生憑證。

   使用 KeyMaterialType 為 `RootCertificatePublicKey`且 KeyUsageType 為 的 `importKey`命令，將根憑證載入 AWS 付款密碼編譯`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`。

   對於中繼憑證，請使用 `importKey`命令搭配 KeyMaterialType `TrustedCertificatePublicKey`和 KeyUsageType `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`。針對多個中繼憑證重複此程序。使用鏈結中最後一個匯入憑證`KeyArn`的 作為後續匯入命令的輸入。

   > **Note**
   >
   > 請勿匯入分葉憑證。在匯入命令期間直接提供。

5. 在 U HSM 方上使用 ECDH 衍生一次性金鑰

   許多 HSMs和相關系統支援使用 ECDH 建立金鑰。將步驟 1 的公有金鑰指定為公有金鑰，並將步驟 3 的金鑰指定為私有金鑰。如需允許的選項，例如衍生方法，請參閱 API 指南。

   > **Note**
   >
   > 雜湊類型等衍生參數必須完全符合兩側。否則，您將產生不同的金鑰。

6. 從來源系統匯出金鑰

   最後，使用標準 TR-31 命令匯出您要傳輸到 AWS 付款密碼編譯的金鑰。將 ECDH 衍生金鑰指定為 KBPK。要匯出的金鑰可以是受 TR-31 有效組合約束的任何 TDES 或 AES 金鑰，只要包裝金鑰至少與要匯出的金鑰一樣強。

7. 呼叫匯入金鑰

呼叫 KeyMaterialType 為 的 import-key APIDiffieHellmanTr31KeyBlock。使用步驟 3 為匯入之最後一個 CA 的 KeyARNcertificate-authority-public-key-identifier、步驟 4 為 的包裝金鑰材料key-material，以及步驟 3 為 的分葉憑證public-key-certificate。包含步驟 1 中的私有金鑰 ARN。

```
$ 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. 使用匯入的金鑰進行密碼編譯操作或後續匯入

   如果匯入的 KeyUsage 是 TR31_K0_KEY_ENCRYPTION_KEY，您可以使用此金鑰進行使用 TR-31 的後續金鑰匯入。對於其他金鑰類型 （例如 TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY)，您可以直接將金鑰用於密碼編譯操作。

使用非對稱技術匯入金鑰 (RSA Unwrap)

Overview： AWS Payment Cryptography 支援 RSA wrap/unwrap，以便在 TR-34 不可行時交換金鑰。如同 TR-34，此技術使用 RSA 非對稱密碼編譯來加密對稱金鑰以進行交換。不過，與 TR-34 不同，此方法沒有傳送方簽署承載。此外，此 RSA 包裝技術不會在傳輸期間維持金鑰中繼資料的完整性，因為它不包含金鑰區塊。

> **Note**
>
> 您可以使用 RSA 包裝來匯入或匯出 TDES 和 AES-128 金鑰。

1. 呼叫初始化匯入命令

   呼叫 get-parameters-for-import 以使用 `KeyMaterialType`的 初始化匯入程序`KEY_CRYPTOGRAM`。在交換 TDES 金鑰`WrappingKeyAlgorithm`時使用 `RSA_2048` 。在交換 TDES `RSA_3072`或 AES-128 金鑰`RSA_4096`時使用 或 。此 API 會為金鑰匯入產生金鑰對，使用憑證根簽署金鑰，並同時傳回憑證和憑證根。使用此金鑰來加密要匯出的金鑰。這些憑證為短期憑證，僅供此用途使用。

```
$ 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. 在金鑰來源系統上安裝公有憑證

   對於許多 HSMs，您需要安裝、載入或信任步驟 1 中產生的公有憑證（和/或其根目錄），以使用它匯出金鑰。

3. 從來源系統匯出金鑰

   許多 HSMs和相關系統支援使用 RSA wrap 匯出金鑰。將步驟 1 的公有金鑰指定為加密憑證 (`WrappingKeyCertificate`)。如果您需要信任鏈，請使用步驟 1 `WrappingKeyCertificateChain`中的 。從 HSM 匯出金鑰時，指定格式為 RSA，填補模式 = PKCS#1 v2.2 OAEP（使用 SHA 256 或 SHA 512)。

4. 呼叫 import-key

   使用 `KeyMaterialType`的 呼叫 import-key API`KeyMaterial`。您需要步驟 1 `ImportToken`的和步驟 3 的 `key-material`（包裝金鑰材料）。提供金鑰參數（例如金鑰用量），因為 RSA 包裝不使用金鑰區塊。

```
$ 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
```

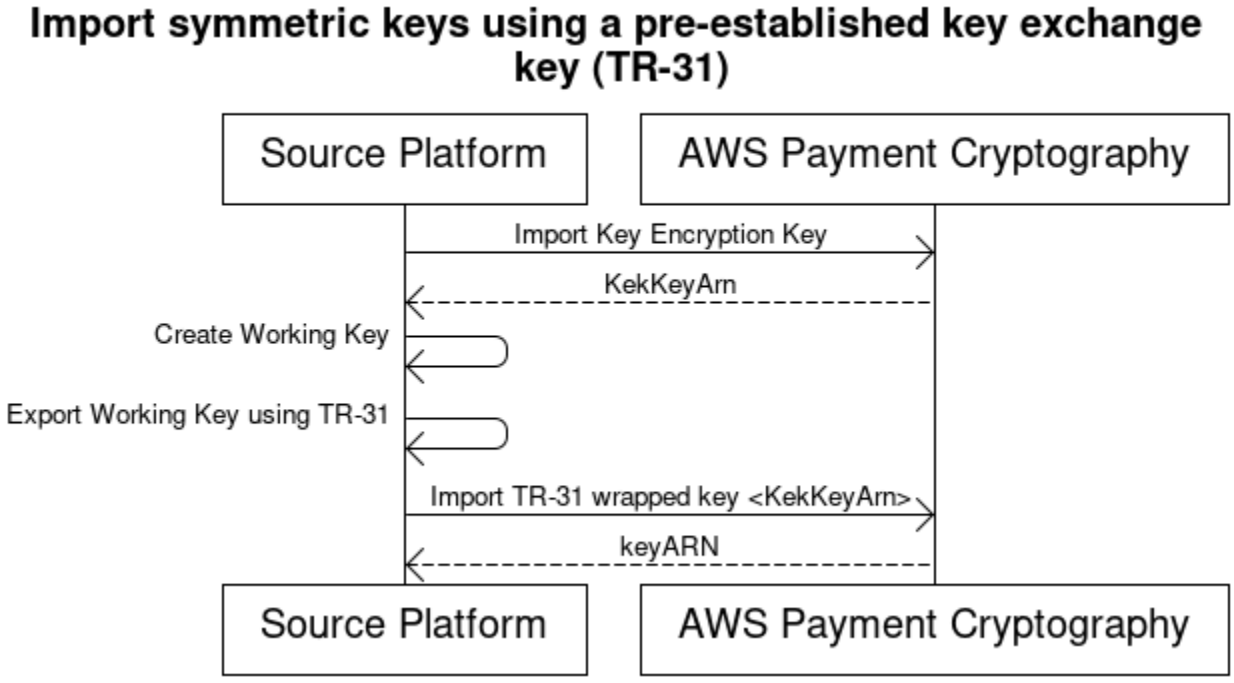
```
{
  "Key": {
    "KeyOrigin": "EXTERNAL",
    "Exportable": true,
    "KeyCheckValue": "DA1ACF",
    "UsageStartTimestamp": 1697643478.92,
    "Enabled": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
    "CreateTimestamp": 1697643478.92,
    "KeyState": "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. 使用匯入的金鑰進行密碼編譯操作或後續匯入

   如果匯入的 KeyUsage是 TR31_K0_KEY_ENCRYPTION_KEY或 TR31_K1_KEY_BLOCK_PROTECTION_KEY，您可以使用此金鑰來後續使用 TR-31 匯入金鑰。如果金鑰類型是任何其他類型 （ 例如 TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY)，您可以直接將金鑰用於密碼編譯操作。

使用預先建立的金鑰交換金鑰 (TR-31) 匯入對稱金鑰



交換多個金鑰或支援金鑰輪換時，合作夥伴通常會先交換初始金鑰加密金鑰 (KEK)。您可以使用 TR-34 或 等技術，將 KEK 與 AWS 付款密碼編譯交換實體金鑰交換。

建立 KEK 之後，您可以使用它來傳輸後續金鑰 （ 包括其他 KEKs)。 AWS 付款密碼編譯使用 ANSI TR-31 支援此金鑰交換，這是 HSM 供應商廣泛使用和支援的金鑰交換。

1. 匯入金鑰加密金鑰 (KEK)

請確定您已匯入 KEK，並且有可用的 keyARN （ 或 keyAlias)。

2. 在來源平台上建立金鑰

   如果金鑰不存在，請在來源平台上建立金鑰。或者，您可以在 AWS 付款密碼編譯上建立 金鑰，並使用 export命令。

3. 從來源平台匯出金鑰

   匯出時，請將匯出格式指定為 TR-31。來源平台將要求匯出金鑰和要使用的金鑰加密金鑰。

4. 匯入 AWS 付款密碼編譯

   呼叫 import-key命令時，請使用 金鑰加密金鑰的 keyARN （ 或別名）`WrappingKeyIdentifier`。使用 來源平台的輸出`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"
  }
}
```

匯入非對稱 (RSA、ECC) 公有金鑰

匯入的所有憑證必須至少與其在鏈中的發行（前身）憑證一樣強大。這表示 RSA_2048 CA 只能用於保護 RSA_2048 分葉憑證，且 ECC 憑證必須受到另一個同等強度的 ECC 憑證保護。ECC P384 憑證只能由 P384 或 P521 CA 發行。匯入時，所有憑證都必須未過期。

匯入 RSA 公有金鑰

AWS 付款密碼編譯支援將公有 RSA 金鑰匯入為 X.509 憑證。若要匯入憑證，請先匯入其根憑證。匯入時，所有憑證都必須未過期。憑證應該是 PEM 格式和 base64 編碼。

1. 將根憑證匯入 AWS 付款密碼編譯

   使用下列命令匯入根憑證：

Example

2. 將公有金鑰憑證匯入 AWS 付款密碼編譯

   您現在可以匯入公有金鑰。由於 TR-34 和 ECDH 依賴於在執行時間傳遞分葉憑證，此選項僅在使用來自另一個系統的公有金鑰加密資料時使用。KeyUsage 將設定為 TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION。

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"
  }
}
```

匯入 ECC 公有金鑰

AWS 付款密碼編譯支援將公有 ECC 金鑰匯入為 X.509 憑證。若要匯入憑證，請先匯入其根 CA 憑證和任何中繼憑證。匯入時，所有憑證都必須未過期。憑證應該是 PEM 格式和 base64 編碼。

1. 將 ECC 根憑證匯入 AWS 付款密碼編譯

   使用下列命令匯入根憑證：

Example

2. 將中繼憑證匯入 AWS 付款密碼編譯

   使用下列命令匯入中繼憑證：

Example

3. 將公有金鑰憑證（分葉）匯入 AWS 付款密碼編譯

   雖然您可以匯入分葉 ECC 憑證，但除了儲存之外，目前在 AWS 付款密碼編譯中沒有已定義的函數。這是因為在使用 ECDH 函數時，分葉憑證會在執行時間傳遞。

匯出金鑰

內容

- 匯出對稱金鑰
  - 使用非對稱技術匯出金鑰 (TR-34)
  - 使用非對稱技術 (ECDH) 匯出金鑰
  - 使用非對稱技術匯出金鑰 (RSA Wrap)
  - 使用預先建立的金鑰交換金鑰 (TR-31) 匯出對稱金鑰
- 匯出 DUKPT 初始金鑰 (IPEK/IK)
- 指定要匯出的金鑰區塊標頭
  - 常見標頭
- 匯出非對稱 (RSA) 金鑰

匯出對稱金鑰

> **Important**
>
> 開始 AWS CLI 之前，請確定您擁有最新版本的 。若要升級，請參閱安裝 AWS CLI。

使用非對稱技術匯出金鑰 (TR-34)

TR-34 使用 RSA 非對稱密碼編譯來加密和簽署對稱金鑰以進行交換。加密可保護機密性，而簽章可確保完整性。匯出金鑰時， AWS 付款密碼編譯會做為金鑰分佈主機 (KDH)，而您的目標系統會成為金鑰接收裝置 (KRD)。

> **Note**
>
> 如果您的 HSM 支援 TR-34 匯出，但不支援 TR-34 匯入，建議您先使用 TR-34 在 HSM 與 AWS 付款密碼編譯之間建立共用 KEK。然後，您可以使用 TR-31 來傳輸剩餘的金鑰。

1. 初始化匯出程序

   執行 get-parameters-for-export 為金鑰匯出產生金鑰對。我們使用此金鑰對來簽署 TR-34 承載。在 TR-34 術語中，這是 KDH 簽署憑證。憑證為短期且僅在 中指定的持續時間內有效ParametersValidUntilTimestamp。

   > **Note**
   >
   > 所有憑證都採用 base64 編碼。

   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. 將 AWS 付款密碼編譯憑證匯入您的接收系統

   從步驟 1 將憑證鏈匯入您的接收系統。

3. 設定接收系統的憑證

   為了保護傳輸的承載，傳送方 (KDH) 會對其進行加密。您的接收系統 （通常是您的 HSM 或合作夥伴的 HSM) 需要產生公有金鑰並建立 X.509 公有金鑰憑證。您可以使用 AWS 私有 CA 來產生憑證，但您可以使用任何憑證授權單位。

   取得憑證後，請使用 ImportKey命令將根憑證匯入 AWS 付款密碼編譯。將 KeyMaterialType 設定為 RootCertificatePublicKey，將 KeyUsageType 設定為 TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。

我們使用 TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE做為 ，KeyUsageType因為這是簽署分葉憑證的根金鑰。您不需要將分葉憑證匯入 AWS 付款密碼編譯 - 您可以內嵌傳遞憑證。

> **Note**
>
> 如果您先前已匯入根憑證，請略過此步驟。對於中繼憑證，請使用 TrustedCertificatePublicKey。

4. 匯出您的金鑰

呼叫 ExportKey API，並將 KeyMaterialType設定為 TR34_KEY_BLOCK。您需要提供：

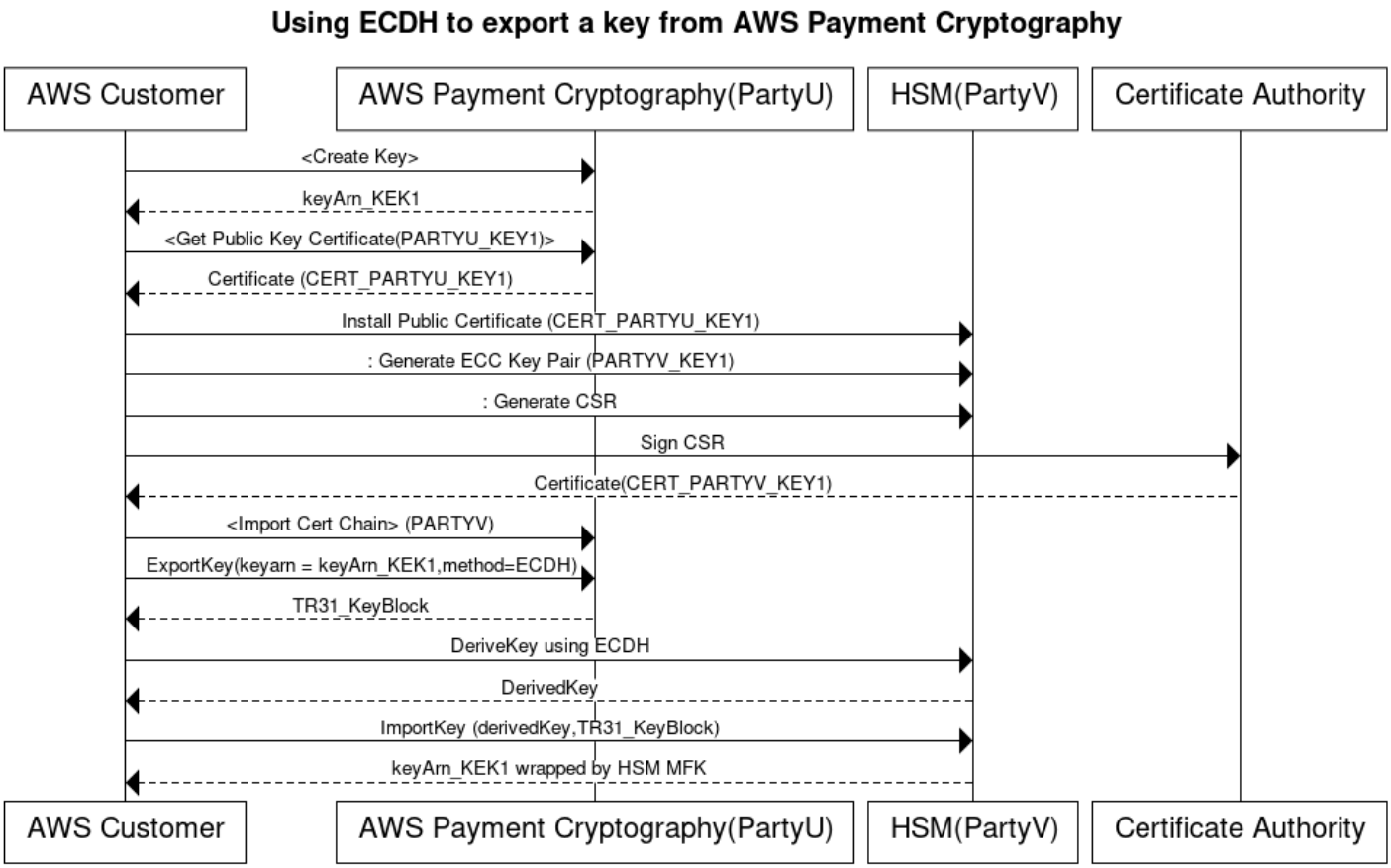
- 步驟 3 中根 CA 的 keyARN 做為 CertificateAuthorityPublicKeyIdentifier
- 步驟 3 的分葉憑證做為 WrappingKeyCertificate
- 您要匯出為 之金鑰的 keyARN （或別名） --export-key-identifier
- 步驟 1 的匯出權杖

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"
  }
}
```

使用非對稱技術 (ECDH) 匯出金鑰



橢圓曲線 Diffie-Hellman (ECDH) 使用 ECC 非對稱密碼編譯在兩方之間建立共用金鑰，而不需要預先交換的金鑰。ECDH 金鑰是暫時性的，因此 AWS 付款密碼編譯不會儲存它們。在此程序中，使用 ECDH 衍生一次性 KBPK/KEK。該衍生金鑰會立即用於包裝您要傳輸的金鑰，這可能是另一個 KBPK、BDK、IPEK 金鑰或其他金鑰類型。

匯出時，AWS 付款密碼編譯稱為 U 方（啟動者），而接收系統稱為 V 方（回應者）。

> **Note**
>
> ECDH 可用來交換任何對稱金鑰類型，但如果尚未建立 KEK，則是唯一可用於傳輸 AES-256 金鑰的方法。

1. 產生 ECC 金鑰對

呼叫 create-key 來建立此程序的 ECC 金鑰對。此 API 會為金鑰匯入或匯出產生金鑰對。在建立時，指定可以使用此 ECC 金鑰衍生的金鑰類型。使用 ECDH 交換 （包裝） 其他金鑰時，請使用 的值TR31_K1_KEY_BLOCK_PROTECTION_KEY。

> **Note**
>
> 雖然低階 ECDH 會產生可用於任何用途的衍生金鑰，但 AWS 付款密碼編譯會允許金鑰僅用於單一衍生金鑰類型，以限制金鑰意外重複使用用於多個用途。

```
$ 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. 取得公有金鑰憑證

   呼叫 get-public-key-certificate 以接收公有金鑰作為您帳戶的 CA 簽署的 X.509 憑證，該憑證是特定區域的 AWS 付款密碼編譯特有的。

   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. 在交易對手系統 （第三方 V) 上安裝公有憑證

   許多 HSMs 都需要安裝、載入或信任步驟 1 中產生的公有憑證，才能建立金鑰。這可能包括整個憑證鏈或僅根憑證，視 HSM 而定。如需特定指示，請參閱您的 HSM 文件。

4. 在來源系統上產生 ECC 金鑰對，並提供 AWS 付款密碼編譯的憑證鏈

   在 ECDH 中，每一方都會產生金鑰對，並同意通用金鑰。若要讓 AWS 付款密碼編譯衍生金鑰，它需要 X.509 公有金鑰格式的對手方公有金鑰。

   從 HSM 傳輸金鑰時，請在該 HSM 上建立金鑰對。對於支援金鑰區塊HSMs，金鑰標頭看起來會與 類似D0144K3EX00E0000。建立憑證時，您通常會在 HSM 上產生 CSR，然後 HSM、第三方或 等服務 AWS 私有 CA 可以產生憑證。

   使用 KeyMaterialType 為 RootCertificatePublicKey且 KeyUsageType 為 的 importKey命令，將根憑證載入 AWS 付款密碼編譯TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。

   對於中繼憑證，請使用 importKey命令搭配 KeyMaterialType TrustedCertificatePublicKey和 KeyUsageType

TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。針對多個中繼憑證重複此程序。使用鏈結中最後一個匯入憑證KeyArn的 作為後續匯出命令的輸入。

> **Note**
>
> 請勿匯入分葉憑證。在匯出命令期間直接提供。

5. 從 AWS 付款密碼編譯衍生金鑰和匯出金鑰

匯出時，服務會使用 ECDH 衍生金鑰，然後立即將其用作 KBPK，以包裝要使用 TR-31 匯出的金鑰。要匯出的金鑰可以是受 TR-31 有效組合約束的任何 TDES 或 AES 金鑰，只要包裝金鑰至少與要匯出的金鑰一樣強。

```
$ 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. 在 Party V HSM 上使用 ECDH 衍生一次性金鑰

   許多 HSMs和相關系統支援使用 ECDH 建立金鑰。將步驟 1 的公有金鑰指定為公有金鑰，並將步驟 3 的金鑰指定為私有金鑰。如需允許的選項，例如衍生方法，請參閱 API 指南。

   > **Note**
   >
   > 雜湊類型等衍生參數必須完全符合兩側。否則，您將產生不同的金鑰。

7. 將金鑰匯入目標系統

   最後，使用標準 TR-31 命令從 AWS 付款密碼編譯匯入金鑰。將 ECDH 衍生金鑰指定為 KBPK，並使用先前從 AWS 付款密碼編譯匯出的 TR-31 金鑰區塊。

使用非對稱技術匯出金鑰 (RSA Wrap)

當 TR-34 無法使用時，您可以使用 RSA wrap/unwrap 進行金鑰交換。如同 TR-34，此方法使用 RSA 非對稱密碼編譯來加密對稱金鑰。不過，RSA 包裝不包含：

- 傳送方簽署承載
- 在傳輸期間維護金鑰中繼資料完整性的金鑰區塊

> **Note**
>
> 您可以使用 RSA 包裝來匯出 TDES 和 AES-128 金鑰。

1. 在接收系統上建立 RSA 金鑰和憑證

   建立或識別用於接收包裝金鑰的 RSA 金鑰。我們需要 X.509 憑證格式的金鑰。請確定憑證由根憑證簽署，您可以將其匯入 AWS 付款密碼編譯。

2. 將根公有憑證匯入 AWS 付款密碼編譯

   使用 import-key搭配 `--key-material`選項來匯入憑證

```
$ 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. 匯出您的金鑰

   告知 AWS Payment Cryptography 使用您的分葉憑證匯出您的金鑰。您需要指定：

   - 您在步驟 2 匯入之根憑證的 ARN
   - 匯出的分葉憑證
   - 要匯出的對稱金鑰

輸出是對稱金鑰的十六進位編碼二進位包裝（加密）版本。

Example範例 – 匯出金鑰

```
$ 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. 將金鑰匯入您的接收系統

許多 HSMs和相關系統支援使用 RSA unwrap 匯入金鑰（包括 AWS 付款密碼編譯）。匯入時，請指定：

- 步驟 1 的公有金鑰做為加密憑證
- RSA 格式
- PKCS#1 v2.2 OAEP 的填補模式（使用 SHA 256)

> **Note**
>
> 我們輸出 hexBinary 格式的包裝金鑰。如果您的系統需要不同的二進位表示法，例如 base64，您可能需要轉換格式。

使用預先建立的金鑰交換金鑰 (TR-31) 匯出對稱金鑰

交換多個金鑰或支援金鑰輪換時，合作夥伴通常會先交換初始金鑰加密金鑰 (KEK)。您可以使用 TR-34 或 等技術，將 KEK 與 AWS 付款密碼編譯交換實體金鑰交換。建立 KEK 之後，您可以使用它來傳輸後續金鑰，包括其他 KEKs。我們支援使用 ANSI TR-31 進行此金鑰交換，HSM 廠商廣泛支援此金鑰交換。

1. 設定您的金鑰加密金鑰 (KEK)

   請確定您已交換 KEK，並且有可用的 keyARN （ 或 keyAlias)。

2. 在 AWS 付款密碼編譯上建立您的金鑰

   如果金鑰尚未存在，請建立金鑰。或者，您可以在其他系統上建立 金鑰，並使用匯入命令。

3. 從 AWS 付款密碼編譯匯出您的金鑰

   以 TR-31 格式匯出時，請指定您要匯出的金鑰和要使用的包裝金鑰。

Example範例 – 使用 TR31 金鑰區塊匯出金鑰

```
$ 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. 將金鑰匯入您的系統

   使用系統的匯入金鑰實作來匯入金鑰。

匯出 DUKPT 初始金鑰 (IPEK/IK)

使用 DUKPT 時，您可以為終端機機群產生單一基礎衍生金鑰 (BDK)。終端機無法直接存取 BDK。反之，每個終端機都會收到唯一的初始終端機金鑰，稱為 IPEK 或初始金鑰 (IK)。每個 IPEK 都是使用唯一的金鑰序號 (KSN) 衍生自 BDK。

KSN 結構因加密類型而異：

- 對於 TDES：10 位元組 KSN 包含：
  - 金鑰集 ID 的 24 位元
  - 終端機 ID 的 19 位元
  - 交易計數器的 21 位元
- 針對 AES：12 位元組 KSN 包含：
  - BDK ID 的 32 位元
  - 衍生識別符 (ID) 的 32 位元

- 交易計數器的 32 位元

我們提供一種機制來產生和匯出這些初始金鑰。您可以使用 TR-31、TR-34 或 RSA 包裝方法匯出產生的金鑰。請注意，IPEK 金鑰不會保留，也無法用於 AWS 付款密碼編譯的後續操作。

我們不會強制執行 KSN 前兩個部分之間的分割。如果您想要將衍生識別符與 BDK 一起存放，您可以使用 AWS 標籤。

> **Note**
>
> KSN 的計數器部分 (32 位元的 AES DUKPT) 不會用於 IPEK/IK 衍生。例如，12345678901234560001 和 12345678901234569999 的輸入會產生相同的 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"
}
}
```

指定要匯出的金鑰區塊標頭

您可以在以 ASC TR-31 或 TR-34 格式匯出時修改或附加金鑰區塊資訊。下表說明 TR-31 金鑰區塊格式，以及您可以在匯出期間修改哪些元素。

金鑰區塊屬性	用途	您可以在匯出期間修改 嗎？	備註
版本 ID	定義用於保護金鑰材料的方法。標準包括： • 版本 A 和 C （金鑰變體 - 已棄用） • B 版 （使用 TDES 衍生） • 版本 D （使用 AES 的金鑰衍生）	否	我們將版本 B 用於 TDES 包裝金鑰，並將版本 D 用於 AES 包裝金鑰。我們僅支援版本 A 和 C 的匯入操作。
金鑰區塊長度	指定剩餘訊息的長度	否	我們會自動計算此值。在解密承載之前，長度可能不正確，因為我們可能會根據規格的要求新增金鑰填補。
金鑰使用方式	定義金鑰的允許用途，例如： • C0 （卡驗證） • B0 （基本衍生金鑰）	否	
演算法	指定基礎金鑰的演算法。我們支援： • T (TDES) • H (HMAC) • A (AES)	否	我們會依原狀匯出此值。
金鑰使用方式	定義允許的操作，例如：	是*	

金鑰區塊屬性	用途	您可以在匯出期間修改 嗎？	備註
	• 產生並驗證 (C) • Encrypt/Decrypt/Wrap/Unwrap (B)		
金鑰版本	指出金鑰替換/輪換的版本編號。如果未指定，則預設為 00。	是 - 可以附加	
金鑰可匯出性	控制是否可以匯出金鑰： • N - 無可匯出性 • E - 根據 X9.24 匯出（索引鍵區塊） • S - 在金鑰區塊或非金鑰區塊格式下匯出	是*	
選用金鑰區塊	是 - 可以附加	選用的金鑰區塊是以密碼編譯方式繫結至金鑰的名稱/值對。例如，DUKPT 金鑰的 KeySetID。我們會根據您的名稱/值對輸入，自動計算區塊數量、每個區塊的長度和填補區塊 (PB)。	

*修改值時，您的新值必須比 AWS 付款密碼編譯中的目前值更嚴格。例如：

- 如果目前的金鑰使用模式是 Generate=True，Verify=True，您可以將其變更為 Generate=True，Verify=False
- 如果金鑰已設定為不可匯出，則您無法將其變更為可匯出

匯出金鑰時，我們會自動套用所匯出金鑰的目前值。不過，您可能想要在傳送至接收系統之前修改或附加這些值。以下是一些常見的案例：

- 匯出金鑰至付款終端機時，請將其可匯出性設定為 ，`Not Exportable`因為終端機通常只會匯入金鑰，且不應匯出金鑰。
- 當您需要將相關聯的金鑰中繼資料傳遞至接收系統時，請使用 TR-31 選用標頭，以密碼編譯方式將中繼資料繫結至金鑰，而不是建立自訂承載。
- 使用 `KeyVersion` 欄位設定金鑰版本來追蹤金鑰輪換。

TR-31/X9.143 定義了常見的標頭，但您可以使用其他標頭，只要它們符合 AWS 付款密碼編譯參數，而且您的接收系統可以接受它們。如需匯出期間金鑰區塊標頭的詳細資訊，請參閱 API 指南中的金鑰區塊標頭。

以下是使用下列規格匯出 BDK 金鑰 （ 例如，匯出至 KIF) 的範例：

- 金鑰版本：02
- KeyExportability：NON_EXPORTABLE
- KeySetID：00ABCDEFAB (00 表示 TDES 金鑰，ABCDEFABCD 是初始金鑰)

由於我們未指定金鑰使用模式，因此此金鑰會繼承 arn：aws：payment-cryptography：us-east-2：111122223333：key/5rplquuwozodpwsp (DeriveKey = true) 的使用模式。

> **Note**
>
> 即使您在此範例中將可匯出性設定為不可匯出，KIF 仍然可以：
>
> - 衍生金鑰，例如 DUKPT 中使用的 IPEK/IK
> - 匯出這些衍生金鑰以安裝在裝置上
>
> 這由標準特別允許。

```
$ 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"
    }
}
```

常見標頭

X9.143 定義常見使用案例的特定標頭。除了 HM(HMAC 雜湊） 標頭之外， AWS 付款密碼編譯不會剖析或利用這些標頭。

標頭名稱	用途	典型驗證	備註
BI	DUKPT 的基礎衍生金鑰識別符	2 個十六進位字元 (TDES 為 00，AES 為 11)，TDES KSI 為 10 個十六進位字元，BDK ID 為 8 個十六進位字元 (AES DUKPT)。	包含 (BDK ID，適用於 AES DUKPT) 或金鑰集識別符 (KSI，適用於 TDES DUKPT)。可在交換 BDK ID 或 KSI 時使用，但不需要交換 IK 和 KS 區塊中包含的其他資料。傳輸到 KIF 時通常會使用 BI，而注入終端機本身時則會使用 IK 或 KS。

標頭名稱	用途	典型驗證	備註
HM	指定 HMAC 操作的雜湊類型	• 10 – SHA-1 • 20 – SHA-224 • 21 – SHA-256 • 22 – SHA-384 • 23 – SHA-512 • 24 – SHA-512/224 • 25 – SHA-512/256 • 30 – SHA3-224 • 31 – SHA3-256 • 32 – SHA3-384 • 33 – SHA3-512 • 40 – SHAKE128 • 41 – SHAKE256	服務會在匯出時自動填入此欄位，並在匯入時剖析它。您可以匯入 SHAKE128 等服務不支援的雜湊類型，但可能無法用於密碼編譯函數。
IK	AES DUKPT 的初始金鑰序號	16 個十六進位字元	此值用於執行個體化接收裝置上初始 DUKPT 金鑰的使用，並識別衍生自 BDK 的初始金鑰。此欄位通常包含衍生資料，但沒有計數器。將 KS 用於 TDES DUKPT。
KS	TDES DUKPT 的初始金鑰序號	20 個十六進位字元	此值用於執行個體化接收裝置上初始 DUKPT 金鑰的使用，並識別衍生自 BDK 的初始金鑰。此欄位通常包含衍生資料 + 歸零的計數器值。使用 IK for AES DUKPT。

標頭名稱	用途	典型驗證	備註
KP	KBPK 的金鑰檢查值	2 個十六進位字元代表 KCV 方法 (X9.24 方法為 00，CMAC 方法為 01)。後面接著通常為 6 個十六進位字元的 KCV 值。例如，010FA329 代表使用 01(CMAC) 方法計算的 0FA329 KCV。	根據 ANSI X9.24-1-2017 Annex A 計算；KCV 用於金鑰驗證，而非完整性機制。
PB	填補區塊	隨機可列印的 ASCII 字元	服務會在匯出時自動填入此欄位，以確保選用標頭是加密區塊長度的倍數

匯出非對稱 (RSA) 金鑰

若要以憑證形式匯出公有金鑰，請使用 get-public-key-certificate命令。此命令會傳回：

- 憑證
- 根憑證

這兩個憑證都在 base64 編碼中。

> **Note**
>
> 此操作不等冪，即使使用相同的基礎金鑰，後續呼叫仍可能會產生不同的憑證。

Example

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

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

進階主題

本節涵蓋進階金鑰交換案例和組態。

主題

- 自攜憑證授權機構 (BYOCA)
- 實體金鑰交換

自攜憑證授權機構 (BYOCA)

根據預設，當服務內建立的非對稱 (RSA，ECC) 金鑰需要公有金鑰憑證時，這些憑證是由 AWS 付款密碼編譯和帳戶唯一憑證授權機構 (CA) 發行。這是為了讓您更輕鬆地使用 X.509，而無須負擔識別或設定 CA 或管理憑證簽署請求 (CSR) 的負擔。

AWS 付款密碼編譯也可在政策或合規原因需要時使用您自己的 CA。

主題

- 概觀
- BYOCA 匯出工作流程
- BYOCA 匯入工作流程
- 其他備註

概觀

透過 BYOCA，您可以使用自己的 CA 進行 TR-34 匯入/匯出、RSA Unwrap 和 ECDH 金鑰傳輸。當您需要整個組織的一致憑證鏈，或合作夥伴需要特定的 CA 憑證時，請使用 BYOCA。下列範例顯示 TR-34 金鑰匯出和 TR-34 金鑰匯入的 BYOCA 工作流程。

標準 TR-34 流程有三個主要差異：

1. 您可以使用 CreateKey 建立 RSA 金鑰。在 之前，`GetParametersForExport`或 為您`GetParametersForImport`建立它。
2. GetCertificateSigningRequest API 會建立 CSR。然後，您的外部 CA 可以簽署它。
3. ExportKey 和 ImportKey APIs呼叫時接受憑證。字符現在是選用的。

> **⚠ 重要考量**
>
> - 這些範例使用 RSA-2048 金鑰並包裝 TDES-2KEY 金鑰。匯出 AES-128 時，請確定所有金鑰都是 RSA-3072 或 RSA-4096。
> - 最常見的錯誤是 表示的金鑰`SigningKeyIdentifier`與 `SigningKeyCertificate` 不相符。

BYOCA 匯出工作流程

下列步驟示範 TR-34 匯出的完整 BYOCA 工作流程。

步驟

- 步驟 1：建立 RSA 金鑰
- 步驟 2：產生憑證簽署請求
- 步驟 3：檢閱 CSR （選用）
- 步驟 4：使用憑證授權單位簽署 CSR
- 步驟 5：匯入 CA 憑證
- 步驟 6：取得 KRD 加密憑證
- 步驟 7：使用 BYOCA 匯出金鑰

步驟 1：建立 RSA 金鑰

首先，建立最終將成為 KDH 簽署憑證的 RSA 金鑰對。您可以新增標籤來識別金鑰的用途。

Example建立用於簽署的 RSA 金鑰

```
$ 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"
    }
}
```

請記下 KeyArn，因為您將在下一個步驟中需要它。

步驟 2：產生憑證簽署請求

使用 GetCertificateSigningRequest API 產生由外部 CA 簽署的憑證簽署請求 (CSR)。輸出是 base64 編碼的 PEM 檔案。如果您 base64 解碼內容並儲存，您將擁有 PEM 格式的有效 CSR。

Example產生 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..."
}
```

`CertificateSigningRequest` 欄位包含整個 PEM 檔案 base64 編碼。PEM 檔案包含 `-----BEGIN CERTIFICATE REQUEST-----`和 `-----END CERTIFICATE REQUEST-----`標記。將此值傳送至您的 CA 以進行簽署。

步驟 3：檢閱 CSR （選用）

您可以選擇性地使用 OpenSSL 來檢閱 CSR 內容，並確保其有效且如預期。

Example使用 OpenSSL 檢閱 CSR

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

步驟 4：使用憑證授權單位簽署 CSR

產生 CSR 後，您需要由憑證授權機構 (CA) 簽署。在生產環境中，您通常會使用 AWS 私有 CA 或組織已建立的 CA 基礎設施。基於測試目的，您可以使用 OpenSSL 建立自我簽署憑證。

使用 AWS 私有 CA

若要使用 簽署 CSR AWS 私有 CA，請先解碼 base64 編碼的 CSR 並將其儲存至檔案，然後使用 IssueCertificate API。

Example使用 簽署 CSR AWS 私有 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"
}
```

然後擷取已簽章的憑證：

Example擷取已簽署憑證

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

儲存憑證內容以用於匯出步驟。將 base64 編碼提供給 `ExportKey` API 時，您將需要對其進行 Base64 編碼。

使用 OpenSSL 進行測試

基於測試目的，您可以使用 OpenSSL 建立自我簽署 CA 並簽署 CSR。首先，建立 CA 私有金鑰和自我簽署憑證：

Example使用 OpenSSL 建立測試 CA

```
$ # 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"
```

然後，從上一個步驟解碼 CSR，並使用測試 CA 簽署：

Example使用 OpenSSL 簽署 CSR

```
$ # 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
```

簽章的憑證現在位於 中signed-cert.pem。在將此憑證提供給 ExportKey API 時，您需要對憑證進行 base64 編碼：

Example Base64 編碼已簽署的憑證

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

步驟 5：匯入 CA 憑證

必須先信任使用中的任何 CA，以防止使用任意憑證。使用 ImportKey API 匯入外部 CA 的根憑證。如果使用中繼 CA，請import-key再次呼叫 ，但請指定 TrustedPublicKey而非 RootCertificatePublicKey，並指定根 CA ARN。

Example匯入根 CA 憑證

```
$ 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"
    }
}
```

請記下 CA KeyArn用於匯出步驟。

步驟 6：取得 KRD 加密憑證

在此範例中，我們將匯入回 AWS 付款密碼編譯，因此我們呼叫 服務，使用 GetParametersForImport API 接收 KRD 公有金鑰憑證。在實際情況下，這將由其他系統提供，例如 HSM、ATM、付款終端機或付款終端機管理系統。

Example取得用於匯入的參數

```
$ 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"
}
```

步驟 7：使用 BYOCA 匯出金鑰

最後，使用 ExportKey API 搭配您自己的 CA 簽署憑證使用 TR-34 匯出金鑰。 ExportKey 提供外部 CA 簽署的簽署憑證。

Example使用 BYOCA 匯出 TR-34

```
$ 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"
    }
}
```

匯出的金鑰區塊現在可以由接收系統使用標準 TR-34 匯入程序來匯入。

BYOCA 匯入工作流程

下列步驟會逐步引導您完成 TR-34 金鑰匯入的 BYOCA 工作流程。使用此工作流程，透過您自己的 CA 簽署 KRD （金鑰接收裝置） 加密憑證，接收 AWS 付款密碼編譯中的金鑰。透過 BYOCA 匯入，您可以使用您已管理的 CA 來控制信任鏈。

步驟

- 步驟 1：建立 RSA 金鑰
- 步驟 2：產生憑證簽署請求
- 步驟 3：檢閱 CSR（選用）
- 步驟 4：使用憑證授權單位簽署 CSR
- 步驟 5：匯入 CA 憑證
- 步驟 6：從傳送系統取得 KDH 簽署憑證
- 步驟 7：使用 BYOCA 匯入金鑰

步驟 1：建立 RSA 金鑰

建立 RSA 金鑰對以用作 KRD 加密金鑰。此金鑰會在 TR-34 匯入期間取消包裝並解密傳入的金鑰。金鑰不可匯出，因為它會保留在服務內做為接收金鑰。

Example建立 KRD 加密金鑰

```
$ 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"
```

```
    }
}
```

請記下 KeyArn 值。下一個步驟需要此值。

步驟 2：產生憑證簽署請求

使用 GetCertificateSigningRequest API 產生 KRD 加密金鑰的憑證簽署請求 (CSR)。輸出是 base64 編碼的 PEM 檔案。如果您 base64 解碼內容並儲存，您將擁有 PEM 格式的有效 CSR。

Example產生 KRD 金鑰的 CSR

```
$ 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..."
}
```

CertificateSigningRequest 欄位包含整個 PEM 檔案 base64 編碼。PEM 檔案包含 -----BEGIN CERTIFICATE REQUEST-----和 -----END CERTIFICATE REQUEST-----標記。將此值傳送至您的 CA 以進行簽署。

步驟 3：檢閱 CSR （選用）

您可以選擇性地使用 OpenSSL 來檢閱 CSR 內容，並確保其有效且如預期。

Example使用 OpenSSL 檢閱 CSR

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

步驟 4：使用憑證授權單位簽署 CSR

產生 CSR 後，請由憑證授權機構 (CA) 簽署。在生產環境中，您通常會使用 AWS 私有 CA 或組織已建立的 CA 基礎設施。基於測試目的，您可以使用 OpenSSL 建立自我簽署憑證。

使用 AWS 私有 CA

若要使用 簽署 CSR AWS 私有 CA，請先解碼 base64 編碼的 CSR，並將其儲存至檔案，然後使用 IssueCertificate API。

Example使用 簽署 CSR AWS 私有 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"
}
```

然後擷取已簽章的憑證：

Example擷取已簽署憑證

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

```
}
```

儲存憑證內容以用於匯入步驟。將憑證提供給 ImportKey API 之前，先對憑證進行 Base64-encode。

使用 OpenSSL 進行測試

基於測試目的，您可以使用 OpenSSL 建立自我簽署 CA 並簽署 CSR。首先，建立 CA 私有金鑰和自我簽署憑證：

Example使用 OpenSSL 建立測試 CA

```
$ # 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"
```

然後，從上一個步驟解碼 CSR，並使用測試 CA 簽署：

Example使用 OpenSSL 簽署 CSR

```
$ # 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
```

簽章的憑證現在位於 中krd-signed-cert.pem。將此憑證提供給 ImportKey API 之前，先對 BaseBase64-encode：

Example Base64 編碼已簽署的憑證

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

步驟 5：匯入 CA 憑證

在使用 CA 之前，您必須先信任它。使用 ImportKey API 匯入外部 CA 的根憑證。如果使用中繼 CA，請import-key再次呼叫 ，但請指定 TrustedPublicKey而非 RootCertificatePublicKey，並指定根 CA ARN。

Example匯入根 CA 憑證

```
$ 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"
    }
}
```

請注意 CA KeyArn用於匯入步驟。

步驟 6：從傳送系統取得 KDH 簽署憑證

在 TR-34 金鑰交換中，傳送端 （金鑰分佈主機或 KDH) 提供其公有金鑰憑證以供簽署。此憑證用於在匯入期間驗證包裝金鑰區塊上的簽章，以確保金鑰材料來自信任的寄件者。在生產環境中，此憑證通常來自傳送系統，例如 HSM、付款終端機或金鑰管理系統，透過自己的憑證分佈程序。

在此範例中，您會使用 AWS 付款密碼編譯做為寄件者和接收者，讓步驟保持簡短。呼叫 GetParametersForExport 以取得傳送系統通常提供的 KDH 簽署憑證。

Example取得匯出的參數

```
$ 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"
}
```

請注意 ExportToken和 SigningKeyCertificate，以便在匯入步驟中使用。

步驟 7：使用 BYOCA 匯入金鑰

最後，使用 TR-34 匯入金鑰，並使用 ImportKey API 搭配您自己的 CA 簽署憑證。提供外部 CA 簽署的 KRD 加密憑證。

使用 2-pass TR-34 時需要 RandomNonce 參數。服務不會為您產生 nonce - 這是您產生的隨機十六進位值，而且在傳送端和接收端上必須相同。使用 1-pass TR-34 （依賴時間戳記而非 nonce) 時，時間戳記不得為未來的時間。服務不會驗證任何特定新鮮度 （例如，在特定時數內），但您可以視需要對您強制執行新鮮度檢查。

Example使用 BYOCA 匯入 TR-34

```
$ 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"
    }
}
```

金鑰現在會匯入 AWS 付款密碼編譯並可供使用。KeyOrigin 欄位會顯示 ，EXTERNAL指出金鑰來自服務外部。

其他備註

- 這些範例使用 AWS CLI 顯示。相同的功能適用於所有 AWS SDKs，包括 Java、Python、Go 和 Rust。
- 如果您使用自我簽署 CA 進行測試，您可以使用 OpenSSL 來建立測試 CA 並簽署 CSR。在生產環境中，使用您組織已建立的 CA 基礎設施。

實體金鑰交換

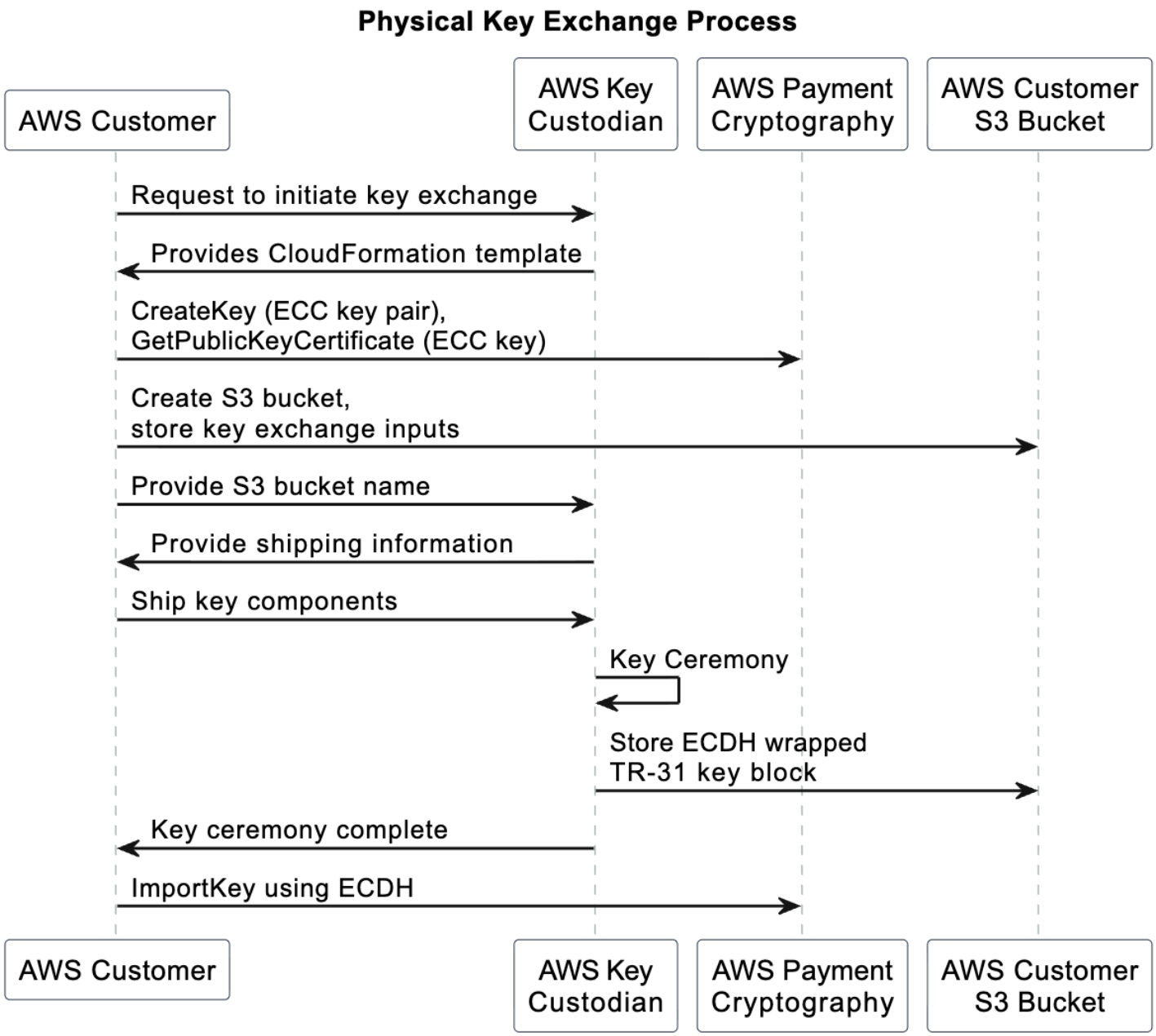
當您的合作夥伴或廠商不支援電子金鑰交換時，您可以使用實體金鑰交換將紙質型密碼編譯金鑰元件安全地轉換為電子格式。受過訓練的 AWS 金鑰監管人員會在 PCI PIN 和 P2PE 認證的 AWS 操作的安全設施中執行金鑰典禮，並使用離線 HSM 將紙質金鑰元件轉換為電子格式。此服務使用 ECDH 型金鑰交換來交付 ECDH 包裝的 TR-31 金鑰區塊，您可直接匯入您的 AWS 付款密碼帳戶。

> **Note**
>
> 我們建議您匯入和匯出金鑰盡可能使用標準型。只有在您的合作夥伴或廠商不支援 ANSI X9.24 TR-34、RSA wrap/unwrap 或 ECDH 等電子金鑰交換方法時，才使用實體金鑰交換。

實體金鑰交換的運作方式

若要啟動紙質金鑰交換，CloudFormation 範本會引導您完成先決條件設定，包括在帳戶中建立 ECC 金鑰對和 S3 儲存貯體。然後，您或您的合作夥伴將紙質金鑰元件運送到 AWS 安全設施，其中經過訓練的 AWS 金鑰監管人員會使用離線 HSM 執行金鑰典禮。輸出是上傳至 S3 儲存貯體的 ECDH 包裝 TR-31 金鑰區塊，您可以使用 使用非對稱技術 (ECDH) 匯入金鑰方法將其匯入您的帳戶。實體金鑰交換支援在 TDES 和 AES 金鑰演算法中匯入 KEK （金鑰用量 K1) 或 BDK （金鑰用量 B0) 金鑰。

下圖顯示end-to-end實體金鑰交換程序。



1. 啟動 – 您提交支援票證或與您的客戶經理合作提交請求。
2. 客戶設定 – AWS 付款密碼編譯提供 CloudFormation 範本，供您完成下列先決條件步驟：
   - 在 AWS 付款密碼帳戶中建立 ECC P521 金鑰對，並擷取公有金鑰憑證。
   - 使用授予 AWS 付款密碼編譯服務主體讀取/寫入存取權的政策建立 Amazon S3 儲存貯體。
   - 將 ECC 公有憑證和簽署根 CA 存放在 Amazon S3 儲存貯體中。
   - 提供金鑰屬性：金鑰用量、金鑰使用模式，以及要傳送的紙質金鑰元件數量。

3. 共用 S3 儲存貯體名稱 – 客戶共用 CloudFormation 堆疊建立的 S3 儲存貯體名稱，其中存放公有金鑰憑證、憑證鏈和金鑰屬性，供 AWS 付款密碼編譯啟動金鑰交換。
4. 運送協調 – AWS 付款密碼編譯提供美國安全設施的運送詳細資訊。您或您的合作夥伴將紙質金鑰元件運送到 AWS 金鑰託管人。
5. 元件接收 – AWS 金鑰保管人會收到每個紙質元件，並為每個元件傳送單獨的確認。
6. 金鑰典禮 – AWS 金鑰保管人會使用離線 HSM 執行金鑰典禮。產生的 TR-31 金鑰區塊，使用 ECDH 衍生的 AES-256 金鑰包裝，來自離線 HSM 的 ECC 公有憑證，及其簽署憑證會上傳至您的 Amazon S3 儲存貯體。
7. 完成 – AWS 付款密碼編譯會傳送金鑰典禮已完成的確認。然後，您可以使用 使用非對稱技術 (ECDH) 匯入金鑰方法將 ECDH 包裝的 TR-31 金鑰區塊匯入您的 AWS 付款密碼帳戶。
8. 帳單 – 成功完成金鑰典禮後，每個交換的金鑰都會向您收費。

安全和合規

實體金鑰交換在設計為符合 PCI PIN 和 PCI P2PE 實體和邏輯安全要求 AWS 的安全設施中運作。有下列控制項：

雙重控制和職責分離

AWS 金鑰託管人是由具有不同報告結構的不同團隊指派。已制定程序，以確保在雙重控制下執行重要典禮步驟。

離線 HSM

金鑰典禮是使用經認證的 PCI PTS HSM 列出的硬體安全模組來執行，這些模組可在沒有網路連線的情況下離線操作。您的金鑰永遠不會存在於 HSM 界限之外的純文字中。

密碼編譯金鑰交付

金鑰材料會使用以 ECDH 為基礎的金鑰交換，從離線 HSM 傳輸到您的 AWS 付款密碼帳戶，以確保end-to-end密碼編譯保護。

稽核與合規

AWS 已制定程序，以滿足針對 PCI PIN 和 P2PE 認證定期評估的適用合規要求。檢閱 AWS Artifact 中的合規套件，以取得您在自己的 PCI 評估中參考的報告。

使用別名

別名是 AWS 付款密碼編譯金鑰的易記名稱。例如，別名可讓您將金鑰稱為 `alias/test-key`而非 `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`。

您可以使用別名來識別大多數金鑰管理（控制平面）操作和密碼編譯（資料平面）操作中的金鑰。

您也可以根據 AWS 付款密碼編譯金鑰的別名允許和拒絕存取，而無需編輯政策或管理授予。此功能是服務對屬性型存取控制 (ABAC) 支援的一部分。

別名也可以更容易在不同的 AWS 區域中重複使用相同的程式碼。在多個區域中建立具有相同名稱的別名，並將每個別名與其區域中的 AWS 付款密碼編譯金鑰建立關聯。當程式碼在每個區域中執行時，別名是指該區域中相關聯的 AWS 付款密碼編譯金鑰。

您可以使用 `CreateAlias` API 建立 AWS 付款密碼編譯金鑰的別名。

AWS 付款密碼編譯 API 提供每個帳戶和區域中別名的完整控制權。API 包含建立別名 (CreateAlias)、檢視別名名稱和連結 keyARN (list-aliaseses)、變更與別名 (update-alias) 相關聯的 AWS 付款密碼編譯金鑰，以及刪除別名 (delete-alias) 的操作。

主題

- 關於別名
- 在應用程式中使用別名
- 相關 API

關於別名

了解別名如何在 AWS 付款密碼編譯中運作。

別名是獨立的 AWS 資源

> 別名不是 AWS 付款密碼編譯金鑰的屬性。您在別名上採取的動作不會影響其相關聯的金鑰。您可以為 AWS 付款密碼編譯金鑰建立別名，然後更新別名，使其與不同的 AWS 付款密碼編譯金鑰相關聯。您甚至可以刪除別名，而不會影響相關聯的 AWS 付款密碼編譯金鑰。如果您刪除 AWS 付款密碼編譯金鑰，與該金鑰相關聯的所有別名都將取消指派。
>
> 如果您將別名指定為 IAM 政策中的資源，則政策會參考別名，而不是相關聯的 AWS 付款密碼編譯金鑰。

每個別名都有易記的名稱

當您建立別名時，您可以指定 字首的別名名稱alias/。例如 alias/test_1234

每個別名一次都與一個 AWS 付款密碼編譯金鑰相關聯

別名及其 AWS 付款密碼編譯金鑰必須位於相同的帳戶和區域中。

AWS 付款密碼編譯金鑰可以同時與多個別名建立關聯，但每個別名只能對應至單一金鑰

例如，此list-aliases輸出顯示alias/sampleAlias1別名僅與一個由 KeyArn 屬性表示的目標 AWS 付款密碼編譯金鑰相關聯。

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

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

多個別名可以與相同的 AWS 付款密碼編譯金鑰相關聯

例如，您可以將 alias/sampleAlias1;和 alias/sampleAlias2 別名與相同的金鑰建立關聯。

```
$ 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"
            }
        ]
    }
```

指定帳戶和區域的別名必須是唯一的

例如，您在每個帳戶和區域只能有一個 alias/sampleAlias1 別名。別名區分大小寫，但我們建議不要使用大小寫不同的別名，因為它們可能會容易出錯。您無法變更別名名稱。但是，您可以刪除別名，並使用所需名稱建立新別名。

但是，您可以在不同區域中使用相同的名稱建立別名。

例如，您可以在alias/sampleAlias2美國東部（維吉尼亞北部）有別名，在美國alias/sampleAlias2西部（奧勒岡）有別名。每個別名都會與其區域中的 AWS 付款密碼編譯金鑰相關聯。如果您的程式碼引用了類似 alias/finance-key 的別名名稱，則可以在多個區域中執行。在每個區域中，它使用不同的別名/sampleAlias2。如需詳細資訊，請參閱在應用程式中使用別名。

您可以變更與別名相關聯的 AWS 付款密碼編譯金鑰

您可以使用 UpdateAlias操作，將別名與不同的 AWS 付款密碼編譯金鑰建立關聯。例如，如果alias/sampleAlias2別名與arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h AWS 付款密碼編譯金鑰相關聯，您可以更新它，使其與arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi金鑰相關聯。

> **Warning**
>
> AWS 付款密碼編譯不會驗證舊金鑰和新金鑰具有所有相同的屬性，例如金鑰用量。使用不同的金鑰類型更新可能會導致您的應用程式發生問題。

> **Warning**
>
> 別名的更新取決於最終一致性。更新別名後會有一段短暫的期間，別名可能會解析為舊金鑰或新金鑰。如果此行為會中斷您的程序，請勿更新別名。

有些金鑰沒有別名

別名是選用功能，除非您選擇以這種方式操作環境，否則並非所有金鑰都會有別名。金鑰可以使用 `create-alias`命令與別名建立關聯。此外，您可以使用 update-alias 操作來變更與別名相關聯的 AWS 付款密碼編譯金鑰，並使用 delete-alias 操作來刪除別名。因此，有些 AWS 付款密碼編譯金鑰可能有多個別名，有些則可能沒有。

將金鑰映射至別名

您可以使用 `create-alias`命令將金鑰（由 ARN 表示）映射至一或多個別名。此命令不是等冪 - 若要更新別名，請使用 update-alias 命令。

```
$ 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"
    }
}
```

在應用程式中使用別名

您可以使用別名來代表應用程式程式碼中的 AWS 付款密碼編譯金鑰。 AWS 付款密碼編譯資料操作以及 List Keys 等其他操作中的 `key-identifier` 參數接受別名名稱或別名 ARN。

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

使用別名 ARN 時，請記住，對應至 AWS 付款密碼編譯金鑰的別名是在擁有 AWS 付款密碼編譯金鑰的帳戶中定義，而且在每個區域中可能有所不同。

其中一個最強大的別名用途是在多個 AWS 區域中執行之應用程式中使用。

您可以在每個區域中建立不同的應用程式版本，或使用字典、組態或切換陳述式來為每個區域選取正確的 AWS 付款密碼編譯金鑰。但在每個區域中建立具有相同別名名稱的別名可能比較容易。請記住，別名名稱區分大小寫。

相關 API

Tags (標籤)

標籤是金鑰和值對，可做為中繼資料來組織您的 AWS 付款密碼編譯金鑰。它們可用來彈性識別金鑰，或將一或多個金鑰分組在一起。

取得金鑰

AWS 付款密碼編譯金鑰代表單一單位的密碼編譯資料，只能用於此服務的密碼編譯操作。GetKeys API 接受 KeyIdentifier 做為輸入，並傳回金鑰中繼資料，包括屬性、狀態和時間戳記，但不傳回實際的密碼編譯金鑰材料。

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"
  }
}
```

取得與金鑰對相關聯的公有金鑰/憑證

Get Public Key/Certificate 會傳回 所指示的公有金鑰KeyArn。這可以是 AWS 付款密碼編譯上產生的金鑰對的公有金鑰部分，或先前匯入的公有金鑰。最常見的使用案例是將公有金鑰提供給將加密資料的外部服務。然後，該資料可以傳遞到利用 AWS Payment Cryptography 的應用程式，並且可以使用 AWS 在 Payment Cryptography 中保護的私有金鑰解密資料。

服務會以公有憑證的形式傳回公有金鑰。API 結果包含 CA 和公有金鑰憑證。這兩個資料元素都是 base64 編碼。

> **Note**
>
> 傳回的公有憑證旨在短暫存留，而非等冪。即使公有金鑰本身保持不變，您也可能在每次 API 呼叫時收到不同的憑證。

Example

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

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

標記金鑰

在 AWS 付款密碼編譯中，您可以在建立金鑰時將標籤新增至 AWS 付款密碼編譯金鑰，並標記或取消標記現有金鑰，除非它們正在等待刪除。 ???標籤是選用的，但它們可以非常有用。

如需標籤的一般資訊，包括最佳實務、標記策略，以及標籤的格式和語法，請參閱 中的標記 AWS 資源Amazon Web Services 一般參考。

主題

- 關於 AWS 付款密碼編譯中的標籤
- 在主控台中檢視金鑰標籤
- 使用 API 操作管理金鑰標籤
- 控制對標籤的存取
- 使用標籤控制對金鑰的存取

關於 AWS 付款密碼編譯中的標籤

標籤是您可以指派 （或 AWS 可指派） 至 AWS 資源的選用中繼資料標籤。每個標籤皆包含標籤索引鍵和標籤值，它們都是區分大小寫的字串。此標籤值可以是空 (null) 字串。資源上的每個標籤都必須有不同的標籤索引鍵，但您可以將相同的標籤新增至多個 AWS 資源。每個資源最多可以有 50 個使用者建立的標籤。

請勿在標籤金鑰或標籤值包含機密或敏感資訊。許多 都可以存取標籤 AWS 服務，包括帳單。

在 AWS 付款密碼編譯中，您可以在建立金鑰時將標籤新增至金鑰，並標記或取消標記現有金鑰，除非它們正在等待刪除。您無法標記別名。標籤是選用的，但它們可以非常有用。

例如，您可以將`"Project"="Alpha"`標籤新增至用於 Alpha 專案的所有 AWS 付款密碼編譯金鑰和 Amazon S3 儲存貯體。另一個範例是將`"BIN"="20130622"`標籤新增至與特定銀行識別號碼 (BIN) 相關聯的所有金鑰。

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

如需標籤的一般資訊，包括格式和語法，請參閱《》中的標記 AWS 資源Amazon Web Services 一般參考。

標籤可協助您執行以下操作：

- 識別和組織您的 AWS 資源。許多 AWS 服務支援標記，因此您可以將相同的標籤指派給來自不同 服務的資源，以指出資源相關。例如，您可以將相同的標籤指派給 AWS 付款密碼編譯金鑰和 Amazon Elastic Block Store (Amazon EBS) 磁碟區或 AWS Secrets Manager 秘密。您也可以使用標籤來識別自動化的金鑰。
- 追蹤您的 AWS 成本。當您將標籤新增至 AWS 資源時， AWS 會產生成本分配報告，其中包含依標籤彙總的用量和成本。您可以使用此功能來追蹤專案、應用程式或成本中心的 AWS 付款密碼編譯成本。

  如需有關使用成本配置標籤的詳細資訊，請參閱《AWS Billing 使用者指南》中的使用成本分配標籤。如需標籤鍵和標籤值規則的相關資訊，請參閱《AWS Billing 使用者指南》中的使用者定義的標籤限制。
- 控制對 AWS 資源的存取。允許和拒絕根據金鑰標籤存取金鑰，是屬性型存取控制 (ABAC) 的 AWS 付款密碼編譯支援的一部分。如需根據標籤控制 AWS Payment Cryptography 存取的資訊，請參閱以 AWS 付款密碼編譯標籤為基礎的授權。如需使用標籤控制 AWS 資源存取的一般資訊，請參閱《IAM 使用者指南》中的使用資源標籤控制 AWS 資源存取。

AWS 當您使用 TagResource、UntagResource 或 ListTagsForResource 操作時，付款密碼編譯會將項目寫入 AWS CloudTrail 日誌。

在主控台中檢視金鑰標籤

若要在主控台中檢視標籤，您需要從包含金鑰的 IAM 政策對金鑰加上標記許可。除了在 主控台中檢視金鑰的許可之外，您還需要這些許可。

使用 API 操作管理金鑰標籤

您可以使用AWS 付款密碼編譯 API 來新增、刪除和列出您管理之金鑰的標籤。以下範例使用 AWS Command Line Interface (AWS CLI)，但您可以使用任何支援的程式設計語言。您無法標記 AWS 受管金鑰。

若要新增、編輯、檢視和刪除金鑰的標籤，您必須擁有必要的許可。如需詳細資訊，請參閱控制對標籤的存取。

主題

- CreateKey：將標籤新增至新金鑰
- TagResource：新增或變更金鑰的標籤
- ListResourceTags：取得金鑰的標籤
- UntagResource：從金鑰刪除標籤

CreateKey：將標籤新增至新金鑰

您可以在建立金鑰時新增標籤。若要指定標籤，請使用 CreateKey 操作的 Tags 參數。

若要在建立金鑰時新增標籤，發起人必須在 IAM 政策中具有payment-cryptography:TagResource許可。許可至少必須涵蓋帳戶和區域中的所有金鑰。如需詳細資訊，請參閱控制對標籤的存取。

CreateKey 的 Tags 參數值是區分大小寫的標籤鍵和標籤值對的集合。金鑰上的每個標籤都必須有不同的標籤名稱。標籤值可以為 null 或空字串。

例如，下列 AWS CLI 命令會建立具有 Project:Alpha 標籤的對稱加密金鑰。指定多個索引鍵/值組時，請使用空格來分隔每一組。

```
$ 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"}]'
```

當此命令成功時，它會傳回包含新金鑰相關資訊的Key物件。但是，Key 不包含標籤。若要取得標籤，請使用 ListResourceTags 操作。

TagResource：新增或變更金鑰的標籤

TagResource 操作會將一或多個標籤新增至金鑰。您無法使用此操作新增或編輯不同 AWS 帳戶中的標籤。

若要新增標籤，請指定新標籤索引鍵和標籤值。若要編輯標籤，請指定現有標籤索引鍵和新標籤值。金鑰上的每個標籤都必須有不同的標籤金鑰。標籤值可以為 null 或空字串。

例如，下列命令會將 **UseCase**和 **BIN** 標籤新增至範例金鑰。

```
$ 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"}]'
```

當此命令成功時，不會傳回任何輸出。若要檢視金鑰上的標籤，請使用 ListResourceTags 操作。

您也可以使用 TagResource 來變更現有標籤的標籤值。若要取代標籤值，請使用不同的值來指定相同的標籤索引鍵。未列在修改命令中的標籤不會變更或移除。

例如，這個命令會將 `Project` 標籤的值從 `Alpha` 變更為 `Noe`。

命令會傳回不含內容的 http/200。若要查看您的變更，請使用 `ListTagsForResource`

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

ListResourceTags：取得金鑰的標籤

ListResourceTags 操作會取得金鑰的標籤。`ResourceArn` (keyArn 或 keyAlias) 參數是必要的。您無法使用此操作來檢視不同 中金鑰上的標籤 AWS 帳戶。

例如，下列命令會取得範例金鑰的標籤。

```
$ 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：從金鑰刪除標籤

UntagResource 操作會從金鑰刪除標籤。若要識別要刪除的標籤，請指定標籤索引鍵。您無法使用此操作從不同的索引鍵刪除標籤 AWS 帳戶。

成功時，UntagResource 操作不會傳回任何輸出。此外，如果在金鑰上找不到指定的標籤金鑰，則不會擲回例外狀況或傳回回應。若要確認操作已運作，請使用 ListResourceTags 操作。

例如，此命令會從指定的金鑰刪除**Purpose**標籤及其值。

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

控制對標籤的存取

若要使用 API 新增、檢視和刪除標籤，委託人需要在 IAM 政策中標記許可。

您也可以使用標籤的 AWS 全域條件索引鍵來限制這些許可。在 AWS 付款密碼編譯中，這些條件可以控制標記操作的存取，例如 TagResource 和 UntagResource。

如需政策和詳細資訊，請參閱《IAM 使用者指南》中的根據標籤索引鍵控制存取。

建立和管理標籤的許可如下所示。

payment-cryptography：TagResource

允許主體新增或編輯標籤。若要在建立金鑰時新增標籤，主體必須在 IAM 政策中具有不限於特定金鑰的許可。

payment-cryptography：ListTagsForResource

允許主體檢視金鑰上的標籤。

payment-cryptography：UntagResource

允許主體從金鑰刪除標籤。

標記政策中的許可

您可以在金鑰政策或 IAM 政策中提供標記許可。例如，下列範例金鑰政策為選取使用者提供金鑰的標記許可。它為所有可以擔任範例管理員或開發人員角色的使用者提供檢視標籤的許可。

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": "*"
    }
  ]
}
```

若要為主體提供多個金鑰的標記許可，您可以使用 IAM 政策。若要讓此政策生效，每個金鑰的金鑰政策必須允許帳戶使用 IAM 政策來控制對金鑰的存取。

例如，下列 IAM 政策允許主體建立金鑰。它還允許他們建立和管理指定帳戶中所有金鑰的標籤。此組合可讓主體使用 CreateKey 操作的標籤參數，在建立金鑰時將標籤新增至金鑰。

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/*"
    }
  ]
}
```

限制標籤許可

您可以使用政策條件來限制標記許可。下列政策條件可套用至 `payment-cryptography:TagResource` 和 `payment-cryptography:UntagResource` 許可。例如，您可以使用 `aws:RequestTag/`*`tag-key`* 條件，允許主體僅新增特定標籤，或防止主體新增具有特定標籤索引鍵的標籤。

- aws:RequestTag
- aws:ResourceTag/tag-key (僅限 IAM 政策)
- aws:TagKeys

當您使用標籤來控制對金鑰的存取時，最佳實務是使用 aws:RequestTag/*tag-key*或 aws:TagKeys條件金鑰來判斷允許哪些標籤（或標籤金鑰）。

例如，下列 IAM 政策與前一個類似。不過，此政策允許主體建立標籤 (TagResource) 並僅為具有 Project 標籤索引鍵的標籤刪除標籤 UntagResource。

由於 TagResource 和 UntagResource 請求可以包含多個標籤，您必須指定具有 aws:TagKeys 條件的 ForAllValues 或 ForAnyValue 集合運算子。ForAnyValue 運算子會要求請求中的至少一個標籤索引鍵與政策中的標籤索引鍵相符。ForAllValues 運算子會要求請求中的所有標籤索引鍵與政策中的其中一個標籤索引鍵相符。ForAllValues 運算子也會傳回 true，如果在請求中沒有標籤，但 TagResource 和 UntagResource 會在未指定標籤時失敗。如需集合運算子的詳細資訊，請參閱《IAM 使用者指南》中的使用多個索引鍵和值。

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"}
      }
    }
```

```
  ]
}
```

使用標籤控制對金鑰的存取

您可以根據金鑰上的標籤來控制對 AWS 付款密碼編譯的存取。例如，您可以撰寫 IAM 政策，允許主體僅啟用和停用具有特定標籤的金鑰。或者，您可以使用 IAM 政策來防止主體在密碼編譯操作中使用金鑰，除非金鑰具有特定標籤。

此功能是屬性型存取控制 (ABAC) AWS 付款密碼編譯支援的一部分。如需有關使用標籤來控制資源 AWS 存取的資訊，請參閱 ABAC 的用途是什麼 AWS？ 和 IAM 使用者指南中的使用資源標籤控制 AWS 對 資源的存取。

AWS Payment Cryptography 支援 aws：ResourceTag/tag-key 全域條件內容金鑰，可讓您根據金鑰上的標籤控制對金鑰的存取。由於多個金鑰可以有相同的標籤，此功能可讓您將許可套用至一組選取的金鑰。您也可以透過變更金鑰的標籤，輕鬆變更集合中的金鑰。

在 AWS 付款密碼編譯中，僅在 IAM 政策中支援 `aws:ResourceTag/`*`tag-key`*條件金鑰。金鑰政策不支援此功能，這僅適用於一個金鑰，或是不使用特定金鑰的操作，例如 ListKeys 或 ListAliases 操作。

使用標籤控制存取可提供一種簡單、可擴展且靈活的方式來管理許可。不過，如果未正確設計和管理，可能會不小心允許或拒絕存取您的金鑰。如果您使用標籤來控制存取，請考慮下列實務。

- 使用標籤來強化最低權限存取的最佳實務。只為 IAM 主體提供他們必須使用或管理的金鑰所需的許可。例如，使用標籤來標記用於專案的金鑰。然後，授予專案團隊僅搭配專案標籤使用金鑰的許可。
- 要謹慎地授予主體 `payment-cryptography:TagResource` 和 `payment-cryptography:UntagResource` 許可，讓其新增、編輯和刪除別名。當您使用標籤來控制對金鑰的存取時，變更標籤可以授予主體使用金鑰的許可，否則他們就沒有使用金鑰的許可。它也可以拒絕存取其他委託人執行其任務所需的金鑰。如果金鑰管理員沒有變更金鑰政策或建立授予的許可，則可以控制對金鑰的存取，如果他們具有管理標籤的許可。

  盡可能使用政策條件，例如 `aws:RequestTag/`*`tag-key`*或 `aws:TagKeys`，將委託人的標記許可限制在特定索引鍵上的特定標籤或標籤模式。
- 檢閱 中目前具有標記和取消標記許可 AWS 帳戶 的主體，並視需要進行調整。IAM 政策可能允許在所有金鑰上標記和取消標記許可。例如，管理員受管政策允許主體在所有金鑰上標記、取消標記和列出標籤。

- 在設定取決於標籤的政策之前，請檢閱 中金鑰上的標籤 AWS 帳戶。請確定您的政策僅適用於您想要包含的標籤。使用 CloudTrail 日誌和 CloudWatch 警示來提醒您標記可能影響金鑰存取的變更。
- 標籤型政策條件使用模式比對；其不會繫結至標籤的特定執行個體。使用標籤型條件索引鍵的政策會影響所有符合模式的新標籤和現有標籤。如果您刪除並重新建立符合政策條件的標籤，則條件會套用至新標籤，就像舊標籤一樣。

例如，請考慮以下 IAM 政策。它允許委託人僅在您帳戶中屬於美國東部（維吉尼亞北部）區域的金鑰上呼叫解密操作，並具有"Project"="Alpha"標籤。您可以將此政策連接至 Alpha 專案範例中的角色。

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"
        }
      }
    }
  ]
}
```

下列範例 IAM 政策允許主體將帳戶中的任何金鑰用於特定密碼編譯操作。但其禁止主體在具有"Type"="Reserved"標籤或無"Type"標籤的金鑰上使用這些密碼編譯操作。

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"
        }
      }
    }
  ]
}
```

了解 AWS 付款密碼編譯金鑰的金鑰屬性

適當的金鑰管理原則是金鑰具有適當的範圍，並且只能用於允許的操作。因此，特定金鑰只能使用特定金鑰使用模式建立。盡可能與 TR-31 定義的可用使用模式保持一致。

雖然 AWS 付款密碼編譯會阻止您建立無效的金鑰，但為了方便起見，此處會提供有效的組合。

對稱金鑰

- TR31_B0_BASE_DERIVATION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_C0_CARD_VERIFICATION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Encrypt = true、Decrypt = true、包裝 = true、Unwrap = true }、{ Encrypt = true、包裝 = true }、{ Decrypt = true、Unwrap = true }、{ NoRestrictions = true }
- TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS
  - 允許的金鑰演算法：TDES_2KEY、TDES_3KEY*、AES_128*、AES_192*、AES_256*
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E1_EMV_MKEY_CONFIDENTIALITY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY、AES_128*、AES_192*、AES_256*
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E2_EMV_MKEY_INTEGRITY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E5_EMV_MKEY_CARD_PERSONALIZATION

  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E6_EMV_MKEY_OTHER
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128* 、AES_192* 、AES_256*
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_K0_KEY_ENCRYPTION_KEY
  - 建議使用 TR31_K1_KEY_BLOCK_PROTECTION_KEY。允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Encrypt = true、Decrypt = true、包裝 = true、Unwrap = true }、{ Encrypt = true、包裝 = true }、{ Decrypt = true、Unwrap = true }、{ NoRestrictions = true }
- TR31_K1_KEY_BLOCK_PROTECTION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Encrypt = true， Decrypt = true， Wrap = true， Unwrap = true } ，{ Encrypt = true， Wrap = true } ，{ Decrypt = true， Unwrap = true } ，{ NoRestrictions = true }
- TR31_M1_ISO_9797_1_MAC_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_M3_ISO_9797_3_MAC_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_M6_ISO_9797_5_CMAC_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_M7_HMAC_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_P0_PIN_ENCRYPTION_KEY

- 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
- 允許的金鑰使用模式組合：{ Encrypt = true， Decrypt = true， Wrap = true， Unwrap = true } ，{ Encrypt = true， Wrap = true } ，{ Decrypt = true， Unwrap = true } ，{ NoRestrictions = true }

- TR31_V1_IBM3624_PIN_VERIFICATION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_V2_VISA_PIN_VERIFICATION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }

非對稱金鑰

- TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION
  - 允許金鑰演算法：RSA_2048、RSA_3072、RSA_4096
  - 允許的金鑰使用模式組合：{ Encrypt = true， Decrypt = true， Wrap = true， Unwrap = true } ，{ Encrypt = true， Wrap = true } ，{ Decrypt = true， Unwrap = true }
  - 注意：{ Encrypt = true， Wrap = true } 是匯入用於加密資料或包裝金鑰的公有金鑰時的唯一有效選項
- TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE
  - 允許金鑰演算法：RSA_2048、RSA_3072、RSA_4096
  - 允許的金鑰使用模式組合：{ Sign = true } ，{ Verify = true }
  - 注意：{ Verify = true } 是匯入用於簽署的金鑰時的唯一有效選項，例如根憑證、中繼憑證或 TR-34 的簽署憑證。
- TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT
  - 用於金鑰協議演算法，例如 ECDH
  - 允許金鑰演算法：ECC_NIST_P256、ECC_NIST_P384、ECC_NIST_P521
  - 允許的金鑰使用模式組合：{ DeriveKey = true }。
  - 注意：DeriveKeyUsage 用於指定將從此基本金鑰衍生的金鑰類型。這是在金鑰建立/匯入時修正的。
- TR31_K2_TR34_ASYMMETRIC_KEY

- 用於 TR-34 等 X9.24 相容金鑰交換機制的非對稱金鑰
- 允許金鑰演算法：RSA_2048、RSA_3072、RSA_4096
- 允許的金鑰使用模式組合：{ DeriveKey = true }。
- 允許的金鑰使用模式組合：{ Encrypt = true， Decrypt = true， Wrap = true， Unwrap = true } ，{ Encrypt = true， Wrap = true } ，{ Decrypt = true， Unwrap = true }
- 注意：{ Encrypt = true， Wrap = true } 是匯入用於加密資料或包裝金鑰的公有金鑰時的唯一有效選項

* 任何密碼編譯操作目前不支援此演算法/金鑰類型組合

資料操作

建立 AWS 付款密碼編譯金鑰之後，即可用來執行密碼編譯操作。不同的操作會執行不同類型的活動，範圍從加密、雜湊，以及領域特定的演算法，例如 CVV2 產生。

如果沒有相符的解密金鑰（對稱金鑰或私有金鑰，取決於加密類型），則無法解密加密的資料。如果沒有對稱金鑰或公有金鑰，就無法同樣驗證雜湊和網域特定的演算法。

如需特定操作的有效金鑰類型資訊，請參閱密碼編譯操作的有效金鑰

> **Note**
>
> 我們建議在非生產環境中使用測試資料。在非生產環境中使用生產金鑰和資料 (PAN、BDK ID 等） 可能會影響您的合規範圍，例如 PCI DSS 和 PCI P2PE。

主題

- 加密、解密和重新加密資料
- 產生和驗證卡片資料
- 產生、翻譯和驗證 PIN 資料
- 產生並驗證 ARQC
- 產生和驗證 MAC
- 密碼編譯操作的有效金鑰

加密、解密和重新加密資料

加密和解密方法可用於使用各種對稱和非對稱技術加密或解密資料，包括 TDES、AES 和 RSA。這些方法也支援使用 DUKPT 和 EMV 技術衍生的金鑰。對於您想要在新金鑰下保護資料而不公開基礎資料的使用案例，也可以使用 ReEncrypt 命令。

> **Note**
>
> 使用加密/解密函數時，所有輸入都假設為 hexBinary - 例如，值 1 將輸入為 31（十六進位），小寫 t 表示為 74（十六進位）。所有輸出也都使用 hexBinary。

如需所有可用選項的詳細資訊，請參閱適用於加密、解密和重新加密的 API 指南。

主題

- 加密資料
- 解密資料

加密資料

Encrypt Data API 用於使用對稱和非對稱資料加密金鑰以及 DUKPT 和 EMV 衍生的金鑰來加密資料。支援各種演算法和變化，包括 TDES、 RSA和 AES。

主要輸入是用來加密資料的加密金鑰、要加密的 hexBinary 格式純文字資料，以及加密屬性，例如初始化向量和 TDES 等區塊加密的模式。對於 ，純文字資料需要 8 個位元組的倍數TDES，對於 ，需要 16 個位元組AES，對於 ，需要 金鑰的長度RSA。如果輸入資料不符合這些要求，則應填充對稱金鑰輸入 (TDES、AES、DUKPT、EMV)。下表顯示每種類型金鑰的純文字長度上限，以及您在 EncryptionAttributes RSA 金鑰中定義的填補類型。

填補類型	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

主要輸出包含 hexBinary 格式的加密資料做為加密文字，以及加密金鑰的檢查總和值。如需所有可用選項的詳細資訊，請參閱 加密 API 指南。

範例

- 使用 AES 對稱金鑰加密資料
- 使用 DUKPT 金鑰加密資料
- 使用 EMV 衍生對稱金鑰加密資料

- 使用 RSA 金鑰加密資料

使用 AES 對稱金鑰加密資料

> **Note**
>
> 所有範例都假設相關的金鑰已存在。您可以使用 CreateKey 操作建立金鑰，或使用 ImportKey 操作匯入金鑰。

Example

在此範例中，我們將使用使用 CreateKey 操作建立或使用 ImportKey 操作匯入的對稱金鑰來加密純文字資料。對於此操作，金鑰必須將 KeyModesOfUse 設定為 `Encrypt`，且 KeyUsage 設定為 `TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY`。如需更多選項，請參閱密碼編譯操作的金鑰。

```
$ 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"
}
```

使用 DUKPT 金鑰加密資料

Example

在此範例中，我們將使用 DUKPT 金鑰加密純文字資料。 AWS 付款密碼編譯支援 TDES和 AES DUKPT 金鑰。對於此操作，金鑰必須將 KeyModesOfUse 設定為 DeriveKey，且 KeyUsage 設定為 TR31_B0_BASE_DERIVATION_KEY。如需更多選項，請參閱密碼編譯操作的金鑰。

```
$ 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"
}
```

使用 EMV 衍生對稱金鑰加密資料

Example

在此範例中，我們將使用已建立的 EMV 衍生對稱金鑰來加密純文字資料。您可以使用這類命令將資料傳送至 EMV 卡。對於此操作，金鑰必須將 KeyModesOfUse 設定為 Derive，且 KeyUsage 設定為 TR31_E1_EMV_MKEY_CONFIDENTIALITY或 TR31_E6_EMV_MKEY_OTHER。如需詳細資訊，請參閱密碼編譯操作的金鑰。

```
$ 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"
}
```

使用 RSA 金鑰加密資料

Example

在此範例中，我們將使用已使用 ImportKey 操作匯入的 RSA 公有金鑰來加密純文字資料。對於此操作，金鑰必須將 KeyModesOfUse 設定為 Encrypt，且 KeyUsage 設定為 TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION。如需更多選項，請參閱密碼編譯操作的金鑰。

對於目前不支援的 PKCS #7 或其他填補機制，請在呼叫服務之前套用，並透過省略填補指標 'Asymmetric={}' 來選取不填補

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

```
{
 "CipherText":
 "12DF6A2F64CC566D124900D68E8AFEAA794CA819876E258564D525001D00AC93047A83FB13 \
 E73F06329A100704FA484A15A49F06A7A2E55A241D276491AA91F6D2D8590C60CDE57A642BC64A897F4832A3930
 \
 0FAEC7981102CA0F7370BFBF757F271EF0BB2516007AB111060A9633D1736A9158042D30C5AE11F8C5473EC70F067
 \
 72590DEA1638E2B41FAE6FB1662258596072B13F8E2F62F5D9FAF92C12BB70F42F2ECDCF56AADF0E311D4118FE3591
 \
 FB672998CCE9D00FFFE05D2CD154E3120C5443C8CF9131C7A6A6C05F5723B8F5C07A4003A5A6173E1B425E2B5E42AD
 \
 7A2966734309387C9938B029AFB20828ACFC6D00CD1539234A4A8D9B94CDD4F23A",
 "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/5dza7xqd6soanjtb",
 "KeyCheckValue": "FF9DE9CE"
}
```

解密資料

Decrypt Data API 用於使用對稱和非對稱資料加密金鑰以及 DUKPT 和 EMV 衍生的金鑰來解密資料。支援各種演算法和變化，包括 TDES、 RSA和 AES。

主要輸入是用來解密資料的解密金鑰、要解密的 hexBinary 格式加密文字資料，以及初始化向量、做為區塊加密的模式等解密屬性。主要輸出包含 hexBinary 格式的純文字解密資料，以及解密金鑰的檢查總和值。如需所有可用選項的詳細資訊，請參閱 解密 API 指南。

範例

- 使用 AES 對稱金鑰解密資料
- 使用 DUKPT 金鑰解密資料
- 使用 EMV 衍生對稱金鑰解密資料
- 使用 RSA 金鑰解密資料

使用 AES 對稱金鑰解密資料

Example

在此範例中，我們將使用對稱金鑰解密加密文字資料。此範例顯示 AES金鑰，但也支援 TDES_3KEY TDES_2KEY和 。對於此操作，金鑰必須將 KeyModesOfUse 設定為 Decrypt，且 KeyUsage 設定為 TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY。如需更多選項，請參閱密碼編譯操作的金鑰。

```
$ 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": "313233343132333431323334313233343132333431323334"
}
```

使用 DUKPT 金鑰解密資料

> **Note**
>
> 將 decrypt-data 與 DUKPT for P2PE 交易搭配使用時，可能會將信用卡 PAN 和其他持卡人資料傳回至您的應用程式，這些資料在決定其 PCI DSS 範圍時需要考慮。

Example

在此範例中，我們將使用使用 CreateKey 操作建立或使用 ImportKey 操作匯入的 DUKPT 金鑰來解密加密文字資料。對於此操作，金鑰必須將 KeyModesOfUse 設定為 DeriveKey，且 KeyUsage 設定為 TR31_B0_BASE_DERIVATION_KEY。如需更多選項，請參閱密碼編譯操作的金鑰。當您使用時DUKPT，對於TDES演算法，加密文字資料長度必須是 16 個位元組的倍數。對於AES演算法，加密文字資料長度必須是 32 個位元組的倍數。

```
$ 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"
}
```

使用 EMV 衍生對稱金鑰解密資料

Example

在此範例中，我們將使用使用 CreateKey 操作建立或使用 ImportKey 操作匯入的 EMV 衍生對稱金鑰來解密加密文字資料。對於此操作，金鑰必須將 KeyModesOfUse 設定為 `Derive`，且 KeyUsage 設定為 `TR31_E1_EMV_MKEY_CONFIDENTIALITY`或 `TR31_E6_EMV_MKEY_OTHER`。如需詳細資訊，請參閱密碼編譯操作的金鑰。

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

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

使用 RSA 金鑰解密資料

Example

在此範例中，我們將使用使用 CreateKey 操作建立的 RSA 金鑰對來解密加密文字資料。對於此操作，金鑰必須已將 KeyModesOfUse 設定為啟用Decrypt，且 KeyUsage 設定為 TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION。如需更多選項，請參閱密碼編譯操作的金鑰。

對於目前不支援的 PKCS #7 或其他填補機制，請省略填補指標 'Asymmetric={}'，並在呼叫服務之後移除填補，以選取不填補。

```
$ 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"
}
```

產生和驗證卡片資料

產生並驗證卡片資料包含衍生自卡片資料的資料，例如 CVV、CVV2、CVC 和 DCVV。

主題

- 產生卡片資料
- 驗證卡片資料

產生卡片資料

Generate Card Data API 用於使用 CVV、CVV2 或動態 CVV2 等演算法產生卡資料。若要查看可用於此命令的金鑰，請參閱密碼編譯操作的有效金鑰一節。

許多密碼編譯值，例如 CVV、CVV2、iCVV、CAV V7，都使用相同的密碼編譯演算法，但會改變輸入值。例如CardVerificationValue1 具有 ServiceCode、卡號和過期日期的輸入。雖然 CardVerificationValue2 僅有兩個輸入，這是因為對於 CVV2/CVC2，ServiceCode 固定為 000。同樣地，對於 iCVV，ServiceCode 固定為 999。有些演算法可能會重新利用現有的欄位，例如 CAVV V8，在這種情況下，您將需要參閱供應商手冊以取得正確的輸入值。

> **Note**
>
> 必須以相同的格式輸入過期日期（例如 MMYY 與 YYMM），才能產生正確的結果。

產生 CVV2

Example

在此範例中，我們將為輸入 PAN和卡片過期日期的指定 PAN 產生 CVV2。這會假設您產生了卡片驗證金鑰。

```
$ 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"
  }
```

產生 iCVV

Example

在此範例中，我們將為輸入為 、PAN服務代碼為 999 且卡片過期日期的指定 PAN 產生 iCVV。這會假設您產生了卡片驗證金鑰。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ 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"
}
```

驗證卡片資料

`Verify Card Data` 用於驗證已使用依賴加密主體的付款演算法建立的資料，例如 `DISCOVER_DYNAMIC_CARD_VERIFICATION_CODE`。

輸入值通常作為傳入交易的一部分提供給發行者或支援平台合作夥伴。若要驗證 ARQC 密碼編譯（用於 EMV 晶片卡），請參閱驗證 ARQC。

如需詳細資訊，請參閱 API 指南中的 VerifyCardValidationData。

如果該值已經過驗證，則 api 將傳回 http/200。如果未驗證該值，則會傳回 http/400。

驗證 CVV2

Example

在此範例中，我們將驗證指定 PAN 的 CVV/CVV2。CVV2 通常由持卡人或使用者在交易期間提供以供驗證。為了驗證其輸入，將在執行時間提供下列值 - 用於驗證的金鑰 (CVK)、PAN、卡片過期日期和輸入的 CVV2。卡片過期格式必須符合用於產生初始值的格式。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue2。

```
$ 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"
  }
```

驗證 iCVV

Example

在此範例中，我們將驗證指定 PAN 的 iCVV，其中包含用於驗證的金鑰 (CVK)、PAN、999 的服務碼、卡片過期日期，以及交易提供的 iCVV 以進行驗證。

iCVV 不是使用者輸入的值 （ 例如 CVV2)，而是內嵌在 EMV 卡上。應考量是否應一律在提供時驗證。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ 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"
}
```

產生、翻譯和驗證 PIN 資料

PIN 資料函數可讓您產生隨機接腳、接腳驗證值 (PVV)，並根據 PVV 或接腳位移驗證傳入加密接腳。

接腳轉換可讓您將接腳從一個工作金鑰轉換為另一個工作金鑰，而不會以 PCI PIN 要求 1 指定的純文字公開接腳。

> **Note**
>
> 由於 PIN 產生和驗證通常是發行者函數，而 PIN 轉譯是典型的取得者函數，我們建議您考慮最低權限的存取，並根據您的系統使用案例適當設定政策。

主題

- 翻譯 PIN 資料
- 產生 PIN 資料
- 驗證 PIN 資料

翻譯 PIN 資料

轉譯 PIN 資料函數用於將加密的 PIN 資料從一組金鑰轉譯到另一組金鑰，而不需要加密的資料離開 HSM。它用於 P2PE 加密，其中工作金鑰應該變更，但處理系統不需要或不允許解密資料。主要輸入是加密的資料、用來加密資料的加密金鑰、用來產生輸入值的參數。另一組輸入是請求的輸出參數，例如用於加密輸出的金鑰，以及用於建立該輸出的參數。主要輸出是新加密的資料集，以及用來產生資料集的參數。

> **Note**
>
> 為了 PCI 合規，傳入和傳出的 PrimaryAccountNumber 值必須相符。不允許將 PIN 從一個 PAN 轉譯為另一個 PAN。

主題

- 從 PEK 到 DUKPT 的 PIN
- 從 PEK 到 PEK 的 PIN

從 PEK 到 DUKPT 的 PIN

Example

在此範例中，我們將使用 DUKPT 將 PIN 從 AES ISO 4 PIN 區塊轉換為使用 ISO 0 PIN 區塊的 PEK TDES 加密。這在付款終端機加密 ISO 4 中的 PIN 時很常見，如果下一個連線尚未支援 AES，則可能會將其轉譯回 TDES 以進行下游處理。

```
$ 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"
        }
```

從 PEK 到 PEK 的 PIN

Example

在此範例中，我們將在一個 PEK (PIN 加密金鑰） 下加密的 PIN 轉換為另一個 PEK。這通常用於在使用不同加密金鑰的不同系統或合作夥伴之間路由交易時，同時透過在整個過程中保持 PIN 加密來維持 PCI PIN 合規。這兩個金鑰在此範例中都使用 TDES 3KEY 加密，但有多種選項可用，包括 AES ISO-4 到 TDES ISO-0、DUKPT 到 PEK 或 AS2805 到 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"
}
```

輸出 PIN 區塊現在會在第二個 PEK 下加密，並可安全地傳輸到保留對應金鑰的下游系統。

產生 PIN 資料

產生 PIN 資料函數用於產生 PIN 相關值，例如 PVV 和 PIN 區塊位移，用於驗證使用者在交易或授權時間的 PIN 項目。此 API 也可以使用各種演算法產生新的隨機接腳。

產生隨機接腳和相符的 Visa PVV

Example

在此範例中，我們將產生新的（隨機）接腳，其中輸出將是加密的 PIN block(PinData.PinBlock) 和 PVV(pinData.Offset). 金鑰輸入為 PAN、Pin Verification Key、 Pin Encryption Key和 PIN block format。

此命令要求金鑰類型為 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"
            }
        }
```

為已知接腳產生 Visa PVV

Example

在此範例中，我們將為指定的（加密）接腳產生 PVV。加密的 PIN 可在上游接收，例如從付款終端機，或使用使用者可選取的 PIN 流程從持卡人接收。金鑰輸入為 PAN、Pin Verification Key、Pin Encryption Key、 Encrypted Pin Block和 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"
            }
        }
```

產生接腳的 IBM3624 接腳位移

IBM 3624 PIN 位移有時也稱為 IBM 方法。此方法會使用驗證資料（通常是 PAN) 和 PIN 金鑰 (PVK) 產生自然/中繼 PIN。自然接腳實際上是衍生的值，對於發行者來說，決定性非常有效率，因為不需要將接腳資料存放在持卡人層級。最明顯的缺點是，此機制不會考慮持卡人可選或隨機接腳。為了允許這些類型的接腳，已將偏移演算法新增至結構描述。偏移代表使用者選擇（或隨機）接腳與自然金鑰之間的差異。位移值由卡片發行者或卡片處理器存放。在交易時間， AWS 付款密碼編譯服務會在內部重新計算自然接腳，並套用偏移來尋找接腳。然後，它會將此值與交易授權提供的值進行比較。

IBM3624 有幾個選項：

- `Ibm3624NaturalPin` 將輸出自然接腳和加密接腳區塊

- Ibm3624PinFromOffset 會在有位移的情況下產生加密的 PIN 區塊
- Ibm3624RandomPin 會產生隨機接腳，然後產生相符的位移和加密接腳區塊。
- Ibm3624PinOffset 會在使用者選取的接腳後產生接腳位移。

在 AWS 付款密碼編譯內部，會執行下列步驟：

- 將提供的平移填補為 16 個字元。如果提供 <16，請使用提供的填補字元在右側進行填補。
- 使用 PIN 產生金鑰加密驗證資料。
- 使用小數化資料表對加密的資料進行小數化。這會將十六進位數字映射到執行個體「A」的小數位，可能映射到 9，而 1 可能映射到 1。
- 從輸出的十六進位表示法取得前 4 位數。這是自然接腳。
- 如果產生使用者選取或隨機接腳，模數會使用客戶接腳減去自然接腳。結果是接腳位移。

範例

- 範例：產生接腳的 IBM3624 接腳位移

範例：產生接腳的 IBM3624 接腳位移

在此範例中，我們將產生新的（隨機）接腳，其中輸出將是加密的 PIN block(PinData.PinBlock)和IBM3624位移值 (pinData.Offset). 輸入為 PAN、驗證資料（通常是平移）、填補字元Pin Verification Key、、 Pin Encryption Key和 PIN block format。

此命令要求接腳產生金鑰為 類型，TR31_V1_IBM3624_PIN_VERIFICATION_KEY而加密金鑰為 類型 TR31_P0_PIN_ENCRYPTION_KEY

Example

下列範例顯示產生隨機接腳，然後使用 Ibm3624RandomPin 輸出加密接腳區塊和 IBM3624 偏移值 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"
              }
          }
```

驗證 PIN 資料

驗證 PIN 資料函數用於驗證 PIN 是否正確。這通常涉及比較先前存放的接腳值與持卡人在 POI 中輸入的接腳值。這些函數會比較兩個值，而不會公開任一來源的基礎值。

使用 PVV 方法驗證加密的 PIN

Example

在此範例中，我們將驗證指定 PAN 的 PIN。PIN 通常由持卡人或使用者在驗證的交易時間提供，並與檔案上的值進行比較（持卡人的輸入會以終端機或其他上游提供者的加密值提供）。為了驗證此輸入，也會在執行時間提供下列值：用來加密輸入接腳的金鑰（通常為 IWK)，PAN以及要驗證的值 (PVV或 PIN offset)。

如果 AWS 付款密碼編譯能夠驗證 PIN 碼，則會傳回 http/200。如果未驗證 PIN 碼，則會傳回 http/400。

```
$ 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",

        }
```

使用 PVV 方法驗證加密的 PIN 碼 - 錯誤 PIN 碼錯誤

Example

在此範例中，我們將嘗試驗證指定 PAN 的 PIN，但由於 PIN 不正確而失敗。

使用 SDKs時，這會顯示為 {"Message"："Pin 區塊驗證失敗。"，"Reason"："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.
```

使用 PVV 方法驗證加密的 PIN 碼 - 錯誤輸入錯誤

Example

在此範例中，我們將嘗試驗證指定 PAN 的 PIN，但由於輸入錯誤而失敗，而傳入的資料不是有效的 PIN。常見原因是：使用 1/錯誤金鑰 2/輸入參數，例如平移或接腳區塊格式不正確 3/接腳區塊損毀。

```
$ 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.
```

針對先前存放的 IBM3624 接腳位移驗證 PIN

在此範例中，您會針對儲存在卡片發行者或處理器檔案中的 PIN 偏移來驗證持卡人提供的 PIN。輸入包括驗證金鑰 (PVK)、加密金鑰 (PEK)、主要帳戶號碼、來自付款終端機 （或其他上游提供者，例如卡片網路） 的加密 PIN 區塊，以及先前存放的 PIN 偏移。如果 PIN 相符，API 會傳回 HTTP/200 回應。

此命令需要 PIN 驗證金鑰 類型為 ，`TR31_V1_IBM3624_PIN_VERIFICATION_KEY`而加密金鑰 類型為 `TR31_P0_PIN_ENCRYPTION_KEY`。

> **⚠ PIN 長度驗證**
>
> 選用`--pin-data-length`參數指定要驗證的 PIN 長度 (4 到 12 位數）。如果您未指定此參數，服務會驗證偏移值保護的整個 PIN 長度。如果您只想要驗證 PIN 的一部分，請使用 `--pin-data-length` 參數。例如，如果 `--pin-data-length` 設定為 4，但實際 PIN 為 6 位數，則服務只會驗證最右側的 4 位數。其餘的 2 位數不會勾選。如果持卡人 PIN 碼是 6 位數，但傳入的 PIN 碼只有 4 位數，例如傳入的網路或商家只支援 4 位數的 PIN 碼，您可能會刻意這麼做。

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"
}
```

產生並驗證 ARQC

授權請求密碼編譯 (ARQC) 是在付款交易期間由 EMV 晶片卡 （或行動錢包等數位方式） 產生的密碼編譯。ARQC 會以密碼編譯方式驗證卡片，並確保交易資料的完整性。它結合了卡片資料、終端機資料和交易特定資料，為每個交易產生唯一的值。

AWS 付款密碼編譯支援兩個 ARQC 操作。您可以產生 ARQC 用於開發和測試目的，也可以驗證 ARQC 用於生產交易授權流程。

主題

- 產生身分驗證請求 (ARQC) 密碼編譯
- 驗證身分驗證請求 (ARQC) 密碼編譯

產生身分驗證請求 (ARQC) 密碼編譯

產生身分驗證請求密碼編譯 API 用於產生 ARQC。此 API 可讓您使用 AWS 付款密碼編譯產生 ARQC，以用於開發和測試目的。

> **Important**
>
> 此操作僅適用於開發和測試案例。不建議在生產付款流程中使用此操作來取代以卡片為基礎的密碼編譯產生。

如需所有可用的選項，請參閱 API 參考中的 GenerateAuthRequestCryptogram。

ARQC 密碼編譯通常需要以下輸入（雖然這可能因實作而有所不同）：

- PAN - 在 PrimaryAccountNumber 欄位中指定
- PAN 序號 (PSN) - 在 PanSequenceNumber 欄位中指定
- 金鑰衍生方法，例如通用工作階段金鑰 (CSK) - 在 SessionKeyDerivationAttributes 中指定
- 主金鑰衍生模式（例如 EMV 選項 A) - 在 MajorKeyDerivationMode 中指定
- 交易資料 - 交易、終端機和卡片資料的字串，例如金額和日期。在 TransactionData 欄位中指定。
- 發行者主金鑰 - 用於衍生密碼編譯 (AC) 金鑰的主金鑰。此金鑰可保護個別交易，並在 KeyIdentifier 欄位中指定。

如需建置和填補交易資料的詳細資訊，請參閱驗證身分驗證請求 (ARQC) 密碼編譯。

主題

- 範例

範例

Visa CVN10

Example

下列範例使用 Visa CVN10 產生 ARQC。

如果 AWS 付款密碼編譯成功產生 ARQC，則會使用產生的密碼編譯傳回 http/200。

```
$ 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
 0000000017000000000000000840008000800008401605170000000093800000B03011203000000 \
--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 CVN18 和 Visa CVN22

Example

下列範例使用 Visa CVN18 或 CVN22 產生 ARQC。CVN18 和 CVN22 之間的密碼編譯操作相同，但交易資料中包含的資料有所不同。與 CVN10 相比，即使使用相同的輸入，也會產生完全不同的密碼編譯。

如果 AWS 付款密碼編譯成功產生 ARQC，則會使用產生的密碼編譯傳回 http/200。

```
$ 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
 00000000170000000000000008400080008000084016051700000000093800000B1F22010300000000000
 \
00000000000000000000000000000000000000000000800000000000000000 \
--session-key-derivation-attributes='{"EmvCommon":
{"ApplicationTransactionCounter":"000B", \
"PanSequenceNumber":"01","PrimaryAccountNumber":"9137631040001422"}}'
```

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

驗證身分驗證請求 (ARQC) 密碼編譯

驗證身分驗證請求密碼編譯 API 用於驗證 ARQC。ARQC 的產生通常在交易授權期間於 EMV Chip 卡（或數位同等項目，例如行動錢包）上執行，但服務可基於開發目的產生。ARQC 對每個交易都是唯一的，旨在以密碼編譯方式顯示卡片的有效性，並確保交易資料完全符合目前的（預期）交易。

AWS 付款密碼編譯提供各種選項，用於驗證 ARQC 和產生選用的 ARPC 值，包括 EMV 4.4 Book 2 中定義的值，以及 Visa 和 Mastercard 使用的其他方案。如需所有可用選項的完整清單，請參閱 API 指南中的 VerifyCardValidationData 一節。

ARQC 密碼編譯通常需要以下輸入（雖然這可能因實作而有所不同）：

- PAN - 在 PrimaryAccountNumber 欄位中指定

- PAN 序號 (PSN) - 在 PanSequenceNumber 欄位中指定
- 金鑰衍生方法，例如通用工作階段金鑰 (CSK) - 在 SessionKeyDerivationAttributes 中指定
- 主金鑰衍生模式 （ 例如 EMV 選項 A) - 在 MajorKeyDerivationMode 中指定
- 交易資料 - 各種交易、終端機和卡片資料的字串，例如交易TransactionData欄位中指定的金額和日期
- 發行者主金鑰 - 主金鑰，用於衍生用於保護個別交易的加密法 (AC) 金鑰，並在 KeyIdentifier 欄位中指定

主題

- 建置交易資料
- 交易資料填補
- 範例

建置交易資料

交易資料欄位的確切內容 （ 和順序 ） 會因實作和網路方案而有所不同，但最低建議欄位 （ 和串連序列 ） 是在 EMV 4.4 Book 2 Section 8.1.1 - Data Selection 中定義。如果前三個欄位是金額 (17.00)、其他金額 (0.00) 和購買國家/地區，則會導致交易資料開始，如下所示：

- 000000001700 - 金額 - 12 個位置隱含兩位數小數位數
- 000000000000 - 其他金額 - 12 個位置隱含兩位數小數
- 0124 - 四位數國碼
- 輸出 （ 部分 ） 交易資料 - 00000000170000000000000000124

交易資料填補

交易資料應在傳送至服務之前填入。大多數方案使用 ISO 9797 方法 2 填補，其中十六進位字串附加 hex 80 後接 00，直到 欄位是加密區塊大小的倍數；TDES 為 8 位元組或 16 個字元，AES 為 16 位元組或 32 個字元。替代方案 （ 方法 1) 並不常見，但僅使用 00 做為填補字元。

ISO 9797 方法 1 填補

未填補：
00000000170000000000000008400080008000084016051700000000093800000B03011203 (74 個字元或 37 個位元組)

填充：
0000000017000000000000000840008000800008401605170000000093800000B030112030000000
(80 個字元或 40 個位元組）

ISO 9797 方法 2 填補

未填補：
0000000017000000000000000840008000800008401605170000000093800000B1F220103000000
(80 個字元或 40 個位元組）

填充：
0000000017000000000000000840008000800008401605170000000093800000B1F220103000008000000
(88 個字元或 44 個位元組）

範例

Visa CVN10

Example

在此範例中，我們將驗證使用 Visa CVN10 產生的 ARQC。

如果 AWS 付款密碼編譯能夠驗證 ARQC，則會傳回 http/200。如果 ARCQ (Authorization Request Cryptogram) 未驗證，則會傳回 http/400 回應。

```
$ 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
 0000000017000000000000000840008000800008401605170000000093800000B03011203000000 \
--session-key-derivation-attributes='{"Visa":{"PanSequenceNumber":"01" \
,"PrimaryAccountNumber":"9137631040001422"}}'
```

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

Visa CVN18 和 Visa CVN22

Example

在此範例中，我們將驗證使用 Visa CVN18 或 CVN22 產生的 ARQC。CVN18 和 CVN22 之間的密碼編譯操作相同，但交易資料中包含的資料會有所不同。與 CVN10 相比，即使輸入相同，也會產生完全不同的密碼編譯。

如果 AWS 付款密碼編譯能夠驗證 ARQC，則會傳回 http/200。如果未驗證 ARCQ，則會傳回 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
 00000000170000000000000008400080008000084016051700000000093800000B1F22010300000000000
 \
00000000000000000000000000000000000000000000800000000000000000
--session-key-derivation-attributes='{"EmvCommon":
{"ApplicationTransactionCounter":"000B", \
"PanSequenceNumber":"01","PrimaryAccountNumber":"9137631040001422"}}'
```

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

產生和驗證 MAC

訊息驗證碼 (MAC) 通常用於驗證訊息的完整性（無論是否經過修改）。密碼編譯雜湊，例如 HMAC（以雜湊為基礎的訊息驗證碼）、CBC-MAC 和 CMAC（以密碼為基礎的訊息驗證碼），透過使用密碼編譯提供額外的 MAC 寄件者保證。HMAC 是以雜湊函數為基礎，而 CMAC 是以區塊加密為基礎。此服務也支援屬於 CBC-MACs 類型的 ISO9797 演算法 1 和 3。

此服務的所有 MAC 演算法會結合密碼編譯雜湊函數和共用私密金鑰。它們會接收訊息和私密金鑰，例如金鑰中的金鑰材料，並傳回唯一的標籤或 mac。如果訊息的某個字元變更，或私密金鑰變更，則產生的標籤會完全不同。透過要求私密金鑰，密碼編譯 MACs也提供真實性；如果沒有私密金鑰，就無法

產生相同的 mac。密碼編譯 MACs 有時稱為對稱簽章，因為它們的運作方式與數位簽章類似，但使用單一金鑰進行簽署和驗證。

AWS 付款密碼編譯支援多種類型的 MACs：

ISO9797 ALGORITHM 1

由 KeyUsage ISO9797_ALGORITHM1 表示。如果欄位不是區塊大小的倍數 (TDES 為 8 個位元組/16 個十六進位字元，AES 為 16 個位元組/32 個字元， AWS 付款密碼編譯會自動套用 ISO9797 填補方法 1。如果需要其他填補方法，您可以在呼叫服務之前套用它們。

ISO9797 ALGORITHM 3 （零售 MAC)

由 KeyUsage ISO9797_ALGORITHM3 表示。適用與演算法 1 相同的填補規則

ISO9797 演算法 5 (CMAC)

由 KeyUsage TR31_M6_ISO_9797_5_CMAC_KEY 表示

HMAC

由 KeyUsage TR31_M7_HMAC_KEY 表示，包括 HMAC_SHA224、HMAC_SHA256、HMAC_SHA384 和 HMAC_SHA512

AS2805.4.1 MAC

由 KeyUsage TR31_M0_ISO_16609_MAC_KEY 表示。如需 AS2805 的詳細資訊，請參閱 ???

DUKPT MAC

DUKPT MAC 通常用於確認往返付款終端機的訊息來源和承載。它使用 DUKPT 衍生技術衍生金鑰，然後執行 MAC。與此選項搭配使用的金鑰會以 TR31_B0_BASE_DERIVATION_KEY KeyUsage的 表示。

EMV MAC

EMV MAC 通常在 EMV 文件中稱為完整性金鑰。它使用 EMV 衍生技術衍生金鑰，然後在內部使用 ISO9797_ALGORITHM3。它通常用於將發行者指令碼傳送到晶片卡以進行重新程式設計。與此選項搭配使用的金鑰會以 TR31_E2_EMV_MKEY_INTEGRITY KeyUsage的 表示。如果您同時傳送指令碼和更新離線接腳，請參閱 GenerateMacEmvPinChange 以執行這兩個操作。

主題

- 產生 MAC
- 驗證 MAC

產生 MAC

產生 MAC API 用於驗證卡片相關資料，例如從卡片磁條追蹤資料，方法是使用已知的密碼編譯金鑰產生 MAC （訊息驗證碼），以便在傳送和接收方之間進行資料驗證。用於產生 MAC 的資料包括訊息資料、秘密 MAC 加密金鑰和 MAC 演算法，以產生唯一的 MAC 值進行傳輸。MAC 的接收者將使用相同的 MAC 訊息資料、MAC 加密金鑰和演算法來重現另一個 MAC 值，以進行比較和資料身分驗證。即使訊息的一個字元變更或用於驗證的 MAC 金鑰不同，產生的 MAC 值也不同。API 支援此操作的 ISO 9797-1 演算法 1 和 ISO 9797-1 演算法 3 MAC （使用靜態 MAC 金鑰和衍生的 DUKPT 金鑰）、HMAC 和 EMV MAC 加密金鑰。

的輸入值`message-data`必須是 hexBinary 資料。

如需此 API 所有選項的詳細資訊，請參閱 GenerateMac 和 VerifyMac。

選用參數 mac-length 可讓您截斷輸出值 （不過這也可以在程式碼中完成）。長度為 8 是指 8 個位元組或 16 個十六進位字元。

您可以透過呼叫 CreateKey 來建立 MAC AWS 金鑰，或呼叫 ImportKey 來匯入。

> **Note**
>
> CMAC 和 HMAC 演算法不需要填補。所有其他要求將資料填入演算法的區塊大小，這是 TDES 的 8 個位元組 (16 個十六進位字元） 和 AES 的 16 個位元組 (32 個十六進位字元） 的倍數。

範例

- 產生 HMAC
- 使用 ISO 9797-1 演算法 3 產生 MAC
- 使用 CMAC 產生 MAC
- 使用 DUKPT CMAC 產生 MAC

產生 HMAC

在此範例中，我們將使用 HMAC 演算法`HMAC_SHA256`和 HMAC 加密金鑰，為卡片資料身分驗證產生 HMAC （以雜湊為基礎的訊息驗證碼）。金鑰必須將 KeyUsage 設定為 `TR31_M7_HMAC_KEY`，並將 KeyModesOfUse 設定為 `Generate`。雜湊長度 （例如 256) 會在建立金鑰時定義，且無法修改。

選用的 mac-length 參數會修剪輸出 MAC，但也可以在服務之外執行。此值以位元組為單位，因此值 16 預期十六進位字串長度為 32。

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"
}
```

使用 ISO 9797-1 演算法 3 產生 MAC

在此範例中，我們將使用 ISO 9797-1 演算法 3 （零售 MAC) 來產生 MAC 以進行卡片資料驗證。金鑰必須將 KeyUsage 設定為 TR31_M3_ISO_9797_3_MAC_KEY，並將 KeyModesOfUse 設定為 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"
}
```

使用 CMAC 產生 MAC

當金鑰為 AES 時，CMAC 最常使用，但也支援 TDES。在此範例中，我們將使用 CMAC (ISO 9797-1 演算法 5) 產生 MAC，以使用 AES 金鑰進行卡片資料驗證。金鑰必須將 KeyUsage 設定為 TR31_M6_ISO_9797_5_CMAC_KEY，並將 KeyModesOfUse 設定為 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"
}
```

使用 DUKPT CMAC 產生 MAC

此範例使用 DUKPT（每個交易衍生的唯一金鑰）與 CMAC 產生 MAC，以進行卡片資料驗證。金鑰必須將 KeyUsage 設定為 TR31_B0_BASE_DERIVATION_KEY，且 KeyModesOfUse DeriveKey設定為 true。

DUKPT 金鑰會使用基本衍生金鑰 (BDK) 和金鑰序號 (KSN)，為每個交易衍生唯一的金鑰。

Example

```
$ aws payment-cryptography-data generate-mac --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/qnobl5lghrzunce6 --message-data
 "3b3130383834393030313037333934313533d32343038323236303030373030303f33" --generation-
attributes="DukptCmac={KeySerialNumber="932A6E954ABB32DD00000001",DukptKeyVariant=BIDIRECTIONAL
```

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

驗證 MAC

確認 MAC API 用於驗證 MAC（訊息驗證碼）以進行卡片相關資料身分驗證。它必須使用與產生 MAC 期間相同的加密金鑰，以重新產生 MAC 值進行身分驗證。您可以透過呼叫 CreateKey 來建立 MAC AWS 加密金鑰，或呼叫 ImportKey 來匯入。API 支援此操作的 DUKPT MAC、HMAC 和 EMV MAC 加密金鑰。

如果該值已經過驗證，則回應參數MacDataVerificationSuccessful會傳回 Http/200，否則Http/400會出現訊息指出 Mac verification failed。

範例

- 驗證 HMAC
- 使用 DUKPT CMAC 驗證 MAC

驗證 HMAC

在此範例中，我們將使用 HMAC 演算法HMAC_SHA256和 HMAC 加密金鑰，驗證卡片資料身分驗證的 HMAC （以雜湊為基礎的訊息驗證碼）。金鑰必須將 KeyUsage 設定為 TR31_M7_HMAC_KEY，且 KeyModesOfUse `Verify` 設定為 true。

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"
}
```

使用 DUKPT CMAC 驗證 MAC

在此範例中，我們將使用 DUKPT （每個交易衍生的唯一金鑰） 與 CMAC 進行卡資料驗證來驗證 MAC。金鑰必須將 KeyUsage 設定為 TR31_B0_BASE_DERIVATION_KEY，且 KeyModesOfUse `DeriveKey`設定為 true。DUKPT 金鑰會使用基本衍生金鑰 (BDK) 和金鑰序號 (KSN)，為每個交易衍生唯一的金鑰。DukptKeyVariant 的值必須與寄件者和接收者相符。當雙向使用單一金鑰時，通常會從終端機使用 REQUEST 到後端、從後端使用 VERIFY 到終端機和雙向使用。

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"
}
```

密碼編譯操作的有效金鑰

某些金鑰只能用於某些操作。此外，某些操作可能會限制金鑰使用的金鑰模式。如需允許的組合，請參閱下表。

> **Note**
>
> 雖然允許，某些組合可能會產生無法使用的情況，例如產生 CVV 代碼(generate)，但無法驗證它們(verify)。

主題

- GenerateCardData
- VerifyCardData
- GeneratePinData （適用於 VISA/ABA 配置）
- GeneratePinData （適用於 IBM3624)
- VerifyPinData （適用於 VISA/ABA 配置）
- VerifyPinData （適用於 IBM3624)
- 解密資料

- 加密資料
- 翻譯 PIN 資料
- 產生/驗證 MAC
- GenerateMacEmvPinChange
- VerifyAuthRequestCryptogram
- 匯入/匯出金鑰
- 未使用的金鑰類型

GenerateCardData

API 端點	密碼編譯操作或演算法	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
GenerateCardData	• AMEX_CARD_SECURITY_CODE_VERSION_1 • AMEX_CARD_SECURITY_CODE_VERSION_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	{ Generate = true } , { Generate = true , Verify = true }
GenerateCardData	• CARD_VERIFICATION_VALUE_1 • CARD_VERIFICATION_VALUE_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY	{ Generate = true } , { Generate = true , Verify = true }
GenerateCardData	• CARDHOLDER_AUTHENTICATION_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true }

API 端點	密碼編譯操作或演算法	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
GenerateCardData	• DYNAMIC_CARD_VERIFICATION_CODE	TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS	• TDES_2KEY	{ DeriveKey = true }
GenerateCardData	• DYNAMIC_CARD_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true }

VerifyCardData

密碼編譯操作或演算法	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
• AMEX_CARD_SECURITY_CODE_VERSION_1 • AMEX_CARD_SECURITY_CODE_VERSION_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	{ Generate = true }，{ Generate = true，Verify = true }
• CARD_VERIFICATION_VALUE_1 • CARD_VERIFICATION_VALUE_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY	{ Generate = true }，{ Generate = true，Verify = true }
• CARDHOLDER_AUTHENT	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true }

密碼編譯操作或演算法	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
ICATION_V ERIFICATI ON_VALUE			
• DYNAMIC_C ARD_VERIF ICATION_CODE	TR31_E4_E MV_MKEY_D YNAMIC_NUMBERS	• TDES_2KEY	{ DeriveKey = true }
• DYNAMIC_C ARD_VERIF ICATION_VALUE	TR31_E6_E MV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true }

GeneratePinData （適用於 VISA/ABA 配置）

`VISA_PIN or VISA_PIN_VERIFICATION_VALUE`

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
PIN 加密金鑰	TR31_P0_P IN_ENCRYP TION_KEY	• TDES_2KEY • TDES_3KEY	• { Encrypt = true， Wrap = true } • { 加密 = true、解密 = true、包裝 = true、取消包裝 = true } • { NoRestrictions = true }
PIN 產生金鑰	TR31_V2_V ISA_PIN_VERIFICATI ON_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true， Verify = true }

GeneratePinData （適用於 **IBM3624**)

IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN,
IBM3624_PIN_FROM_OFFSET)

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
PIN 加密金鑰	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	對於 IBM3624_NATURAL_PIN、IBM3624_RANDOM_PIN、IBM3624_PIN_FROM_OFFSET • { Encrypt = true，Wrap = true } • { 加密 = true、解密 = true、包裝 = true、取消包裝 = true } • { NoRestrictions = true } 對於 IBM3624_PIN_OFFSET • { Encrypt = true、Unwrap = true } • { 加密 = true、解密 = true、包裝 = true、取消包裝 = true }

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
			• { NoRestrictions = true }
PIN 產生金鑰	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true，Verify = true }

VerifyPinData（適用於 VISA/ABA 配置）

`VISA_PIN`

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
PIN 加密金鑰	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	• { Decrypt = true、Unwrap = true } • { 加密 = true、解密 = true、包裝 = true、取消包裝 = true } • { NoRestrictions = true }
PIN 產生金鑰	TR31_V2_VISA_PIN_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	• { Verify = true } • { Generate = true，Verify = true }

VerifyPinData（適用於 **IBM3624**）

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
PIN 加密金鑰	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	對於 IBM3624_NATURAL_PIN、IBM3624_RANDOM_PIN、IBM3624_PIN_FROM_OFFSET • { Decrypt = true、Unwrap = true } • { 加密 = true、解密 = true、包裝 = true、取消包裝 = true } • { NoRestrictions = true }
PIN 驗證金鑰	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	• { Verify = true } • { Generate = true，Verify = true }

解密資料

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
杜克普	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true } • { NoRestrictions = true }

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
EMV	TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	• { DeriveKey = true }
RSA	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• { Decrypt = true、Unwrap=true} • {Encrypt=true，Wrap=true，Decrypt = true，Unwrap=true}
對稱金鑰	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Decrypt = true、Unwrap=true} • {Encrypt=true，Wrap=true，Decrypt = true，Unwrap=true} • { NoRestrictions = true }

加密資料

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
杜克普	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true } • { NoRestrictions = true }

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
EMV	TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	• { DeriveKey = true }
RSA	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• { Encrypt = true，Wrap=true} • {Encrypt=true，Wrap=true，Decrypt = true，Unwrap=true}
對稱金鑰	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Encrypt = true，Wrap=true} • {Encrypt=true，Wrap=true，Decrypt = true，Unwrap=true} • { NoRestrictions = true }

翻譯 PIN 資料

Direction	金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
傳入資料來源	杜克普	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true } • { NoRestrictions = true }

Direction	金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
傳入資料來源	非DUKPT (PEK、AWK、IWK 等）	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Decrypt = true、Unwrap = true } • { Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true } • { NoRestrictions = true }
傳出資料目標	杜克普	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true } • { NoRestrictions = true }
傳出資料目標	非DUKPT (PEK、IWK、AWK 等）	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Encrypt = true， Wrap = true } • { Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true } • { NoRestrictions = true }

產生/驗證 MAC

MAC 金鑰用於建立訊息/資料主體的密碼編譯雜湊。不建議建立金鑰模式有限的金鑰，因為您無法執行相符的操作。不過，如果另一個系統打算執行操作對的另一半，您可以匯入/匯出只有一個操作的金鑰。

允許的金鑰用量	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
MAC 金鑰	TR31_M1_ISO_9797_1_MAC_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true，Verify = true } • { Verify = true } • { Generate = true }
MAC 金鑰（零售 MAC)	TR31_M1_ISO_9797_3_MAC_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true，Verify = true } • { Verify = true } • { Generate = true }
MAC 金鑰 (CMAC)	TR31_M6_ISO_9797_5_CMAC_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Generate = true } • { Generate = true，Verify = true } • { Verify = true } • { Generate = true }
MAC 金鑰 (HMAC)	TR31_M7_HMAC_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Generate = true } • { Generate = true，Verify = true } • { Verify = true }
MAC 金鑰 (AS2805)	TR31_M0_ISO_16609_MAC_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true，Verify = true } • { Verify = true }

GenerateMacEmvPinChange

GenerateMacEmvPinChange 結合了 EMV 離線 PIN 變更操作的 MAC 產生和 PIN 加密。此操作需要兩種不同的金鑰類型：MAC 產生的完整性金鑰和 PIN 加密的機密性金鑰。

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
安全傳訊完整性金鑰	TR31_E2_EMV_MKEY_INTEGRITY	• TDES_2KEY	• { NoRestrictions = true }
安全傳訊機密性金鑰	TR31_E1_EMV_MKEY_CONFIDENTIALITY	• TDES_2KEY	• { DeriveKey = true }
目前的 PIN PEK (PIN 加密金鑰)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Decrypt = true、Unwrap = true } • { Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true } • { NoRestrictions = true }
新的 PIN PEK (PIN 加密金鑰)	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Decrypt = true、Unwrap = true } • { Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true } • { NoRestrictions = true }

金鑰類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
ARQC 金鑰 Note 僅適用於 Visa 和 Amex 衍生方案。	TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS	• TDES_2KEY	• { DeriveKey = true }

VerifyAuthRequestCryptogram

允許的金鑰用量	EMV 選項	允許的金鑰演算法	允許的金鑰使用模式組合
• 選項 A • 選項 B	TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS	• TDES_2KEY	• { DeriveKey = true }

匯入/匯出金鑰

操作類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
TR-31 包裝金鑰	TR31_K1_KEY_BLOCK_PROTECTION_KEY TR31_K0_KEY_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Encrypt = true，Wrap = true }（僅限匯出） • { Decrypt = true、Unwrap = true }（僅限匯入） • { Encrypt = true、Decrypt = true、 Wrap =

操作類型	允許的金鑰用量	允許的金鑰演算法	允許的金鑰使用模式組合
			true、Unwrap = true }
匯入信任的 CA	TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE	• RSA_2048 • RSA_3072 • RSA_4096	• { Verify = true }
匯入用於非對稱加密的公有金鑰憑證	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• { Encrypt=true，Wrap=true }
用於金鑰協議演算法的金鑰，例如 ECDH	TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT	• ECC_NIST_P256 • ECC_NIST_P384 • ECC_NIST_P521	• { DeriveKey = true }

未使用的金鑰類型

AWS 付款密碼編譯目前未使用下列金鑰類型

- TR31_P1_PIN_GENERATION_KEY

常用案例

AWS 付款密碼編譯支援許多典型的付款密碼編譯操作。下列主題可做為如何將這些操作用於典型常見使用案例的指南。如需所有命令的清單，請檢閱 AWS 付款密碼編譯 API。

主題

- 發行者和發行者處理器
- 取得和付款引導程式

發行者和發行者處理器

發行者使用案例通常由幾個部分組成。本節依 函數組織 （例如使用接腳）。在生產系統中，金鑰通常範圍限定於指定的卡片儲存貯體，並在儲存貯體設定期間建立，而不是內嵌，如下所示。

主題

- 一般函數
- 網路特定函數

一般函數

主題

- 產生隨機接腳和相關聯的 PVV，然後驗證值
- 產生或驗證指定卡片的 CVV
- 產生或驗證特定卡片的 CVV2
- 產生或驗證特定卡片的 iCVV
- 驗證 EMV ARQC 並產生 ARPC
- 產生和驗證 EMV MAC
- 針對 PIN 變更產生 EMV MAC

產生隨機接腳和相關聯的 PVV，然後驗證值

主題

- 建立金鑰 （ 金鑰 ）
- 產生隨機 PIN 碼、產生 PVV 並傳回加密的 PIN 碼和 PVV
- 使用 PVV 方法驗證加密的 PIN
- 使用者受管 PINs

建立金鑰 （ 金鑰 ）

為了產生隨機接腳和 PVV，您需要兩個金鑰：用於產生 PVV 的接腳驗證金鑰 (PVK)，以及用於加密接腳的接腳加密金鑰。接腳本身會在服務內安全地隨機產生，且與任一金鑰的密碼編譯無關。

PGK 必須是以 PVV 演算法本身為基礎的演算法 TDES_2KEY 金鑰。PEK 可以是 TDES_2KEY、TDES_3KEY 或 AES_128。在此情況下，由於 PEK 旨在供系統內部使用，因此 AES_128 是不錯的選擇。如果 PEK 用於與其他系統 （ 例如，卡網路、收單機構、ATMs) 交換，或作為遷移的一部分進行移動，則基於相容性原因，TDES_2KEY 可能是更適當的選擇。

建立 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/ivi5ksfsuplneuyt。在下一個步驟中，您需要用到。

建立 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```

請注意KeyArn代表金鑰的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/ov6icy4ryas4zcza。在下一個步驟中，您需要用到。

產生隨機 PIN 碼、產生 PVV 並傳回加密的 PIN 碼和 PVV

Example

在此範例中，我們將產生新的 （隨機） 4 位數接腳，其中輸出將是加密的 PIN block(PinData.PinBlock) 和 PVV(pinData.VerificationValue). 金鑰輸入為 PAN、 Pin Verification Key（也稱為 PIN 產生金鑰）、 Pin Encryption Key和 PIN 區塊格式。

此命令要求金鑰類型為 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"
            }
        }
```

使用 PVV 方法驗證加密的 PIN

Example

在此範例中，我們將驗證指定 PAN 的 PIN。PIN 通常由持卡人或使用者在驗證的交易期間提供，並與檔案上的值進行比較（持卡人的輸入會以終端機或其他上游提供者的加密值提供）。為了驗證此輸入，也會在執行時間提供下列值 - 加密的 PIN、用來加密輸入 PIN 的金鑰（通常稱為 IWK)，PAN以及要驗證的值 (PVV或 `PIN offset`)。

如果 AWS 付款密碼編譯能夠驗證 PIN 碼，則會傳回 http/200。如果未驗證 PIN 碼，則會傳回 http/400。

```
$ 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",
}
```

使用者受管 PINs

主題

- 概觀
- 建立金鑰
- 設定使用者可選取的 PIN
- 重設 PIN （綠色 PIN)
- 顯示 PIN

概觀

AWS 付款密碼編譯支援使用者受管（客戶受管）PIN 流程，其中 PIN 會透過您的客戶後端在持卡人裝置和服務之間安全地交換。裝置和服務使用橢圓曲線 Diffie-Hellman (ECDH) 金鑰協議，在每一端衍生共用對稱金鑰，且該金鑰可保護 ISO 格式 4 PIN 區塊。由於金鑰衍生而非交換，因此透明 PIN 和透明 PIN 區塊永遠不會傳輸到裝置之外。

這些流程涉及三方。持卡人裝置會產生其 EC 金鑰對、衍生 ECDH AES-128 金鑰、加密或解密 ISO 格式的 4 PIN 區塊，並保留清晰的 PIN；沒有 AWS 登入資料。客戶後端（您的發行者或處理器伺服器）會保留 AWS 登入資料、呼叫 AWS 付款密碼編譯、簽署裝置憑證，以及存放 PVV. AWS Payment 密碼編譯執行密碼編譯操作。裝置只會與客戶後端通訊，而客戶後端會對 AWS 付款密碼進行所有呼叫。

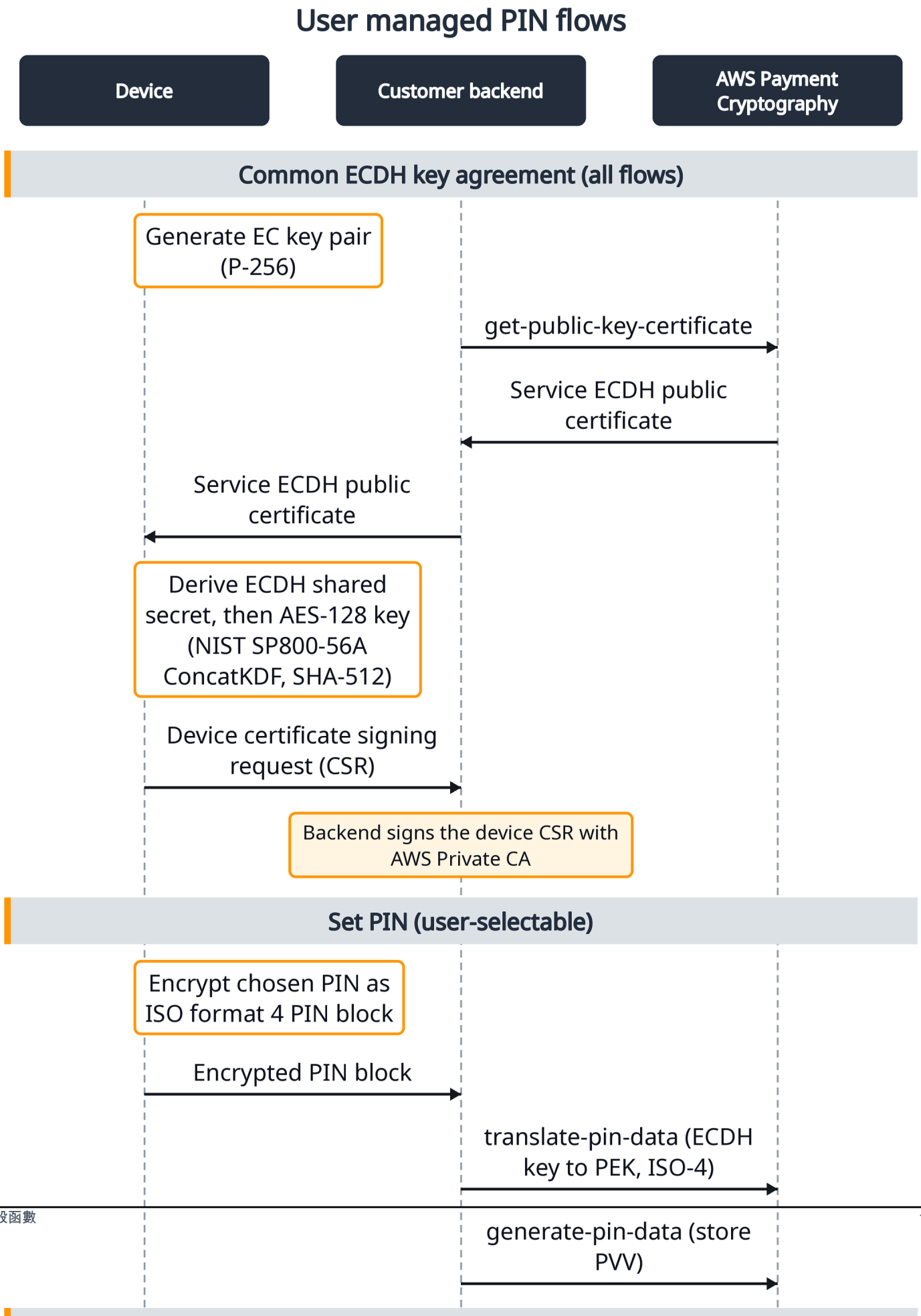
本主題涵蓋三個流程：

- 使用者可選取的 PIN（設定 PIN) - 持卡人會在裝置上選擇自己的 PIN。
- 綠色 PIN 碼（重設 PIN 碼）- 服務會產生交付給持卡人的隨機 PIN 碼，例如在第一次發行或忘記 PIN 碼重設期間。
- 顯示 PIN - 針對狹窄的使用案例，現有的 PIN 區塊會傳回至授權的裝置。

這三個流程共用相同的金鑰設定和相同的 ECDH 裝置協議機制，如建立金鑰中所述。如需金鑰協議本身的詳細資訊，請參閱使用非對稱技術 (ECDH) 匯入金鑰。

> **Note**
>
> 這些流程只會在裝置上處理清晰PINs 和 PIN 區塊，絕不會傳輸到 AWS 付款密碼。請先與您的安全團隊仔細檢閱每個流程，再進行部署。

下圖顯示使用者受管 PIN 流程。


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)

如需使用適用於 Python 的 AWS 開發套件執行這些流程的完整範例，請參閱 ECDH PIN 集，並在 GitHub 的範例儲存庫中顯示流程範例。 AWS GitHub

> **Note**
>
> 典型的流程包括執行 ECDH 金鑰協議，並加密或解密 ISO 格式 4 PIN 區塊的持卡人裝置。如果您已透過其他方式保護裝置和客戶後端之間的 PINs，則可以直接在客戶後端和 AWS 付款密碼編譯之間使用這些流程的修改版本。在該變體中，客戶後端會執行 ECDH 金鑰協議和 PIN 區塊操作，以取代裝置。

建立金鑰

這三個流程都使用相同的一組金鑰：與裝置簽訂金鑰協議的 ECDH 協議金鑰、產生 PVV 的 PIN 碼產生/驗證金鑰 (PGK)，以及將 PIN 區塊轉譯為後端儲存的 PIN 加密金鑰 (PEK)。您也可以匯入裝置憑證授權單位 (CA) 的公有金鑰，讓 AWS Payment Cryptography 信任翻譯期間呈現的裝置憑證。

建立 ECDH 協議金鑰。此金鑰用於衍生與裝置共用的對稱金鑰。

```
$ 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}'
```

回應會回傳請求參數，並包含 ECDH 金鑰的 ARN。

```
{
    "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"
    }
}
```

建立用於產生 PVV 的接腳產生/驗證金鑰 (PGK)。

```
$ 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"
    }
}
```

建立 PIN 加密金鑰 (PEK)，裝置 PIN 區塊會轉換為該金鑰以進行後端儲存。

```
$ 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"
    }
}
```

匯入裝置 CA 的公有金鑰，讓 AWS 付款密碼編譯可以驗證稍後翻譯呼叫中呈現的裝置憑證。<base64-CA-cert> 以 base64 編碼的 CA 憑證取代 。

```
$ 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"
    }
}
```

請記下每個傳回的 KeyArn。以下範例使用以下常值預留位置：

- ECDH 協議金鑰： arn:aws:payment-cryptography:us-east-1:111122223333:key/xxxxxxxxxxxxxxxx
- 匯入的 CA 公有金鑰： 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

在持卡人裝置（通常是行動電話或類似裝置）上，下列步驟會在裝置的應用程式碼中針對每個設定 PIN、重設 PIN 和顯示 PIN 流程執行：在曲線 P-256 上產生 EC 金鑰對；從客戶後端取得服務 ECDH 公有憑證；衍生 ECDH 共用秘密，然後使用具有 SharedInformation值的 NIST SP800-56A ConcatKDF (SHA-512) 使用 16 位元組 AES-128 金鑰；並使用衍生金鑰來加密或解密 ISO 格式 4 PIN 區塊。裝置沒有 AWS 登入資料，而且不會直接呼叫 AWS 付款密碼編譯；它會與客戶後端交換此資料，以便進行服務呼叫。

由於裝置通常是手機，因此其 EC 金鑰對和憑證會在第一次使用時佈建，並在之後定期重新整理。裝置會產生 EC 金鑰對和憑證簽署請求 (CSR)，而客戶後端會使用 AWS 私有 CA 或類似 CA 簽署該 CSR。客戶後端會在其translate-pin-data呼叫中使用產生的簽署裝置憑證，而 AWS 付款密碼編譯會根據您匯入的 CA 公有金鑰進行驗證。

客戶後端會取得服務 ECDH 公有憑證，並將其提供給裝置。在 KeyCertificate 欄位中，憑證會以 base64 編碼的 PEM 傳回，而 base64 解碼為 PEM 檔案。裝置在衍生共用秘密時，會使用此憑證做為服務公有金鑰。

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

> **Note**
>
> SharedInformation 是在裝置上產生的 32 個隨機位元組，並以十六進位字串的形式透過客戶後端傳遞至 AWS 付款密碼編譯。它必須符合裝置在 KDF 中使用的值，否則衍生的金鑰將不一致。

設定使用者可選取的 PIN

在設定 PIN 流程中，持卡人會在裝置上選擇自己的 PIN 碼 （ 例如 1234)。裝置衍生 ECDH 共用金鑰，將選擇的 PIN 加密為該金鑰下的 ISO 格式 4 PIN 區塊，並將加密的 PIN 區塊、其憑證簽署請求 (CSR) 和共用資訊傳送給客戶後端。客戶後端會簽署 CSR，然後執行兩次資料平面呼叫。此範例使用範例 PAN 171234567890123。顯示的 PIN 區塊、憑證和共用資訊值是虛構預留位置。

首先，將受 ECDH 保護的 PIN 區塊轉譯至伺服器 PEK。傳入金鑰是 ECDH 協議金鑰，以 Diffie-Hellman 參數和簽章的裝置憑證包裝；傳出金鑰是 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}'
```

回應包含以 ISO 格式 4 在 PEK 下重新加密的 PIN 區塊。

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

接著，產生 PEK 加密 PIN 區塊的 PVV。PGK 會產生 PVV，而 PEK 會識別提供的 PIN 區塊所加密的金鑰。

```
$ 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}'
```

回應包含 中的 PVV `PinData.VerificationValue`和加密的 PIN 區塊。將 PVV 存放在客戶後端，以便稍後驗證 PIN 碼。如需驗證，請參閱產生隨機接腳和相關聯的 PVV，然後驗證值。

```
{
    "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"
    }
}
```

如需詳細資訊，請參閱 API 參考指南中的 TranslatePinData 和 GeneratePinData。

重設 PIN （綠色 PIN)

在重設 PIN （綠色 PIN) 流程中， AWS 付款密碼編譯會產生隨機 PIN，客戶後端會傳回裝置以顯示給持卡人，例如在第一次發行或忘記 PIN 重設期間。裝置會執行建立金鑰中所述的相同 ECDH 設定，而客戶後端會代其進行資料平面呼叫。此範例使用範例 PAN 171234567890123；PIN 區塊和共用資訊值是虛構預留位置。

首先，產生隨機 PIN 及其 PVV，在 PEK 下加密 PIN 區塊。

```
$ 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}'
```

回應包含 中的 PVV `PinData.VerificationValue`和 PEK 加密的 PIN 區塊 (ISO 格式 4)。將 PVV 存放在客戶後端，以供日後驗證。

```
{
    "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"
    }
}
```

接著，將 PEK 加密的 PIN 區塊 (ISO 格式 4) 轉譯為裝置 ECDH 衍生金鑰 (ISO 格式 4)，以便只有請求的裝置才能讀取它。

```
$ 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}'
```

回應包含以 ISO 格式 4 在裝置 ECDH 衍生金鑰下重新加密的 PIN 區塊。

```
{
```

```
    "PinBlock": "A1B2C3D4E5F60718",
    "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxxx",
    "KeyCheckValue": "1A2B3C"
}
```

客戶後端會將 ECDH 加密的 PIN 區塊傳回裝置。裝置會使用其 ECDH 衍生的金鑰（在裝置應用程式碼中）解密傳回的 PIN 區塊，並向持卡人顯示 PIN。由於金鑰僅在該裝置上衍生，因此沒有其他方可以解密 PIN 區塊。

如需詳細資訊，請參閱 API 參考指南中的 GeneratePinData 和 TranslatePinData。

顯示 PIN

顯示 PIN 流程會將現有的 PIN 區塊傳回持卡人裝置，讓裝置可以顯示或以其他方式顯示 PIN。裝置會執行建立金鑰中所述的相同 ECDH 設定，而客戶後端會將存放的 PEK 加密 PIN 區塊轉譯回 ECDH 衍生金鑰，並將其傳回裝置。此範例使用範例 PAN 171234567890123；PIN 區塊和共用資訊值是虛構預留位置。

> **Note**
>
> 考慮驗證持卡人並讓他們設定新的 PIN 碼，而不是公開現有的 PIN 碼。值得注意的例外是使用離線 PIN 的國家/地區，例如歐洲，其中 PIN 也存放在晶片卡上。在這些情況下，顯示現有的 PIN 比重新程式設計卡片更簡單。

將儲存的 PEK 加密 PIN 區塊 (ISO 格式 4) 轉換為裝置 ECDH 衍生金鑰 (ISO 格式 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}'
```

回應包含以 ISO 格式 4 在裝置 ECDH 衍生金鑰下重新加密的 PIN 區塊。

```
{
    "PinBlock": "A1B2C3D4E5F60718",
    "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/
xxxxxxxxxxxxxxxx",
    "KeyCheckValue": "1A2B3C"
}
```

客戶後端會將 ECDH 加密的 PIN 區塊傳回裝置。裝置會使用其 ECDH 衍生的金鑰 （ 在裝置應用程式碼中 ） 解密傳回的 PIN 區塊，以顯示 PIN。由於金鑰僅在該裝置上衍生，因此沒有其他方可以解密 PIN 區塊。

如需詳細資訊，請參閱 API 參考指南中的 TranslatePinData。

產生或驗證指定卡片的 CVV

CVV 或 CVV1 是傳統上內嵌在卡片磁性條紋中的值。它與 CVV2 不同 （ 持卡人可見並用於線上購買 ）。

第一步是建立金鑰。在本教學課程中，您會建立 CVK 雙長度 3DES (2KEY TDES) 金鑰。

> **Note**
>
> CVV、CVV2 和 iCVV 全都使用類似的演算法，但會改變輸入資料。所有 都使用相同的金鑰類型 TR31_C0_CARD_VERIFICATION_KEY，但建議針對每個用途使用不同的金鑰。這些可以使用別名和/或標籤來區分，如以下範例所示。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```


請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/r52o3wbqxyf6qlqr。在下一個步驟中，您需要用到。

產生 CVV

Example

在此範例中，我們將為指定的 PAN 產生 CVV，其輸入為 PAN，服務碼 （ 如 ISO/IEC 7813 所定義 ） 為 121，卡片過期日期。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ 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"
                }
```

驗證 CVV

Example

在此範例中，我們將使用 CVK、PAN、服務代碼 121、卡片過期日期和交易期間提供的 CVV 輸入來驗證指定 PAN 的 CVV。

如需所有可用的參數，請參閱《 API 參考指南》中的 CardVerificationValue1。

> **Note**
>
> CVV 不是使用者輸入的值 （ 例如 CVV2)，但通常內嵌在磁條上。應考慮是否應一律在提供時驗證。

```
$ 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"
}
```

產生或驗證特定卡片的 CVV2

CVV2 是傳統上在卡片背面提供的值，用於線上購買。對於虛擬卡，它也可能顯示在應用程式或螢幕上。密碼編譯方式與 CVV1 相同，但具有不同的服務代碼值。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/7f7g4spf3xcklhzu。在下一個步驟中，您需要用到。

產生 CVV2

Example

在此範例中，我們將為輸入 PAN和卡片過期日期的指定 PAN 產生 CVV2。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue2。

```
$ 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"
                    }
```

驗證 CVV2

Example

在此範例中，我們將使用 CVK 的輸入、PAN卡片過期日期和交易期間提供的 CVV 來驗證指定 PAN 的 CVVCVV2。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue2。

> **Note**
>
> CVV2 和其他輸入是使用者輸入的值。因此，這不一定是定期無法驗證問題的跡象。

```
$ 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"
   }
```

產生或驗證特定卡片的 iCVV

iCVV 使用與 CVV/CVV2 相同的演算法，但 iCVV 內嵌在晶片卡中。其服務代碼為 999。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/c7dsi763r6s7lfp3。在下一個步驟中，您需要用到。

產生 iCVV

Example

在此範例中，我們將為輸入為 PAN、服務碼（如 ISO/IEC 7813 所定義）為 999 且卡片過期日期的指定 PAN 產生 iCVV。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ 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"
                    }
```

驗證 iCVV

Example

針對驗證，輸入為 CVK、PAN、服務代碼 999、卡片過期日期，以及交易期間提供的 iCVV 以進行驗證。

如需所有可用的參數，請參閱《 API 參考指南》中的 CardVerificationValue1。

> **Note**
>
> iCVV 不是使用者輸入的值（例如 CVV2），但通常內嵌在 EMV/晶片卡上。應考慮是否應一律在提供時驗證。

```
$ 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"
   }
```

驗證 EMV ARQC 並產生 ARPC

ARQC (Authorization Request Cryptogram) 是由 EMV（晶片）卡產生的密碼編譯，用於驗證交易詳細資訊以及授權卡的使用。它包含來自卡片、終端機和交易本身的資料。

在後端的驗證時間，相同的輸入會提供給 AWS 付款密碼編譯，密碼編譯會在內部重新建立，並與交易提供的值進行比較。就這個意義而言，它類似於 MAC。EMV 4.4 Book 2 定義此函數的三個層面 - 金鑰衍生方法（稱為通用工作階段金鑰 - CSK），以產生一次性交易金鑰、最低承載和產生回應的方法 (ARPC)。

個別卡片配置可以指定要納入的其他交易欄位，或這些欄位出現的順序。其他（通常已棄用）結構描述特定的衍生結構描述也存在，並涵蓋在本文件的其他部分。

如需詳細資訊，請參閱 API 指南中的 VerifyCardValidationData。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
                }
            }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/pw3s6nl62t5ushfk。在下一個步驟中，您需要用到。

產生 ARQC

ARQC 僅由 EMV 卡產生。因此， AWS 付款密碼編譯沒有產生此類承載的設施。基於測試目的，許多程式庫可在線上取得，這些程式庫可以產生適當的承載，以及各種方案通常提供的已知值。

驗證 ARQC

Example

如果 AWS 付款密碼編譯能夠驗證 ARQC，則會傳回 http/200。驗證 ARQC 之後，可以選擇提供 ARPC （回應） 並包含在回應中。

```
$ 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"
}
```

產生和驗證 EMV MAC

EMV MAC 是 MAC，使用 EMV 衍生金鑰的輸入，然後對產生的資料執行 ISO9797-3 （零售） MAC。EMV MAC 通常用於將命令傳送至 EMV 卡，例如解除封鎖指令碼。

> **Note**
>
> AWS 付款密碼編譯不會驗證指令碼的內容。如需要包含的特定命令詳細資訊，請參閱您的方案或卡片手冊。

如需詳細資訊，請參閱 API 指南中的 MacAlgorithmEmv。

主題

- 建立金鑰
- 產生 EMV MAC

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
        }
```

```
    }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/pw3s6nl62t5ushfk。在下一個步驟中，您需要用到。

產生 EMV MAC

典型的流程是後端程序會產生 EMV 指令碼（例如卡片解除封鎖），使用此命令簽署它（衍生特定卡片的一次性金鑰），然後傳回 MAC。然後，命令 + MAC 會傳送至要套用的卡片。將命令傳送至卡片超出 AWS 付款密碼編譯的範圍。

> **Note**
>
> 此命令適用於未傳送加密資料（例如 PIN) 的命令。EMV Encrypt 可以與此命令結合，在呼叫此命令之前將加密的資料附加到發行者指令碼

訊息資料

訊息資料包含 APDU 標頭和命令。雖然這可能因實作而有所不同，但此範例是 unblock (84 24 00 00 08) 的 APDU 標頭，後面接著 ATC (0007)，然後是先前交易的 ARQC (999E57FD0F47CACE)。服務不會驗證此欄位的內容。

工作階段金鑰衍生模式

此欄位定義工作階段金鑰的產生方式。EMV_COMMON_SESSION_KEY 通常用於新實作，而 EMV2000 | AMEX | MASTERCARD_SESSION_KEY | VISA 也可以使用。

MajorKeyDerivationMode

EMV 定義模式 A、B 或 C。模式 A 是最常見的，而 AWS 付款密碼編譯目前支援模式 A 或模式 B。

PAN

帳戶號碼，通常在晶片欄位 5A 或 ISO8583 欄位 2 中提供，但也可以從卡片系統擷取。

PSN

卡片序號。如果未使用，請輸入 00。

SessionKeyDerivationValue

這是每個工作階段衍生資料的 。它可以是欄位 9F26 的最後一個 ARQC(ApplicationCryptogram)，也可以是 9F36 的最後一個 ATC，具體取決於衍生方案。

填補

填補會自動套用，並使用 ISO/IEC 9797-1 填補方法 2。

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"
}
```

針對 PIN 變更產生 EMV MAC

EMV PIN 變更結合了兩個操作：為發行者指令碼產生 MAC，以及為 EMV 晶片卡上的離線 PIN 變更加密新的 PIN。只有在將 PIN 碼存放在晶片卡上的某些國家/地區（這對於歐洲國家/地區來說很常見），才需要此命令。當持卡人需要變更其 PIN 碼，且新 PIN 碼必須與 MAC 一起安全地傳輸至卡片，以驗證命令的真實性時，通常會使用此功能。

> **Note**
>
> 如果您只需要將命令傳送至卡片，但不需要變更 PIN，請考慮改用 ARPC CSU 或產生 EMV MAC 命令。

如需詳細資訊，請參閱 API 指南中的 GenerateMacEmvPinChange。

針對 PIN 變更產生 EMV MAC 和加密 PIN

此操作需要兩個金鑰：用於產生 MAC 的 EMV 完整性金鑰 (KeyUsage：TR31_E2_EMV_MKEY_INTEGRITY)，以及用於 PIN 加密的 EMV 機密性金鑰 (KeyUsage：TR31_E4_EMV_MKEY_CONFIDENTIALITY)。典型的流程是後端程序會產生 EMV PIN 變更指令碼，其中包含發行者指令碼的 MAC 和加密的新 PIN。然後，命令和加密的 PIN 會傳送到卡片以更新離線 PIN。將命令傳送至卡片超出 AWS 付款密碼編譯的範圍。

訊息資料

訊息資料包含發行者指令碼的 APDU 命令。服務不會驗證此欄位的內容。

新的加密 PIN 區塊

將傳送至卡片的新加密 PIN 區塊。這必須使用 PIN 加密金鑰做為加密值提供。

新的 PIN PEK 識別符

用來加密新 PIN 碼的金鑰，然後再傳遞至此 API。

安全傳訊完整性金鑰

用於產生 MAC 的 EMV 完整性金鑰 (KeyUsage： TR31_E2_EMV_MKEY_INTEGRITY)。

安全傳訊機密性金鑰

用於 PIN 加密的 EMV 機密性金鑰 (KeyUsage： TR31_E4_EMV_MKEY_CONFIDENTIALITY)。

MajorKeyDerivationMode

EMV 定義了模式 A、B 或 C。模式 A 是最常見的，而 AWS 付款密碼編譯目前支援模式 A 或模式 B。

Mode

加密模式，通常是用於 PIN 變更操作的 CBC。

PAN

帳戶號碼，通常在晶片欄位 5A 或 ISO8583 欄位 2 中提供，但也可以從卡片系統擷取。

PanSequenceNumber

卡片序號。如果未使用，請輸入 00。

ApplicationCryptogram

這是每個工作階段衍生資料，通常是欄位 9F26 的最後一個 ARQC。

PinBlockLengthPosition

指定 PIN 區塊長度的編碼位置。通常設定為 NONE。如果您不確定，請檢查您的卡片方案規格。

PinBlockPaddingType

指定 PIN 區塊的填補類型。通常設定為 NO_PADDING。如果您不確定，請檢查您的卡片方案規格。

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"
}
```

網路特定函數

主題

- Visa 特定函數
- Mastercard 特定函數
- American Express 特定函數
- JCB 特定函數
- UnionPay 特定函數

Visa 特定函數

主題

- ARQC - CVN18/CVN22
- ARQC - CVN10
- 3DS CAVV V7
- dCVV （動態卡驗證值） - CVN17

ARQC - CVN18/CVN22

CVN18 和 CVN22 使用金鑰衍生的 CSK 方法。確切的交易資料在這兩種方法之間有所不同 - 如需建構交易資料欄位的詳細資訊，請參閱方案文件。

ARQC - CVN10

CVN10 是一種較舊的 EMV 交易 Visa 方法，使用每個卡片金鑰衍生而非工作階段（每個交易）衍生，也使用不同的承載。如需承載內容的相關資訊，請聯絡 方案以取得詳細資訊。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
        }
    }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/pw3s6nl62t5ushfk。在下一個步驟中，您需要用到。

驗證 ARQC

Example

在此範例中，我們將驗證使用 Visa CVN10 產生的 ARQC。

如果 AWS 付款密碼編譯能夠驗證 ARQC，則會傳回 http/200。如果未驗證 arqc，則會傳回 http/400 回應。

```
$ 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

對於 Visa Secure (3DS) 交易，發行者存取控制伺服器 (ACS) 會產生 CAVV （持卡人身分驗證值）。CAVV 是持卡人身分驗證發生的證據，對於每個身分驗證交易都是唯一的，並由授權訊息中的取得者提供。CAVV v7 會將交易的其他資料繫結至核准，包括商家名稱、購買金額和購買日期等元素。透過這種方式，它實際上是交易承載的密碼編譯雜湊。

CAVV V7 以密碼編譯方式使用 CVV 演算法，但輸入都已變更/重新使用。有關如何產生輸入以產生 CAVV V7 承載，請參閱適當的第三方/ Visa 文件。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```

請注意KeyArn代表金鑰的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/dnaeyrjgdjjtw6dk。在下一個步驟中，您需要用到。

產生 CAVV V7

Example

在此範例中，我們將為具有規格中指定輸入的特定交易產生 CAVV V7。請注意，對於此演算法，欄位可能會重複使用/重新使用，因此不應假設欄位標籤符合輸入。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ 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"
            }
```

驗證 CAVV V7

Example

為了進行驗證，輸入為 CVK、計算的輸入值，以及在交易期間提供的 CAVV 以進行驗證。

如需所有可用的參數，請參閱《 API 參考指南》中的 CardVerificationValue1。

> **Note**
>
> CAVV 不是使用者輸入的值 （ 例如 CVV2)，而是由發行者 ACS 計算。應考慮是否應一律在提供時驗證。

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

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

dCVV （ 動態卡驗證值 ） - CVN17

dCVV （ 動態卡驗證值 ） 是 Visa 特定的動態密碼編譯，用於非接觸式 EMV 交易。它稱為早期 EMV，並透過為每個交易產生唯一的驗證值來提供增強的安全性。dCVV 使用輸入，包括主帳戶號碼 (PAN)、PAN 序號 (PSN)、應用程式交易計數器 (ATC)、卡片過期日期和服務代碼。它在某些地方仍然使用，但大部分已被 CVN18 等其他演算法取代。

如需所有可用的參數，請參閱 API 參考指南中的 DynamicCardVerificationValue。

要DynamicCardVerificationValue映射到下列 Visa 授權和 EMV 交易資料的輸入：

`ApplicationTransactionCounter` (ATC)

由晶片維護的交易計數器，每次交易都會遞增。在 EMV 資料中，這是標籤 9F36。

`PanSequenceNumber`

區分共用相同 之卡片的 PAN 序號PAN。在 EMV 資料中，這是標籤 5F34。

`CardExpiryDate`

卡片的應用程式過期日期 (EMV 標籤 5F24)，以其他卡片驗證呼叫所使用的相同格式提供。

ServiceCode

卡片軌道 2（同等）資料中攜帶的三位數服務代碼。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/mw7dn3qxvkfh8ztc。在下一個步驟中，您需要用到。

產生 dCVV

Example

在此範例中，我們將為非接觸式 EMV 交易產生 dCVV。輸入包括 PAN、PAN 序號、應用程式交易計數器、卡片過期日期和服務代碼。

```
$ 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"
}
```

驗證 dCVV

Example

在此範例中，我們將驗證交易期間提供的 dCVV。必須提供用於產生 的相同輸入以進行驗證。

如果 AWS 付款密碼編譯能夠驗證，則會傳回 http/200。如果未驗證值，則會傳回 http/400 回應。

```
$ 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 特定函數

主題

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

DCVC3

DCVC3 會早於 EMV CSK 和 Mastercard CVN12 結構描述，並代表使用動態金鑰的另一種方法。它有時也會重新用於其他使用案例。在此配置中，輸入為 PAN、PSN、Track1/Track2 資料、無法預測的數字和交易計數器 (ATC)。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
        }
    }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/hrh6qgbi3sk4y3wq。在下一個步驟中，您需要用到。

產生 DCVC3

Example

雖然 DCVC3 通常是由晶片卡產生，但也可以手動產生，例如在此範例中

```
$ 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"
  }
```

驗證 DCVC3

Example

在此範例中，我們將驗證 DCVC3。請注意，ATC 應做為十六進位號碼提供，例如，計數器 11 應表示為 000B。此服務預期有 3 位數 DCVC3，因此如果您已儲存 4 （或 5) 位數的值，只需截斷左側字元，直到您有 3 位數為止 （例如 15321 應該導致驗證資料值為 321)。

如果 AWS 付款密碼編譯能夠驗證，則會傳回 http/200。如果未驗證值，則會傳回 http/400 回應。

```
$ aws payment-cryptography-data verify-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk
 --primary-account-number=5413123456784808 --verification-attributes
 DynamicCardVerificationCode='{ApplicationTransactionCounter=000B,TrackData=524106000000069D13
 --validation-data 398
```

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

ARQC - CVN14/CVN15

CVN14 和 CVN15 使用金鑰衍生的 EMV CSK 方法。確切的交易資料在這兩種方法之間有所不同 - 如需建構交易資料欄位的詳細資訊，請參閱方案文件。

ARQC - CVN12/CVN13

CVN12 和 CVN13 是適用於 EMV 交易的較舊 Mastercard 特定方法，可將無法預測的數字納入每個交易衍生，也使用不同的承載。如需承載內容的相關資訊，請聯絡 機制。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/pw3s6nl62t5ushfk。在下一個步驟中，您需要用到。

驗證 ARQC

Example

在此範例中，我們將驗證使用 Mastercard CVN12 產生的 ARQC。

如果 AWS 付款密碼編譯能夠驗證 ARQC，則會傳回 http/200。如果未驗證 arqc，則會傳回 http/400 回應。

```
$ 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 SPA2 AAV

SPA2（安全付款應用程式）AAV（帳戶身分驗證值）用於 Mastercard 3DS 交易（也稱為 Mastercard 密度檢查）。它為使用 HMAC 型 MAC 產生的電子商務交易提供密碼編譯身分驗證。使用交易特定資料和共用私密金鑰產生 AAV。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
        }
    }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-west-2：111122223333：key/q5vjtshsg67cz5gn。在下一個步驟中，您需要用到。

產生 SPA2 AAV

Example

在此範例中，我們將使用 HMAC MAC 產生產生 SPA2 AAV 的發行者身分驗證值 (IAV) 元件。訊息資料包含要驗證的交易特定資訊。訊息資料的格式應遵循 Mastercard 的 SPA2 規格，此範例並未涵蓋。

> **Note**
>
> 請檢閱 Mastercard 規格的格式，以將 IAV 插入 AAV 值。

```
$ 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"
}
```

驗證 SPA2 AAV

Example

在此範例中，我們將驗證 SPA2 AAV。提供相同的訊息資料和 MAC 值以供驗證。

如果 AWS 付款密碼編譯能夠驗證 MAC，則會傳回 http/200。如果未驗證 MAC，則會傳回 http/400 回應。

```
$ 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 特定函數

主題

- CSC1
- CSC2
- iCSC
- 3DS AEVV

CSC1

CSC 第 1 版也稱為 Classic CSC 演算法。服務可以提供 3、4 或 5 位數的號碼。

如需所有可用的參數，請參閱 API 參考指南中的 AmexCardSecurityCodeVersion1。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
                }
            }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/esh6hn7pxdttzgq。在下一個步驟中，您需要用到。

產生 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"
  }
```

驗證 CSC1

Example

在此範例中，我們將驗證 CSC1。

如果 AWS 付款密碼編譯能夠驗證，則會傳回 http/200。如果未驗證值，則會傳回 http/400 回應。

```
$ 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 第 2 版也稱為增強型 CSC 演算法。服務可以提供 3、4 或 5 位數的號碼。CSC2 的服務碼通常是 000。

如需所有可用的參數，請參閱 API 參考指南中的 AmexCardSecurityCodeVersion2。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
            }
        }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/erlm445qvunmvoda。在下一個步驟中，您需要用到。

產生 CSC2

在此範例中，我們將產生長度為 4 的 CSC2。CSC 可以產生長度為 3、4 或 5。對於 American Express，PANs應為 15 位數，並以 34 或 37 開頭。過期日期通常格式為 YYMM。服務代碼可能有所不同 - 檢閱您的手冊，但典型值為 000、201 或 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"
}
```

驗證 CSC2

Example

在此範例中，我們將驗證 CSC2。

如果 AWS 付款密碼編譯能夠驗證，則會傳回 http/200。如果未驗證值，則會傳回 http/400 回應。

```
$ 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 也稱為靜態 CSC 演算法，並使用 CSC 第 2 版計算。服務可以提供 3、4 或 5 位數的號碼。

使用服務代碼 999 計算聯絡卡的 iCSC。使用服務代碼 702 計算非接觸式卡的 iCSC。

如需所有可用的參數，請參閱 API 參考指南中的 AmexCardSecurityCodeVersion2。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
        }
    }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-1：111122223333：key/7vrybrbvjcvwtunv。在下一個步驟中，您需要用到。

產生 iCSC

在此範例中，我們將為使用服務代碼 702 的感應式卡片產生長度為 4 的 iCSC。CSC 可以產生長度為 3、4 或 5。對於 American Express，PANs應為 15 位數，並以 34 或 37 開頭。

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"
}
```

驗證 iCSC

Example

在此範例中，我們將驗證 iCSC。

如果 AWS 付款密碼編譯能夠驗證，則會傳回 http/200。如果未驗證值，則會傳回 http/400 回應。

```
$ 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 (3D 安全帳戶驗證值） 用於 American Express 3-D安全身分驗證。它使用與 CSC2 相同的演算法，但具有不同的輸入參數。過期日期欄位應填入無法預測的 （隨機） 號碼，且服務代碼包含 AEVV 身分驗證結果代碼 (1 位數） 加上第二要素驗證代碼 (2 位數）。輸出長度應為 3 位數。

如需所有可用的參數，請參閱 API 參考指南中的 AmexCardSecurityCodeVersion2。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
        }
     }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/kw8djn5qxvfh3ztm。在下一個步驟中，您需要用到。

產生 3DS AEVV

在此範例中，我們將產生長度為 3 的 3DS AEVV。過期日期欄位包含無法預測的 （隨機） 號碼 （例如 1234)，而服務代碼包含 AEVV 身分驗證結果代碼 (1 位數） 加上第二要素驗證代碼 (2 位數），例如 543，其中 5 是身分驗證結果代碼，43 是第二要素驗證代碼。對於 American Express，PANs應為 15 位數，並以 34 或 37 開頭。

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"
}
```

驗證 3DS AEVV

Example

在此範例中，我們將驗證 3DS AEVV。

如果 AWS 付款密碼編譯能夠驗證，則會傳回 http/200。如果未驗證值，則會傳回 http/400 回應。

```
$ 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 特定函數

主題

- ARQC - CVN04
- ARQC - CVN01

ARQC - CVN04

JCB CVN04 使用金鑰衍生的 CSK 方法。如需建構交易資料欄位的詳細資訊，請參閱方案文件。

ARQC - CVN01

CVN01 是一種較舊的 EMV 交易 JCB 方法，使用每個卡片金鑰衍生而非工作階段（每個交易）衍生，也使用不同的承載。Visa 也會使用此訊息，因此元素名稱具有該名稱，即使它也用於 JCB。如需承載內容的相關資訊，請聯絡方案文件。

建立金鑰

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 以及金鑰檢查值 (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"
                    }
                }
```

請記下代表金鑰KeyArn的 ，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/pw3s6nl62t5ushfk。在下一個步驟中，您需要用到。

驗證 ARQC

Example

在此範例中，我們將驗證使用 JCB CVN01 產生的 ARQC。這使用與 Visa 方法相同的選項，因此參數的名稱。

如果 AWS 付款密碼編譯能夠驗證 ARQC，則會傳回 http/200。如果未驗證 arqc，則會傳回 http/400 回應。

```
$ 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 特定函數

下列範例與 UnionPay International (UPI) 相關，有時也稱為中國 UnionPay (CUP)。

主題

- ARQC 驗證
- CVN2（卡片驗證號碼）
- AVN（身分驗證號碼）

ARQC 驗證

UnionPay ARQC 驗證使用 EMV_OPTION_A 金鑰衍生模式。工作階段金鑰衍生自 PrimaryAccountNumber、PAN Sequence Number 和 Application Transaction Counter，使用 UnionPay 特定的方案。如需交易資料格式和欄位排序的相關資訊，請參閱方案文件。

驗證 ARQC

Example使用 UnionPay 金鑰衍生來驗證 ARQC

此範例說明如何驗證使用 UnionPay 金鑰衍生產生的 ARQC。

如果 AWS 付款密碼編譯驗證 ARQC，則會傳回 HTTP 200 回應。如果 ARQC 驗證失敗， AWS 付款密碼編譯會傳回具有原因代碼的 HTTP 400 回應。

```
$ 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"
}
```

如果無法驗證 ARQC，AWS 付款密碼編譯會傳回錯誤回應：

```
An error occurred (VerificationFailedException) when calling the
 VerifyAuthRequestCryptogram operation: Auth request cryptogram verification failed.

Reason: INVALID_AUTH_REQUEST_CRYPTOGRAM
```

CVN2（卡片驗證號碼）

CVN2 是相當於 CVV2 的 UnionPay。這是用於線上和card-not-present交易的卡片驗證號碼。以密碼編譯方式，CVN2 使用與 CVV2 (CardVerificationValue2) 相同的演算法。輸入是主要帳戶號碼 (PAN) 和卡片過期日期 (4 位數的 MMYY 格式）。

建立金鑰

若要建立 CVN2 金鑰，請使用 `create-key`命令搭配 `TR31_C0_CARD_VERIFICATION_KEY`用量和`TDES_2KEY`演算法。

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 和金鑰檢查值 (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"
    }
}
```

請記下 `KeyArn`值，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/7f7g4spf3xcklhzu。您需要在下列步驟中執行此操作。

產生 CVN2

Example為指定的 PAN 產生 CVN2

此範例會為輸入 PAN和卡片過期日期的指定 PAN 產生 CVN2。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue2。

```
$ 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"
}
```

驗證 CVN2

Example驗證指定 PAN 的 CVN2

此範例會驗證指定 PAN 的 CVN2PAN，其輸入為 、卡片過期日期，以及交易期間提供的 CVN2。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue2。

> **Note**
>
> CVN2 是使用者輸入的值。定期驗證失敗不一定表示發生問題。

```
$ 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 （身分驗證號碼）

AVN （身分驗證號碼） 是 UnionPay 3DS 密碼編譯。它使用 CVV 演算法 (CardVerificationValue1) 搭配如下的重用輸入欄位，但請查看結構描述文件以取得最終詳細資訊。

- PAN — 主要帳戶號碼 (16-19 位數，來自 AReq)
- CardExpiryDate — 從 DS 交易 ID (dsTransID) 衍生的值的四位數最低有效數字

- ServiceCode — 3 位數：身分驗證交易結果 (1 位數） + 填入位元 '0' (1 位數） + AV 產生器指標 (1 位數）

建立金鑰

若要建立 AVN 金鑰，請使用 `create-key`命令搭配 `TR31_C0_CARD_VERIFICATION_KEY`用量和`TDES_2KEY`演算法。

```
$ 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"}]'
```

回應會回傳請求參數，包括後續呼叫的 ARN 和金鑰檢查值 (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"
    }
}
```

請記下 KeyArn值，例如 arn：aws：payment-cryptography：us-east-2：111122223333：key/dnaeyrjgdjjtw6dk。您需要在下列步驟中執行此操作。

產生 AVN

Example為指定的 PAN 產生 AVN

此範例會為指定的 PAN 產生 AVN。CardExpiryDate 欄位包含從 dsTransID 衍生的四個最低有效數字。ServiceCode 欄位包含身分驗證結果、填入位元和產生器指標。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ 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"
}
```

驗證 AVN

Example驗證指定 PAN 的 AVN

此範例會使用產生的相同輸入欄位，以及要驗證的 AVN 值，來驗證指定 PAN 的 AVN。

如需所有可用的參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ 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"
}
```

取得和付款引導程式

收單機構、PSPs和 Payment Facilators 通常具有與發行者不同的一組密碼編譯要求。常用案例包括：

資料解密

資料（特別是平移資料）可能由付款終端機加密，且需要由後端解密。Decrypt Data 和 Encrypt Data 支援各種方法，包括 TDES、AES 和 DUKPT 衍生技術。 AWS 付款密碼編譯服務本身也符合 PCI P2PE 規範，並已註冊為 PCI P2PE 解密元件。

TranslatePin

為了維持 PCI PIN 合規，擷取系統在安全裝置上輸入後，不應讓持卡人接腳保持清晰。因此，若要將接腳從終端機傳遞到下游系統（例如付款網路或發行者），需要使用與付款終端機使用的金鑰不同的金鑰來重新加密。使用 servicebbb 安全地將加密的 PIN 從一個金鑰轉換為另一個金鑰，即可完成轉換 Pin。使用此命令，您可以在 TDES、AES 和 DUKPT 衍生等各種方案之間轉換接腳，以及 ISO-0, ISO-3 和 ISO-4 等接腳區塊格式。

VerifyMac

來自付款終端機的資料可能是 MAC，以確保資料未在傳輸中修改。驗證 Mac 和 GenerateMac 是否支援各種技術，包括 TDES、AES 和 DUKPT 衍生技術，可與 ISO-9797-1 演算法 1、ISO-9797-1 演算法 3（零售 MAC) 和 CMAC 技術搭配使用。

其他主題

- 使用動態金鑰

使用動態金鑰

動態金鑰允許一次性或有限使用金鑰用於密碼編譯操作，例如 EncryptData。當金鑰材料頻繁輪換（例如在每個卡片交易上），並且想要避免將金鑰材料匯入服務時，可以使用此流程。短期金鑰可用作 softPOS/Mpoc 或其他解決方案的一部分。

> **Note**
>
> 這可用於代替使用 AWS 付款密碼編譯的典型流程，其中密碼編譯金鑰是建立或匯入服務，而金鑰是使用金鑰別名或金鑰 arn 指定。

下列操作支援動態金鑰：

- EncryptData
- DecryptData
- ReEncryptData
- TranslatePin

解密資料

下列範例顯示搭配解密命令使用動態金鑰。在此情況下，金鑰識別符是保護解密金鑰的包裝金鑰 (KEK)（以 TR-31 格式在包裝金鑰參數中提供）。包裝金鑰應為 D0 的關鍵用途，以搭配解密命令以及 B 或 D 的使用模式使用。

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"
}
```

翻譯 PIN 碼

下列範例顯示使用動態金鑰和轉譯接腳命令，將動態金鑰轉譯為半靜態取得器工作金鑰 (AWK)。在這種情況下，傳入金鑰識別符是包裝金鑰 (KEK)，用於保護 TR-31 格式提供的動態 PIN 加密金鑰 (PEK)。包裝金鑰應該是 的金鑰用途，P0以及 B 或 D 的使用模式。傳出金鑰識別符是 類型的金鑰TR31_P0_PIN_ENCRYPTION_KEY和 Encrypt=true、Wrap=true 的使用模式

Example

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

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

AWS 付款密碼編譯的區域特定功能

某些功能可能是區域特定的，不會以其他方式使用。這些功能會在本節中詳細說明。

AS2805

Australia Standard 2805 (AS2805) 是主要用於卡片式付款交易之電子轉帳的標準。它由 Standards Australia 維護。標準包含 6 本書籍，涵蓋從訊息格式到加密標準的許多主題。

第 6 部分提供金鑰管理的指引host-to-host(node-to-node通訊和相關密碼編譯要求，而其他部分則涵蓋其他層面。此標準中的所有密碼編譯目前都以 TDES 為基礎。

AS2805 與其他實作相比有許多差異，摘要如下。

金鑰保護

依賴金鑰變體而不是 TR-31/X9.143. AWS Payment Cryptography 中的 等金鑰區塊，將所有金鑰存放為內部的金鑰區塊，但允許使用 AS2805 定義的變體進行匯入、匯出和計算。

單向金鑰

AS2805 強制使用單向金鑰。如果兩個節點都需要產生訊息驗證碼 (MAC)，它們會使用兩個金鑰。

Pin 區塊

AS2805 為每個交易定義唯一 PIN 加密金鑰的金鑰衍生技術。這可以用來取代 DUKPT。相較於 DUKPT 使用交易計數器，AS2805 方案倚賴交易資料（追蹤數目和交易金額）。

金鑰交換驗證

定義在開始交換工作金鑰之前驗證 KEK 的程序，例如 PIN 金鑰。在其他方案中，KEK 不常交換，並使用 KCV 驗證。

AS2805 使用金鑰變體的概念，而不是金鑰區塊，以確保金鑰僅用於預期（和唯一）用途。以下是使用金鑰匯入、匯出或執行其他密碼編譯函數時， AWS 付款密碼編譯如何在變體和金鑰區塊之間映射。

AS2805 金鑰類型	AWS 付款密碼編譯金鑰類型
TERMINAL_MAJOR_KEY_VARIANT_00	TR31_K0_KEY_ENCRYPTION_KEY
PIN_ENCRYPTION_KEY_VARIANT_28	TR31_P0_PIN_ENCRYPTION_KEY

AS2805 金鑰類型	AWS 付款密碼編譯金鑰類型
MESSAGE_AUTHENTICATION_KEY_VARIANT_24	TR31_M0_ISO_16609_MAC_KEY
DATA_ENCRYPTION_KEY_VARIANT_22	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY
VARIANT_MASK_82、VARIANT_MASK_82C0	KEK 驗證程序中可用的選項。這些金鑰類型是暫時性的，不會由 服務存放。

假設有兩個節點，節點 1 和節點 2，下列範例是從節點 1 的角度來看。 AWS 付款密碼編譯支援來自程序兩端的 APIs。

主題

- 初始金鑰 (KEK) 交換
- KEK 驗證
- 建立和傳輸工作金鑰
- 匯出工作金鑰
- 接腳轉譯
- Mac 產生和驗證

初始金鑰 (KEK) 交換

在 AS2805 中，每一端都有自己的 KEK。KEK(s) 是指每當傳送端需要保護/包裝金鑰並將其傳送至 node2 時，將使用的傳送端金鑰。KEK(r) 是由 opposite(node2) 端建立的金鑰。

> **Note**
>
> 這些術語是相對的 - 一端建立金鑰 （傳送端），另一端接收到金鑰。因此，指定 KEY1，其在節點 1 上稱為 KEK(s)，在節點 2 上稱為 KEK(r)。

AS2805 的 KEK 始終是金鑰類型 = TR31_K0_KEY_ENCRYPTION_KEY，因為它們用於保護密碼編譯，而不是金鑰區塊。這對應至 AS2805 6.1 中定義的 TERMINAL_MAJOR_KEY_VARIANT_00

步驟：

1.建立金鑰

使用 CreateKey api 建立金鑰。您將建立 TR31_K0_KEY_ENCRYPTION_KEY 類型的金鑰

2.決定與 node2 交換金鑰的方法

決定如何與對手方交換 KEK。對於 AS2805，最常見且可互通的方法是 RSA Wrap。

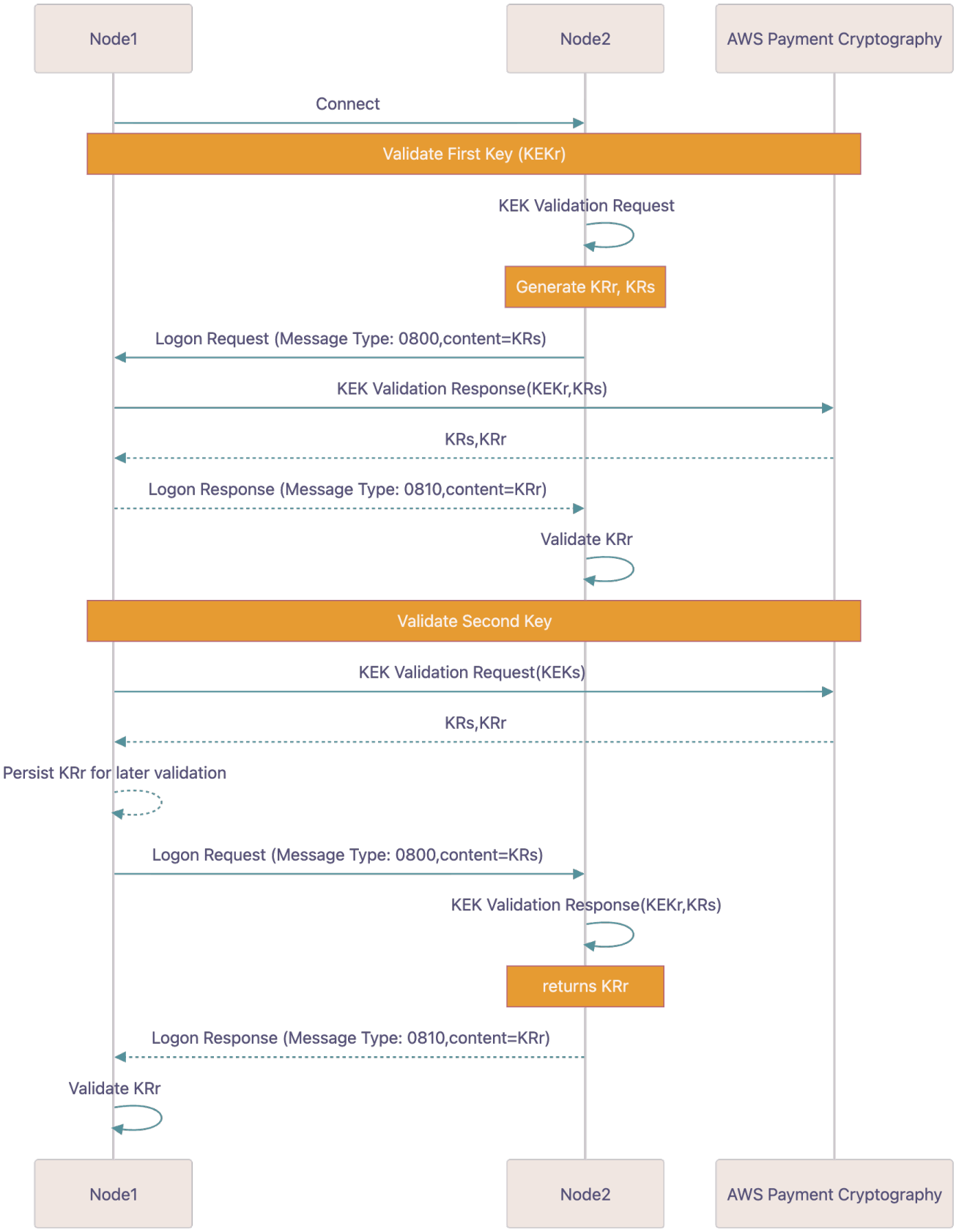
3.匯出 KEKs

根據上述選擇，您將會從 node2 收到公有金鑰憑證。您將使用該憑證執行匯出以保護金鑰 （ 如果使用 ECDH，則衍生金鑰 ）。

4. 匯入 KEKr

根據上述選擇，您會將公有金鑰憑證傳送至 node2。您將使用該憑證執行匯入至 ，將節點 2 的 KEKr 載入服務。

KEK 驗證


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

當您的服務 (node1) 連線到 node2 時，每一端都會確保他們使用相同的 KEK 進行後續操作，並使用稱為 KEK 驗證的程序。

1。驗證第一個金鑰的步驟

1.1 接收 KRs

Node2 KRs，並將其作為登入程序的一部分傳送給您。他們可以使用 AWS 付款密碼編譯來產生此值或其他解決方案。

1.2 產生 KEK 驗證回應

您的節點將產生 KEK 驗證回應，並將輸入做為步驟 1 中提供的 KEK(r) 和 KRs。

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 傳回計算的 KRr

將計算的 KRr 傳回至 node2。該節點會將其與步驟 1 的計算值進行比較。

2.驗證第二個金鑰的步驟

2.1 產生 KRr KRs

您的節點將使用 AWS 付款密碼編譯產生隨機值和此值的反轉 （反轉） 副本。服務會輸出這兩個由 KEK 包裝的值 (KEK)。這些稱為 KR(s) 和 KR(r)。

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 將 KRs傳送至 node2

將 KRs傳送至 node2. 保留 KRr 以供稍後驗證。

2.3 Node2 產生 KEK 驗證回應

Node2 使用 KEKr KRs、產生 KRr 並將其傳回給您的服務。

2.4 驗證回應

比較步驟 1 的 KRr 和步驟 3 傳回的值。如果相符，請繼續。

建立和傳輸工作金鑰

AS2805 中使用的一般工作金鑰包含兩組金鑰：

節點之間的金鑰，例如：區域 PIN 金鑰 (ZPK)、區域加密金鑰 (ZEK) 和區域身分驗證金鑰 (ZAK)。

如果未使用 DUKPT，則終端機和節點之間的金鑰，例如：終端機主金鑰 (TMK) 和終端機接腳金鑰 (TPK)。

> **Note**
>
> 我們建議您盡可能將每個終端機金鑰的金鑰和利用 TR-34 和 DUKPT 等技術，盡可能使用較少數量的金鑰。

Example

在此範例中，我們使用選用標籤來追蹤此金鑰的用途和使用方式。標籤不會用作系統密碼編譯函數的一部分，但可用於分類、財務追蹤，並可用於套用 IAM 政策。

```
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"
 }
}
```

匯出工作金鑰

為了維持與其他方的相容性， AWS 付款密碼編譯支援 AS2805 對稱金鑰包裝技術，這些技術使用金鑰變體，而不是 TR-31 等金鑰區塊。如果各方之間共用多個金鑰，則每個金鑰都應個別匯出。如果資料是雙向傳送，則相同類型的各方之間可能會有兩個金鑰，例如 ZAK(s) 和 ZAK(r)，由每一端用來產生訊息驗證碼。

要以這些格式匯入和匯出的其他參數會在命令上指定。

```
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"
    }
}
```

接腳轉譯

AS2805 說明第 6.4 節中的工作階段特定金鑰衍生模式。它做為 DUKPT 提供相同的用途，而且任一演算法都可以做為 DUKPT 使用，如第 6.7 節所述。在此機制中，工作階段 PIN 金鑰 （ 稱為 KPE) 衍生自使用 SystemTraceAuditNumber(STAN) 和 TransactionAmount 作為衍生資料的終端機 Pin 金鑰。

翻譯接腳是一種常見函數，可以翻譯成各種格式或從中翻譯。在此範例中，我們會將接腳從 KPE 轉換為接腳加密金鑰 (PEK)，例如將接腳傳送到付款網路時。

```
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 產生和驗證

產生和驗證 MAC 命令支援各種 MACs，包括 HMAC、CMAC、EMV MAC 等。對於 AS2805，AS2805.4.1 中定義了額外的變化。在 AS2805 中，通常會使用此 MAC 驗證傳入訊息，而傳出訊息也包含 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"
}
```

AWS 付款密碼編譯的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間的共同責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 -AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。 AWS 還提供您可以安全使用的服務。在AWS 合規計畫中，第三方稽核人員會定期測試和驗證我們的安全有效性。若要了解適用於 AWS 付款密碼編譯的合規計劃，請參閱合規計劃的 AWS 服務範圍。
- 雲端的安全性 - 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本主題可協助您了解如何在使用 AWS 付款密碼編譯時套用共同責任模型。其中說明如何設定 AWS 付款密碼編譯以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 AWS 付款密碼編譯資源。

主題

- AWS 付款密碼編譯中的資料保護
- AWS 付款密碼編譯中的彈性
- 中的基礎設施安全 AWS Payment Cryptography
- 透過 VPC 端點連線至 AWS 付款密碼編譯
- 使用混合式後量子 TLS
- AWS 付款密碼編譯的安全最佳實務

AWS 付款密碼編譯中的資料保護

AWS 共同責任模型適用於 AWS 付款密碼編譯中的資料保護。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。 如需歐洲資料保護的詳細資訊，請參閱一般資料保護規則 (GDPR) 中心。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 AWS 付款密碼編譯或使用主控台 AWS CLI、API 或 AWS SDKs 的其他 AWS 服務 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

AWS Payment Cryptography 會存放並保護您的付款加密金鑰，使其高度可用，同時為您提供強大且靈活的存取控制。

主題

- 保護金鑰資料
- 資料加密
- 靜態加密
- 傳輸中加密
- 網際網路流量隱私權

保護金鑰資料

根據預設，AWS Payment Cryptography 會保護由 服務管理之付款金鑰的密碼編譯金鑰資料。此外，AWS Payment Cryptography 提供匯入在服務外部建立之金鑰資料的選項。如需付款金鑰和金鑰材料的技術詳細資訊，請參閱 AWS 付款密碼編譯詳細資訊。

資料加密

AWS Payment Cryptography 中的資料包含 AWS Payment Cryptography 金鑰、其代表的加密金鑰材料，以及其用量屬性。金鑰材料僅以純文字形式存在於 AWS Payment Cryptography 硬體安全模組 (HSMs) 內，且僅在使用中時存在。否則，金鑰材料和屬性會加密並存放在持久性持久性儲存中。

AWS Payment Cryptography 為付款金鑰產生或載入的金鑰材料絕不會讓 AWS Payment Cryptography HSMs 的界限處於未加密狀態。它可以透過 AWS Payment Cryptography API 操作來匯出加密。

靜態加密

AWS Payment Cryptography 會為 PCI PTS HSM 列出的 HSMs 中的付款金鑰產生金鑰材料。不使用時，金鑰資料會由 HSM 金鑰加密，並寫入耐久的持久性儲存裝置。付款密碼編譯金鑰的金鑰材料和保護金鑰材料的加密金鑰絕不會以純文字形式保留 HSMs。

付款密碼編譯金鑰的金鑰材料加密和管理完全由 服務處理。

如需詳細資訊，請參閱 AWS Key Management Service 密碼編譯詳細資訊。

傳輸中加密

AWS 付款密碼編譯為付款金鑰產生或載入的金鑰材料絕不會在 AWS 付款密碼編譯 API 操作中以純文字匯出或傳輸。 AWS 付款密碼編譯會使用金鑰識別符來代表 API 操作中的金鑰。

不過，有些 API 操作會匯出先前共用或非對稱金鑰交換金鑰所加密的金鑰。此外，客戶可以使用 API 操作來匯入付款金鑰的加密金鑰材料。

所有 AWS 付款密碼編譯 API 呼叫都必須使用 Transport Layer Security (TLS) 簽署和傳輸。 AWS 付款密碼編譯需要 PCI 定義為「嚴格密碼編譯」的 TLS 版本和密碼套件。所有服務端點都支援 TLS 1.2—1.3 和混合式後量子 TLS。

如需詳細資訊，請參閱 AWS Key Management Service 密碼編譯詳細資訊。

網際網路流量隱私權

AWS 付款密碼編譯支援 AWS 管理主控台和一組 API 操作，可讓您建立和管理付款金鑰，並在密碼編譯操作中使用它們。

AWS 付款密碼編譯支援從您的私有網路到 AWS 的兩個網路連線選項。

- 透過網際網路的 IPSec VPN 連線。
- AWS Direct Connect，透過標準乙太網路光纖纜線將您的內部網路連結至 AWS Direct Connect 位置。

所有付款密碼編譯 API 呼叫都必須使用 Transport Layer Security (TLS) 簽署和傳輸。這些呼叫還需要支援完整轉寄密碼的現代加密套件。只能透過 AWS 內部網路從已知的 AWS Payment Cryptography API 主機傳輸到存放付款金鑰之金鑰資料的硬體安全模組 (HSMs)。

若要從虛擬私有雲端 (VPC) 直接連線至 AWS Payment Cryptography，而不透過公有網際網路傳送流量，請使用採用 AWS PrivateLink 技術的 VPC 端點。如需詳細資訊，請參閱透過 VPC 端點連線至 AWS Payment Cryptography。

AWS Payment Cryptography 也支援 Transport Layer Security (TLS) 網路加密通訊協定的混合式後量子金鑰交換選項。當您連線到 AWS Payment Cryptography API 端點時，您可以將此選項與 TLS 搭配使用。

AWS 付款密碼編譯中的彈性

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置。區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援網路連線相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域和可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

區域隔離

AWS Payment Cryptography 是一項區域性服務，可在多個區域使用。

AWS Payment Cryptography 的區域隔離設計可確保一個 AWS 區域中的可用性問題不會影響任何其他區域中的 AWS Payment Cryptography 操作。AWS Payment Cryptography 旨在確保零計劃停機時間，所有軟體更新和擴展操作都能無縫且無法察覺地執行。

AWS Payment Cryptography Service Level Agreement (SLA) 包含所有 Payment Cryptography APIs 99.99% 的服務承諾。為了履行此承諾，AWS Payment Cryptography 可確保執行 API 請求所需的所有資料和授權資訊，都可用於接收請求的所有區域主機。

AWS Payment Cryptography 基礎設施會在每個區域中至少三個可用區域 (AZs中複寫。為了確保多個主機故障不會影響 AWS Payment Cryptography 效能，AWS Payment Cryptography 旨在服務來自區域中任何 AZs 的客戶流量。

您對付款金鑰屬性或許可所做的變更會複寫到 區域中的所有主機，以確保該區域中的任何主機都可以正確處理後續請求。使用您的付款金鑰對密碼編譯操作的請求會轉送至 AWS Payment Cryptography 硬體安全模組 (HSMs) 機群，其中任何一個都可以使用付款金鑰執行操作。

多租用戶設計

AWS Payment Cryptography 的多租戶設計可讓它滿足可用性 SLA，並維持高請求率，同時保護金鑰和資料的機密性。

部署多個完整性強制執行機制，以確保您為密碼編譯操作指定的付款金鑰始終是使用的付款金鑰。

付款密碼編譯金鑰的純文字金鑰材料受到廣泛保護。金鑰材料會在建立後立即在 HSM 中加密，而且加密的金鑰材料會立即移至安全儲存。系統會在 HSM 內擷取並解密已加密的金鑰，以便及時使用。純文字金鑰僅在完成密碼編譯操作所需的時間內保留在 HSM 記憶體中。純文字金鑰資料永遠不會離開 HSM；它永遠不會寫入持久性儲存。

如需 AWS Payment Cryptography 用來保護金鑰之機制的詳細資訊，請參閱 AWS Payment Cryptography 密碼編譯詳細資訊。

中的基礎設施安全 AWS Payment Cryptography

作為受管服務， AWS Payment Cryptography 受到 Amazon Web Services：安全程序概觀白皮書中所述的 AWS 全球網路安全程序的保護。

您可以使用 AWS 發佈的 API 呼叫， AWS Payment Cryptography 透過網路存取 。用戶端必須支援 Transport Layer Security (TLS) 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

實體主機的隔離

AWS Payment Cryptography 使用的實體基礎設施安全性受 Amazon Web Services：安全程序概觀之實體和環境安全一節中所述的控制。您可以在上一節所列的合規報告和第三方稽核問題清單中找到更多詳細資訊。

專用commercial-off-the-shelf PCI PTS HSM 列出的硬體安全模組 (HSMs) 支援 AWS Payment Cryptography。AWS Payment Cryptography 金鑰的金鑰材料只會存放在 HSMs 的揮發性記憶體中，而且只有在使用 Payment Cryptography 金鑰時。HSMs位於 Amazon 資料中心內的受存取控制機架中，可強制對任何實體存取進行雙重控制。如需 AWS Payment Cryptography HSMs 操作的詳細資訊，請參閱 AWS Payment Cryptography Cryptography Details。

透過 VPC 端點連線至 AWS 付款密碼編譯

您可以透過虛擬私有雲端 (VPC) 中的私有介面端點直接連線至 AWS 付款密碼編譯。當您使用界面 VPC 端點時，VPC 與 AWS 付款密碼編譯之間的通訊會完全在 AWS 網路內進行。

AWS 付款密碼編譯支援採用 技術的 Amazon Virtual Private Cloud (Amazon VPC) 端點AWS PrivateLink。每個 VPC 端點皆會由一個或多個具私有 IP 地址彈性網路界面 (ENI) 來表示，而該界面位於 VPC 子網路中。

介面 VPC 端點會將您的 VPC 直接連線至 AWS 付款密碼編譯，無需網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線。VPC 中的執行個體不需要公有 IP 地址，即可與 AWS 付款密碼編譯通訊。

大區 (Regions)

AWS 支援付款密碼的所有 中 AWS 區域 都支援 VPC 端點和 VPC AWS端點政策。

主題

- AWS 付款密碼編譯 VPC 端點的考量事項
- 建立用於 AWS 付款密碼編譯的 VPC 端點
- 連線至 AWS 付款密碼編譯 VPC 端點
- 控制對 VPC 端點的存取
- 在政策陳述式中使用 VPC 端點
- 記錄您的 VPC 端點

AWS 付款密碼編譯 VPC 端點的考量事項

> **Note**
>
> 雖然 VPC 端點可讓您在最少一個可用區域 (AZ) 中連線至服務，但我們建議您連線至三個可用區域，以實現高可用性和備援目的。

在您設定 AWS 付款密碼編譯的介面 VPC 端點之前，請檢閱 AWS PrivateLink 指南中的介面端點屬性和限制主題。

AWS VPC 端點的付款密碼編譯支援包括下列項目。

- 您可以使用 VPC 端點從 VPC 呼叫所有AWS 付款密碼編譯控制平面操作和AWS 付款密碼編譯資料平面操作。
- 您可以建立連線至 AWS 付款密碼編譯區域端點的介面 VPC 端點。
- AWS 付款密碼編譯由控制平面和資料平面組成。您可以選擇設定一個或兩個子服務， AWS PrivateLink 但每個子服務都會分別設定。
- 您可以使用 AWS CloudTrail 日誌，透過 VPC 端點稽核您對 AWS 付款密碼編譯金鑰的使用。如需詳細資訊，請參閱記錄您的 VPC 端點。

建立用於 AWS 付款密碼編譯的 VPC 端點

您可以使用 Amazon VPC 主控台或 Amazon VPC API，為 AWS 付款密碼編譯建立 VPC 端點。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的「建立介面端點」。

- 若要為 AWS 付款密碼編譯建立 VPC 端點，請使用下列服務名稱：

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

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

例如，在美國西部（奧勒岡）區域 (us-west-2) 中，服務名稱為：

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

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

若要更輕鬆使用 VPC 端點，您可以為 VPC 端點啟用私有 DNS 名稱。如果您選取啟用 DNS 名稱選項，標準 AWS 付款密碼編譯 DNS 主機名稱會解析為您的 VPC 端點。例如，`https://controlplane.payment-cryptography.us-west-2.amazonaws.com` 會解析為連接至服務名稱 `com.amazonaws.us-west-2.payment-cryptography.controlplane` 的 VPC 端點。

此選項可讓您更輕鬆使用 VPC 端點。根據預設， AWS SDKs 和 AWS CLI 會使用標準 AWS 付款密碼編譯 DNS 主機名稱，因此您不需要在應用程式和命令中指定 VPC 端點 URL。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的透過介面端點存取服務。

連線至 AWS 付款密碼編譯 VPC 端點

您可以使用 AWS SDK、 AWS CLI 或 ，透過 VPC 端點連線至 AWS 付款密碼編譯 AWS Tools for PowerShell。若要指定 VPC 端點，請使用它的 DNS 名稱。

例如，此 list-keys 命令會使用 `endpoint-url` 參數來指定 VPC 端點。若要使用如下的命令，請將範例 VPC 端點 ID 換成您帳戶中的 ID。

```
$ aws payment-cryptography list-keys --endpoint-url https://
vpce-1234abcdf5678c90a-09p7654s-us-east-1a.ec2.us-east-1.vpce.amazonaws.com
```

如果您在建立 VPC 端點時啟用私有主機名稱，則不需要在 CLI 命令或應用程式組態中指定 VPC 端點 URL。標準 AWS 付款密碼編譯 DNS 主機名稱會解析為您的 VPC 端點。 AWS CLI 和 SDKs 預設會使用此主機名稱，因此您可以開始使用 VPC 端點連線到 AWS 付款密碼編譯區域端點，而無需變更指令碼和應用程式中的任何內容。

若要使用私有主機名稱，您 VPC 的 `enableDnsHostnames` 和 `enableDnsSupport` 屬性必須設為 `true`。如需設定這些屬性，請使用 ModifyVpcAttribute 操作。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的檢視和更新 VPC 的 DNS 屬性。

控制對 VPC 端點的存取

若要控制 AWS 對付款密碼編譯之 VPC 端點的存取，請將 VPC 端點政策連接至您的 VPC 端點。端點政策會決定主體是否可以使用 VPC 端點來呼叫具有特定 AWS 付款密碼編譯資源的 AWS 付款密碼編譯操作。

您可以在建立端點時建立 VPC 端點政策，並且可以隨時變更 VPC 端點政策。使用 VPC 管理主控台，或 CreateVpcEndpoint 或 ModifyVpcEndpoint 操作。您也可以使用 AWS CloudFormation 範本建立和變更 VPC 端點政策。如需有關如何使用 VPC 管理主控台的說明，請參閱《AWS PrivateLink 指南》中的建立介面端點和修改介面端點。

如需撰寫及格式化 JSON 政策文件的說明，請參閱《IAM 使用者指南》中的 IAM JSON 政策參考。

主題

- 關於 VPC 端點政策
- 預設 VPC 端點政策
- 建立 VPC 端點政策
- 檢視 VPC 端點政策

關於 VPC 端點政策

若要讓使用 VPC 端點的 AWS 付款密碼編譯請求成功，委託人需要兩個來源的許可：

- 身分型政策必須授予委託人在 資源 (AWS 付款加密金鑰或別名） 上呼叫 操作的許可。
- VPC 端點政策必須授予委託人許可，才能使用端點提出請求。

例如，金鑰政策可能會授予委託人在特定 AWS 付款密碼編譯金鑰上呼叫 Decrypt 的許可。不過，VPC 端點政策可能不允許該主體使用端點呼叫Decrypt該 AWS 付款密碼編譯金鑰。

或者，VPC 端點政策可能允許委託人使用端點，在某些 AWS 付款密碼編譯金鑰上呼叫 StopKeyUsage。但是，如果委託人沒有來自 IAM 政策的這些許可，請求會失敗。

預設 VPC 端點政策

每個 VPC 端點都有 VPC 端點政策，但您不需要指定政策。如果您未指定政策，則預設端點政策會允許端點上所有資源的所有委託人進行所有操作。

不過，對於 AWS 付款密碼編譯資源，委託人也必須具有從 IAM 政策呼叫 操作的許可。因此，實際上，預設政策指出，如果委託人具有對資源呼叫操作的許可，則其也可以使用端點來進行呼叫。

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

若要僅允許主體將 VPC 端點用於其允許操作的子集，請建立或更新 VPC 端點政策。

建立 VPC 端點政策

VPC 端點政策決定委託人是否具有使用 VPC 端點對資源執行操作的許可。對於 AWS 付款密碼編譯資源，委託人也必須具有從 IAM 政策執行操作的許可。

每個 VPC 端點政策陳述式都需要下列元素：

- 可執行動作的委託人
- 可執行的動作
- 可在其中執行動作的資源

政策陳述式不會指定 VPC 端點。相反地，它適用於連接政策的任何 VPC 端點。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用 VPC 端點控制對服務的存取。

以下是 AWS 付款密碼編譯的 VPC 端點政策範例。連接到 VPC 端點時，此政策允許 ExampleUser使用 VPC 端點來呼叫指定 AWS 付款密碼編譯金鑰上的指定操作。在使用像這樣的政策之前，請將範例主體和金鑰識別符取代為來自您帳戶的有效值。

```
{
   "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 會記錄使用 VPC 端點的所有操作。不過，您的 CloudTrail 日誌不包含其他帳戶中主體請求的操作，或其他帳戶中 AWS 付款密碼編譯金鑰的操作。

因此，您可能想要建立 VPC 端點政策，以防止外部帳戶中的主體使用 VPC 端點來呼叫本機帳戶中任何金鑰的任何 AWS 付款密碼編譯操作。

下列範例使用 aws：PrincipalAccount 全域條件金鑰，拒絕存取所有 AWS 付款密碼編譯金鑰上所有操作的所有主體，除非主體位於本機帳戶中。使用這類政策之前，請將範例帳戶 ID 取代為有效值。

```
{
  "Statement": [
```

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

檢視 VPC 端點政策

若要檢視端點的 VPC 端點政策，請使用 VPC 管理主控台或 DescribeVpcEndpoints 操作。

下列 AWS CLI 命令會取得具有指定 VPC 端點 ID 的端點政策。

使用此命令之前，請將範例端點 ID 取代為您帳戶的有效 ID。

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==`vpce-1234abcdf5678c90a`].[PolicyDocument]'
--output text
```

在政策陳述式中使用 VPC 端點

當請求來自 VPC 或使用 VPC 端點時，您可以控制對 AWS 付款密碼編譯資源和操作的存取。若要這樣做，請使用其中一個 IAM 政策

- 使用 aws:sourceVpce 條件索引鍵，以根據 VPC 端點來授予或限制存取。
- 使用 aws:sourceVpc 條件索引鍵，以根據託管私有端點的 VPC 來授予或限制存取。

> **Note**
>
> 當請求來自 Amazon VPC 端點時，aws:sourceIP條件索引鍵無效。若要限制對 VPC 端點的請求，請使用 aws:sourceVpce 或 aws:sourceVpc 條件金鑰。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的VPC 端點和 VPC 端點服務的身分與存取管理

您可以使用這些全域條件金鑰來控制對 AWS Payment Cryptography 金鑰、別名和 CreateKey 等操作的存取，這些操作不依賴於任何特定資源。

例如，以下範例金鑰政策允許使用者僅在請求使用指定的 VPC 端點時，使用 AWS 付款密碼編譯金鑰執行特定的密碼編譯操作，封鎖來自網際網路和 AWS PrivateLink 連線的存取（如果設定）。當使用者向 AWS 付款密碼編譯提出請求時，請求中的 VPC 端點 ID 會與政策中的`aws:sourceVpce`條件索引鍵值進行比較。如果不相符，則會拒絕請求。

若要使用這類政策，請將預留位置 AWS 帳戶 ID 和 VPC 端點 IDs 取代為帳戶的有效值。

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"
```

```
                }
            }
        }

    ]
}
```

您也可以使用 aws:sourceVpc條件金鑰，根據 VPC 端點所在的 VPC 限制對 AWS 付款密碼編譯金鑰的存取。

下列範例金鑰政策允許僅在 金鑰來自 時才管理 AWS 付款密碼編譯金鑰的命令vpc-12345678。此外，它允許僅在密碼編譯操作使用 AWS 付款密碼編譯金鑰的命令來自 vpc-2b2b2b2b。如果應用程式在一個 VPC 中執行，但您使用第二個隔離的 VPC 來執行管理功能，您可能會使用如下的政策。

若要使用這類政策，請將預留位置 AWS 帳戶 ID 和 VPC 端點 IDs 取代為帳戶的有效值。

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": "*"
        }
    ]
}
```

記錄您的 VPC 端點

AWS CloudTrail 會記錄使用 VPC 端點的所有操作。當對 AWS 付款密碼編譯的請求使用 VPC 端點時，VPC 端點 ID 會出現在記錄請求的AWS CloudTrail 日誌項目中。您可以使用端點 ID 稽核 AWS 付款密碼編譯 VPC 端點的使用。

為了保護您的 VPC，VPC 端點政策拒絕的請求，但本應允許，則不會記錄在 中AWS CloudTrail。

例如，此範例日誌項目會記錄使用 VPC 端點的 GenerateMac 請求。vpcEndpointId 欄位出現在日誌項目結尾。


```
{
      "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,
```

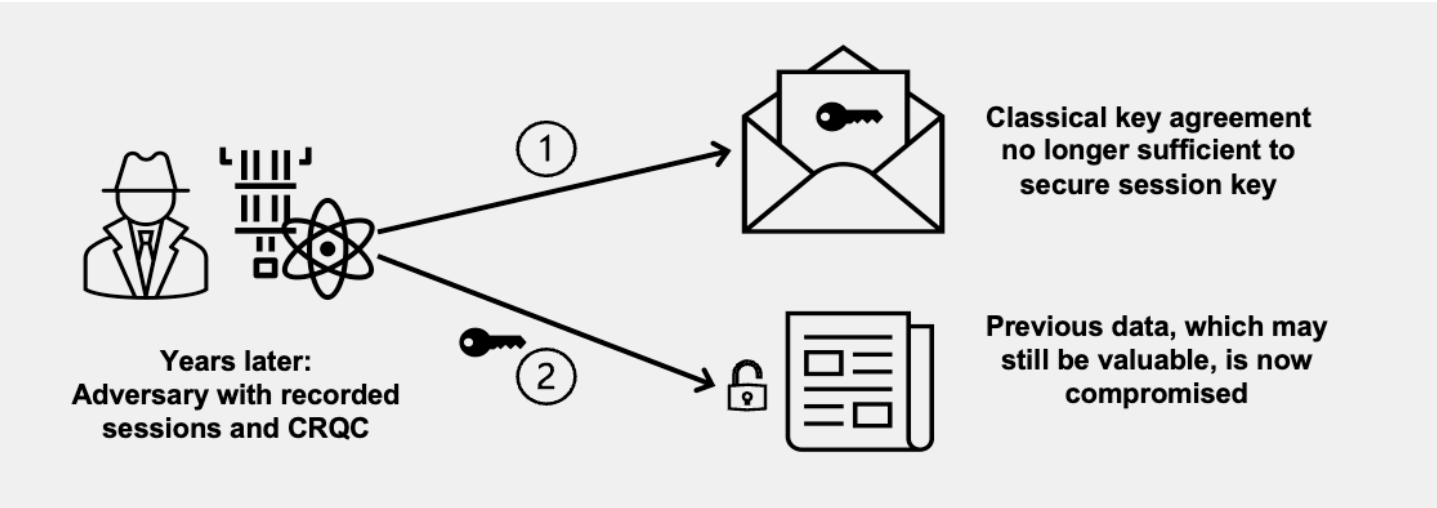
```
                    "unwrap": false,
                    "generate": true,
                    "sign": false,
                    "verify": true,
                    "deriveKey": false,
                    "noRestrictions": false
                }
            },
            "exportable": true
        },
        "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"
      }
  }
```

使用混合式後量子 TLS

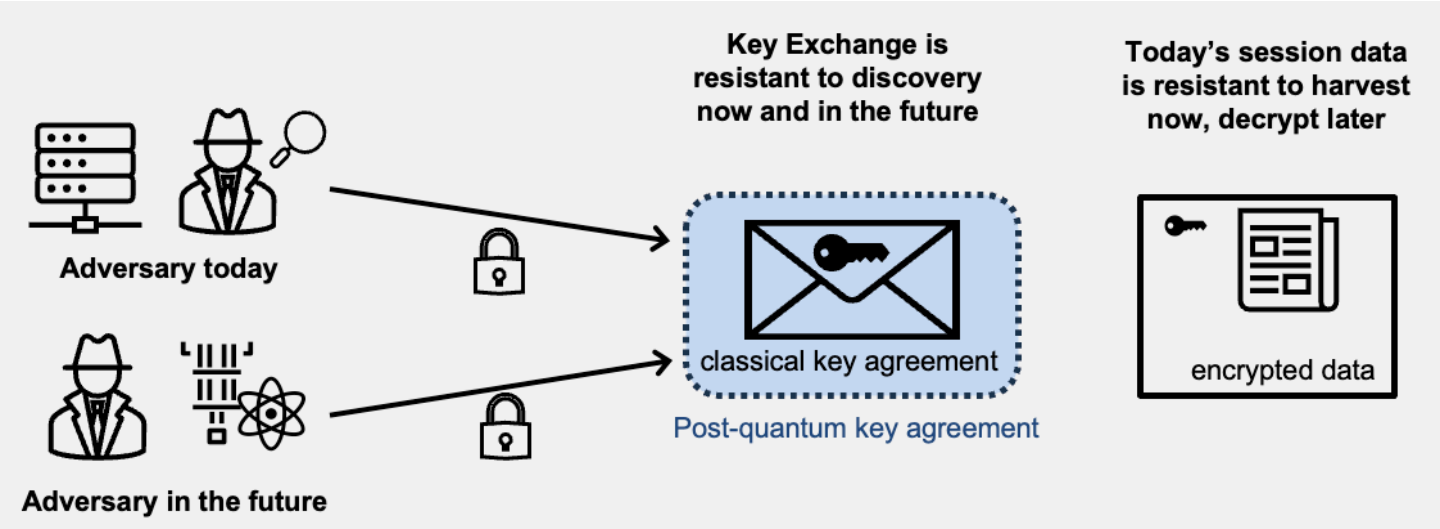
AWS 付款密碼編譯和許多其他服務支援 Transport Layer Security (TLS) 網路加密通訊協定的混合式後量子金鑰交換選項。您可以在連線至 API 端點或使用 AWS SDKs時使用此 TLS 選項。這些選用的混合式後量子金鑰交換功能至少與現今使用的 TLS 加密功能同樣安全，且還能提供其他長期安全優勢。

您傳送至已啟用服務的資料受到 Transport Layer Security (TLS) 連線提供的加密保護。 AWS 付款密碼編譯支援以 RSA 和 ECC 為基礎的傳統密碼套件，適用於 TLS 工作階段，對金鑰交換機制造成暴力攻擊，與目前技術不相容。不過，如果大規模或密碼編譯相關的量子電腦 (CRQC) 在未來變得實際，現有的 TLS 金鑰交換機制將容易受到這些攻擊的影響。對手現在可能會開始收集加密的資料，並希望他們可以在未來解密資料（現在收集，稍後解密）。如果您開發的應用程式依賴於透過 TLS 連線傳遞之資料的長期機密性，您應該考慮在大規模量子電腦可供使用之前遷移至量子後密碼編譯的計劃。AWS 正在為未來做好準備，我們也希望您做好充分準備。



為了保護今天加密的資料免受潛在的未來攻擊， AWS 正在與密碼編譯社群參與開發量子抗性或後量子演算法。 AWS 已實作混合式後量子金鑰交換密碼套件，這些套件結合傳統元素和後量子元素，以確保您的 TLS 連線至少與傳統密碼套件一樣強大。

使用最新版本的 AWS SDKs 時，這些混合式密碼套件可用於您的生產工作負載。如需如何啟用/停用此行為的詳細資訊，請參閱啟用混合式後量子 TLS。



關於 TLS 中的混合式後量子金鑰交換

AWS 使用的演算法是一種混合式演算法，結合了 Elliptic Curve Diffie-Hellman (ECDH)，這是目前在 TLS 中使用的傳統金鑰交換演算法，與 Module-Lattice-Based金鑰封裝機制 (ML-KEM)、公有金鑰加密和金鑰建立演算法，美國國家標準技術研究所 (NIST) 已指定為第一個標準量子後金鑰協議演算法。此混合會獨立使用各演算法，以產生金鑰。然後以密碼編譯方式結合兩個金鑰。

使用方式

如需在 AWS SDKs 中啟用混合式後量子 TLS 的資訊，請參閱啟用混合式後量子 TLS。

進一步了解 PQC

如需國家標準技術研究 (NIST) 的後量子加密法專案的資訊，請參閱後量子加密法。

如需有關 NIST 後量子密碼學標準化的資訊，請參閱後量子密碼學標準化。

啟用混合式後量子 TLS

本主題已移至 AWS SDKs和工具參考指南。

AWS 付款密碼編譯的安全最佳實務

AWS 付款密碼編譯支援許多內建或可選擇性地實作的安全功能，以增強加密金鑰的保護，並確保它們用於其預期用途，包括 IAM 政策、廣泛的政策條件金鑰集，以精簡金鑰政策和 IAM 政策，以及內建對金鑰區塊的 PCI PIN 規則強制執行。

> **Important**
>
> 提供的一般準則不代表完整的安全解決方案。由於並非所有的最佳實務都適用於所有情形，因此這些實務並非為規範性的。

- 金鑰用量和使用模式： AWS 付款密碼編譯遵循並強制執行金鑰用量和使用模式限制，如 ANSI X9 TR 31-2018 互通性安全金鑰交換金鑰區塊規格中所述，並與 PCI PIN 安全要求 18-3 一致。這會限制將單一金鑰用於多種用途的能力，並以密碼編譯方式將金鑰中繼資料（例如允許的操作）繫結至金鑰材料本身。 AWS 付款加密會自動強制執行這些限制，例如金鑰加密金鑰 (TR31_K0_KEY_ENCRYPTION_KEY) 也無法用於資料解密。如需詳細資訊，請參閱了解 AWS 付款密碼編譯金鑰的金鑰屬性。
- 限制共用對稱金鑰材料：只與最多一個其他實體共用對稱金鑰材料（例如 Pin 加密金鑰或金鑰加密金鑰）。如果需要將敏感資料傳輸到更多實體或合作夥伴，請建立其他金鑰。 AWS 付款密碼編譯永遠不會公開對稱金鑰材料或非對稱私有金鑰材料。
- 使用別名或標籤將金鑰與特定使用案例或合作夥伴建立關聯：別名可用來輕鬆表示與金鑰相關聯的使用案例，例如別名/BIN_12345_CVK，以表示與 BIN 12345 相關聯的卡片驗證金鑰。若要提供更多彈性，請考慮建立標籤，例如 bin=12345、use_case=acquiing、country=us、partner=foo。別名和標籤也可以用於限制存取，例如在發行和取得使用案例之間強制執行存取控制。
- 實行最低權限存取：IAM 可用來限制系統而非個人的生產存取，例如禁止個別使用者建立金鑰或執行密碼編譯操作。IAM 也可以用來限制存取可能不適用於您的使用案例的命令和金鑰，例如限制為取得者產生或驗證接腳的能力。使用最低權限存取的另一種方法是限制對特定服務帳戶的敏感操作（例如金鑰匯入）。如需範例，請參閱 AWS 付款密碼編譯身分型政策範例。

另請參閱

- AWS 付款密碼編譯的身分和存取管理
- 《IAM 使用者指南》中的 IAM 安全最佳實務。

AWS 付款密碼編譯的合規驗證

與其他 AWS 服務一樣，客戶需要清楚了解共同的責任模型，以確保安全與合規。作為專門支援付款的服務，遵守適用的 PCI 標準對於了解 AWS 付款密碼編譯客戶尤其重要。 AWS PCI DSS 和 PCI 3DS 評估包括 AWS 付款密碼編譯。共用責任指南中可能參考了這些服務， AWS Artifact可從中取得這些報告。PCI PIN 安全和Point-to-Point加密 (P2PE) 評估專屬於 AWS 付款密碼編譯。

本節提供有關服務合規狀態和範圍的資訊，以及有助於規劃應用程式 PCI PIN 安全和 PCI P2PE 評估的資訊。

主題

- 服務的合規
- PIN 合規規劃
- 在 P2PE 解決方案中使用 AWS 付款密碼編譯解密元件

服務的合規

在多個合規計畫中，第三方稽核人員會評估 AWS 付款密碼編譯的安全性和 AWS 合規性。這些包括 SOC、PCI 等。

AWS 除了 PCI DSS 和 PCI 3DS 之外，還針對數個 PCI 標準評估了付款密碼編譯。其中包括 PCI PIN 安全 (PCI PIN) 和 PCI Point-to-Point(P2PE) 加密。如需可用的證明和合規指南 AWS Artifact ，請參閱 。

如需特定合規計劃範圍內 AWS 的服務清單，請參閱合規計劃範圍內的 AWS 服務。如需一般資訊，請參閱 AWS 合規計劃 。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載報告 in AWS Artifact

您在使用 AWS 付款密碼編譯時的合規責任取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。 AWS 提供下列資源來協助合規：

- 安全與合規快速入門指南—這些部署指南討論架構考量，並提供在 上部署以安全與合規為中心之基準環境的步驟 AWS。
- AWS 合規資源 - 此工作手冊和指南集合可能適用於您的產業和位置。
- 《 AWS Config 開發人員指南》中的使用規則評估資源AWS Config：評估資源組態符合內部實務、產業準則和法規的程度。

- AWS Security Hub CSPM- AWS 此服務提供 內安全狀態的完整檢視 AWS ，協助您檢查是否符合安全產業標準和最佳實務。

PIN 合規規劃

本指南說明您將需要為使用 AWS 付款密碼編譯的 PIN 處理應用程式準備 PCI PIN 評估的文件和證據。

與其他 AWS 服務 和合規標準一樣，您有責任安全地使用服務、設定存取控制，並使用符合 PCI PIN 要求的安全參數。本指南將在符合需求時討論這些組態。

主題

- 常見主題
- 評估範圍
- 交易處理操作

常見主題

將應用程式從連線至 HSM 遷移至受管服務，例如 AWS 付款密碼編譯，為客戶及其評估者帶來常見問題和概念。本節提供的資訊說明 服務的安全使用如何解決這些情況。

主題

- 共同的責任
- 最低 HSM 組態
- 客戶與 APC 之間的金鑰交換

共同的責任

承擔應用程式完整安全與合規責任的客戶將重組其合規，以利用 AWS Payment Cryptography 的金鑰管理、安全控制和受管 HSM 功能 (「服務」)。這將完全將一些要求轉移到 AWS，如 AWS Payment Cryptography 的第三方評估所證明。有些需求會在客戶的應用程式和服務之間共用。應用程式負責：

- 為服務提供準確的資訊
- 根據服務的建議和 PCI PIN 安全要求使用安全控制
- 使用 服務提供的工具實作必要的安全控制

客戶及其評估人員將使用 上發佈的共同責任和實作指南， AWS Artifact 以實作控制和控制監控，然後規劃並完成評估。

最低 HSM 組態

PCI Data Security Standard 是其他 PCI 標準的基礎標準，需要以其函數所需的最低功能來設定所有系統。PCI PIN、P2PE 和其他解決方案標準將此要求套用至解決方案中的 HSMs。HSMs只能啟用解決方案所需的函數。

AWS 服務應視為系統，並針對最低必要功能進行設定。AWS 上的支付卡產業資料安全標準 (PCI DSS) 4.0 版建議使用 IAM 為解決方案所使用的每個 AWS 服務設定最低功能。這也適用於 AWS 付款密碼編譯。IAM 政策允許精細的許可，將密碼編譯函數限制為僅依賴它們的應用程式元件。

客戶與 APC 之間的金鑰交換

PIN 安全要求 8-4 和 15-2 需要公有金鑰才能交換，並對金鑰的載入進行身分驗證並保護完整性。對於功能上描述於 ANSI/ASC X9 TR-34 且受 PCI PIN Annex A 規範的 POI 遠端金鑰載入，公有金鑰通常會在由 Annex A2-compliant憑證授權單位簽署的憑證中傳遞。對於組織之間的交換，公有金鑰會使用其他機制來確保真實性和完整性。

客戶和 AWS 之間的所有互動都是透過 AWS APIs 進行，可相互驗證每個 API 呼叫，並確保使用 TLS 進行呼叫和回應的完整性。客戶應用程式的身分驗證是由 AWS Identity and Access Management 使用安全權杖和 SigV4 等機制進行管理。AWS API 端點是由客戶使用內建於 AWS SDKs的 TLS 伺服器身分驗證進行身分驗證。然後，TLS 會確保客戶和每個 AWS API 之間傳遞的所有資料的機密性和完整性。

APC APIs GetParametersForImport 和 ImportKey 會實作從客戶到服務的金鑰傳輸。雖然 GetParametersForImport 提供的憑證授權機構 (CA) 不符合 Annex A2-compliant，但對帳戶而言是安全且唯一的。雖然無法依賴此 CA 以符合要求 8-4 和 15-2，但它確實提供匯入金鑰的完整性驗證。您也可以利用 GetCertificateSigningRequest API 來使用自己的 CA。

提供公有金鑰身分驗證和完整性保證的機制包括：

- AWS API 身分驗證提供的身分驗證
- 金鑰的完整性是由 GetParametersForImport 所提供憑證的 MAC 功能所提供，即使憑證中的身分資訊不受信任也一樣。TLS 用來保護客戶和 AWS 之間工作階段的 MAC 也會確保金鑰的完整性

APC 提供的憑證和金鑰區塊符合 Annex A1，其指定非對稱方法對憑證和金鑰保護的要求。

評估範圍

規劃任何評估的第一步是記錄範圍。對於 PCI PIN，範圍是保護 PINs的系統和程序，包括保護密碼編譯金鑰和裝置 - 付款終端機，也稱為points-of-interaction(POI)、HSMs 和其他安全密碼編譯裝置 (SCD)。

我們不會解決您保留完全責任的要求，因為這些地址區域不在服務範圍內。例如，付款終端機的組態和佈建。請參閱 PCI PIN 的 AWS 付款密碼編譯共同責任指南，可在 取得 AWS Artifact

主題

- 共同的責任
- 高階網路圖表
- 索引鍵資料表
- 文件參考

共同的責任

AWS 付款密碼編譯是一種加密和支援組織 (ESO) 和 PIN 取得第三方服務機構 (TPS)，如 Visa PIN 安全計劃所定義，並列在「Amazon Web Services， LLC」下的 Visa 全球服務供應商註冊表中。這表示 Visa 允許此服務供 PIN 取得第三方 VisaNet 處理器 (VNP)、做為服務提供者的 PIN 取得客戶 VisaNet 處理器，以及其他 TPS 和 ESO 提供者使用，而不需要客戶 PIN 評估者 (PCI 合格 PIN 評估者或 PCI QPA) 進一步評估。

其他卡片品牌或付款網路供應商可能依賴 Visa PIN 安全計劃或擁有自己的計劃。有關其他付款網路計劃的服務合規問題，請聯絡 AWS 支援 。

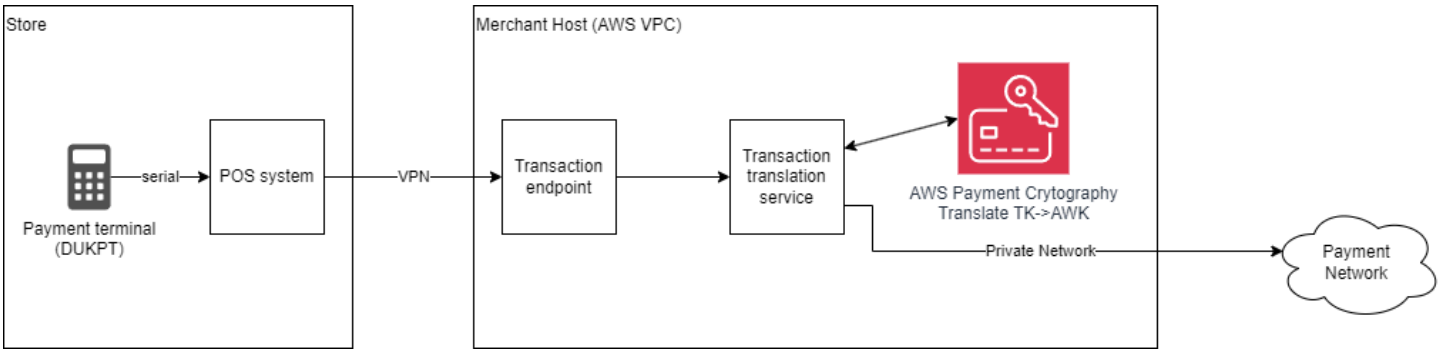
AWS 提供 PCI PIN 安全合規證明 (AOC) 和 AWS 付款密碼編譯的共同責任指南 AWS Artifact。在 PIN 處理中使用服務提供者很常見，但直到 3.1 版的 PCI PIN 安全標準不會解決第三方服務提供者管理問題。Visa PIN 安全計劃也一樣。客戶 QPA 已遵循 PCI DSS AOC 和共同責任指南所建立的模型，該指南將 AWS合規視為成功測試適用要求。

高階網路圖表

PCI PIN 報告範本需要：「對於參與處理 PIN 型交易的實體，提供描述具有相關聯金鑰類型用量之 PIN 型交易流程的網路示意圖。此外，使用非對稱技術進行遠端金鑰分發KIFs 和實體應該提供金鑰材料流程

AWS 付款密碼編譯已報告 PCI PIN 評估的內部服務結構。您的圖表將說明呼叫服務 APIs以進行 PIN 處理。

使用 AWS 付款密碼編譯的 PIN 應用程式之高階網路圖表範例：



索引鍵資料表

報告要求直接或間接列出保護 PINs的所有金鑰。服務中存在的任何金鑰都可以使用 ListKeysAPI 列出。

請務必提供擁有應用程式金鑰的所有區域和帳戶的金鑰清單。

文件參考

有關安全使用 AWS 付款密碼編譯的供應商文件和建議，請參閱 使用者指南和 API 參考。這些會在本指南中適當地連結。

交易處理操作

PCI PIN 要求會整理在控制目標中。每個控制目標都會將保護 PINs 安全方面的要求分組。

主題

- 控制目標 1：用於受這些要求控管之交易的 PINs碼會使用設備和方法處理，以確保它們的安全。
- 控制目標 2：用於 PIN 加密/解密和相關金鑰管理的加密金鑰是使用程序建立，以確保無法預測任何金鑰或判斷某些金鑰比其他金鑰更有可能。
- 控制目標 3：以安全的方式傳遞或傳輸金鑰。
- 控制目標 4：以安全的方式處理對 HSMs 和 POI PIN 接受裝置的金鑰載入。
- 控制目標 5：金鑰的使用方式可防止或偵測其未經授權的使用。
- 控制目標 6：以安全的方式管理金鑰。
- 控制目標 7：用於處理 PINs和金鑰的設備是以安全的方式管理。

控制目標 1：用於受這些要求控管之交易的 PINs碼會使用設備和方法處理，以確保它們的安全。

要求 1： AWS 付款密碼編譯使用的 HSMs 已評估為 PCI PIN 評估的一部分。對於使用 服務的客戶，要求 1-3 和 1-4 相對於服務管理的 HSM 是「就地」。HSM 的問題清單會指出 AWS QPA 已向 證明測試。PIN 合規證明可供參考 AWS Artifact。解決方案中的其他 SCD，例如 POI，將需要清查和參考。

要求 2：程序的文件必須指定如何在向人員洩露、實作PINs 翻譯協定以及在線上和離線處理期間保護 (PIN) 方面保護持卡人 PIN。此外，您的文件應包含每個區域中使用的密碼編譯金鑰管理方法摘要。

要求 3：POI 必須設定為安全 PIN 加密和傳輸。 AWS 付款加密僅支援要求 3-3 中指定的 PIN 區塊轉譯。

要求 4：應用程式不得存放 PIN 區塊。PIN 區塊即使已加密，也不得保留在交易日誌或日誌中。服務不會儲存 PIN 區塊，而且 PIN 評估會驗證它們不在日誌中。

請注意，PCI PIN 安全標準適用於取得「在 ATMs 和point-of-sale (POS) 終端機線上和離線支付卡交易處理期間的個人識別號碼 (PIN) 資料的安全管理、處理和傳輸」，如標準所述。不過，標準通常用於評估密碼編譯金鑰管理是否有超出該預期範圍的付款。這可能包括存放 PINs發行者使用案例。這些案例要求的例外狀況應與評估的預期對象達成一致。

控制目標 2：用於 PIN 加密/解密和相關金鑰管理的加密金鑰是使用程序建立，以確保無法預測任何金鑰或判斷某些金鑰比其他金鑰更有可能。

要求 5： AWS 付款密碼編譯產生的金鑰已評估為 PCI PIN 評估的一部分。這可以在「產生者」資料欄中指定。

要求 6： AWS 付款密碼編譯中保留之金鑰的安全控制已評估為服務 PCI PIN 評估的一部分。包含與您應用程式內和任何其他服務提供者產生金鑰相關的安全控制描述。

要求 7：您必須擁有金鑰產生政策文件，該文件應指定金鑰的產生方式，且所有受影響方都必須了解這些程序/政策。使用 APC API 建立金鑰的程序應包括使用具有金鑰建立許可的角色，以及執行指令碼或建立 key. AWS CloudTrail logs 之其他程式碼的核准。 日誌包含具有日期和時間、金鑰 ARN 和使用者 ID 的所有 CreateKey 事件。存取實體媒體的 HSM 序號和日誌被評估為服務 PIN 評估的一部分。

控制目標 3：以安全的方式傳遞或傳輸金鑰。

要求 8： AWS 付款密碼編譯的金鑰傳遞已評估為 PCI PIN 評估的一部分。匯入付款 AWS 密碼編譯之前和匯出之後，您將需要記錄傳輸的金鑰保護機制。服務提供所有金鑰的金鑰檢查值，以驗證正確的傳輸。

要求 8-4 要求以保護公有金鑰完整性和真實性的方式傳遞公有金鑰。您的應用程式與 之間的傳遞， AWS 是由應用程式的身分驗證使用 AWS Identity and Access Management 方法 AWS，透過 TLS 伺服器憑證對應用程式進行 AWS API 端點身分驗證所控制。此外，從 AWS 付款密碼編譯匯出或匯入的公有金鑰具有暫時性、客戶特定 CAs 簽署的憑證 （ 請參閱 GetPublicKeyCertificate、GetParametersForImport 和 GetParametersForExport)。這些 CAs不能用作唯一身分驗證方法，因為它們不符合 PCI PIN 安全附件 A2。不過，憑證仍會透過提供身分驗證的 IAM 為公有金鑰提供完整性保證。

使用非對稱方法與業務合作夥伴交換公有金鑰時，您必須使用安全的檔案交換網站等，透過通訊管道提供 以進行企業身分驗證。

要求 9：服務不使用或直接支援純文字金鑰元件。

要求 10：服務會強制執行保護傳輸金鑰的相對金鑰強度。從 AWS 付款密碼編譯匯出至 之前和匯出之後，您必須負責金鑰傳遞，並使用適用於金鑰匯入、匯出和產生之正確 API 和 TR-31 參數。您應該有記錄的程序來描述金鑰傳輸機制，以及用於傳輸的密碼編譯金鑰清單。

要求 11：程序的文件必須指定傳遞金鑰的方式。使用 AWS 付款密碼編譯 API 的金鑰傳遞程序應包括使用具有金鑰匯入和匯出許可的角色，以及執行指令碼或建立 key. AWS CloudTrail logs 的其他程式碼的核准，其中包含所有 ImportKey 和 ExportKey 事件。

控制目標 4：以安全的方式處理對 HSMs 和 POI PIN 接受裝置的金鑰載入。

要求 12：您負責從元件或共享載入金鑰。HSM 主金鑰的管理被評估為服務 PIN 評估的一部分。 AWS 付款密碼編譯不會從個別共用或元件載入金鑰。請參閱 密碼編譯詳細資訊 一節。

要求 13 和 14：在匯入至服務之前和匯出之後，您將需要描述傳輸的金鑰保護。

要求 15： AWS 付款密碼編譯提供服務和公有金鑰完整性保證中所有金鑰的金鑰檢查值。您的應用程式負責使用這些檢查，在匯入至服務或從服務匯出後驗證金鑰。您應該記錄程序，以確保驗證機制已就緒。

要求 15-2 要求以保護公有金鑰完整性和真實性的方式載入公有金鑰。ImportKey 與 GetParametersForImport 一起提供提供的簽署憑證驗證。如果提供的憑證是自我簽署的，則身分驗證必須由不同的機制提供，例如安全的檔案交換。

要求 16：程序的文件必須指定 金鑰如何載入服務。使用 API 匯入金鑰的程序應包括使用具有金鑰匯入許可的角色，以及執行指令碼或載入 key. AWS CloudTrail logs ImportKey之其他程式碼的核准。您應該在文件中包含記錄機制。此服務為所有金鑰提供金鑰檢查值，以驗證正確的金鑰載入。

控制目標 5：金鑰的使用方式可防止或偵測其未經授權的使用。

要求 17：此服務為啟用追蹤金鑰共用關係的金鑰提供機制，例如標籤和別名。此外，金鑰檢查值應分開保留，以證明共用金鑰時不會使用已知或預設的金鑰值。

要求 18：此服務透過 GetKey 和 ListKeys 提供金鑰完整性檢查，並透過 提供金鑰管理事件 AWS CloudTrail，可用於偵測未經授權的替換或監控各方之間的金鑰同步。服務只會將金鑰存放在金鑰區塊中。從 AWS 付款密碼編譯匯出至 之前和匯出之後，您必須負責金鑰儲存和使用 。

如果在處理 PIN 型交易或非預期金鑰管理事件期間發生任何差異，您應該制定立即調查的程序。

要求 19：服務僅在金鑰區塊中使用金鑰，強制 KeyUsage、KeyModeOfUse 和所有操作的其他金鑰屬性。這包括對私有金鑰操作的限制。您應該將公有金鑰用於單一用途，例如加密或數位簽章驗證，但不能同時使用兩者。您應該針對生產和測試/開發系統使用不同的帳戶。

要求 20：您仍需對此要求負責。

控制目標 6：以安全的方式管理金鑰。

要求 21：金鑰儲存和與 AWS 付款密碼編譯搭配使用已評估為服務 PCI PIN 評估的一部分。對於與金鑰元件相關的儲存需求，您需負責存放它們，如 21-2 和 21-3 所述。在匯入至 服務之前和匯出自服務之後，您將需要在政策文件中描述金鑰保護機制。

要求 22： AWS 付款密碼編譯的金鑰洩露程序已評估為服務 PCI PIN 評估的一部分。您需要描述關鍵入侵偵測和回應程序，包括監控和回應來自 的通知 AWS。

要求 23： AWS 付款密碼編譯不支援變體或其他可逆金鑰計算方法。APC 主金鑰或由它們加密的金鑰絕不可供客戶使用。可逆金鑰計算的使用已評估為服務 PCI PIN 評估的一部分。

要求 24：內部秘密和私有金鑰的銷毀實務 AWS 付款密碼編譯已評估為服務 PCI PIN 評估的一部分。從 APC 匯出至 之前和匯出之後，您將需要描述金鑰的金鑰銷毀程序。關鍵元件相關的銷毀要求 (24-2.2 和 24-2.3) 仍由您負責。

要求 25：在 AWS Payment Cryptography 中存取秘密和私有金鑰已評估為服務 PCI PIN 評估的一部分。從 AWS 付款密碼編譯匯出至 之前和匯出之後，您將需要擁有金鑰存取控制的程序和文件。

要求 26：您將需要描述對在服務之外使用之金鑰、金鑰元件或相關資料的任何存取的記錄。您的應用程式使用服務執行的所有金鑰管理活動的日誌可透過 取得 AWS CloudTrail。

要求 27：您將需要描述在服務之外使用的金鑰、金鑰元件或相關資料的備份程序。

要求 28：使用 API 進行所有金鑰管理的程序應包括使用具有金鑰管理許可的角色，以及執行指令碼或管理金鑰的其他程式碼的核准。 AWS CloudTrail 日誌包含所有金鑰管理事件

控制目標 7：用於處理 PINs和金鑰的設備是以安全的方式管理。

要求 29：使用 AWS 付款密碼編譯可滿足您對 HSMs 的實體和邏輯保護的要求。

要求 30：您的應用程式將保留對 POI 裝置要求的所有實體和邏輯保護的責任。

要求 31： AWS 付款密碼編譯所使用的安全密碼編譯裝置 (SCD) 的保護已評估為服務 PCI PIN 評估的一部分。您需要示範對應用程式使用的任何其他 SCDs的保護。

要求 32： AWS 付款密碼編譯所使用的 SCDs 的使用已評估為服務 PCI PIN 評估的一部分。您需要示範應用程式所使用的任何其他 SCDs的存取控制和保護。

要求 33：您將需要描述受您控制的任何 PIN 處理設備的保護。

在 P2PE 解決方案中使用 AWS 付款密碼編譯解密元件

PCI P2PE 解決方案可以使用AWS 付款密碼編譯解密元件。這記錄在 PCI Point-to-Point加密：安全要求和測試程序 (P2PE 解決方案和第三方和/或 P2PE 元件提供者的使用）：「解決方案提供者（或作為解決方案提供者的商家）可以將某些 P2PE 函數外包給 PCI 列出的 P2PE 元件提供者，並在其 P2PE 驗證報告 (P-ROV) 中報告 PCI 列出的 P2PE 元件的使用」，該報告可在 PCI 網站上取得。

與其他 AWS 服務和合規標準一樣，您有責任安全地使用服務、設定存取控制，並使用符合 PCI P2PE 要求的安全參數。AWS Payment Cryptography P2PE Decryption Component User's Guide 可在 上取得， AWS Artifact提供將 AWS Payment Cryptography 與您的 PCI P2PE Solution 整合的詳細說明，以及合規報告所需的年度解密元件報告。

AWS 付款密碼編譯的身分和存取管理

AWS Identity and Access Management (IAM) 是 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證 （ 登入 ） 和授權 （ 具有許可 ），以使用 AWS 付款密碼編譯資源。IAM 是您可以免費使用 AWS 服務 的 。

主題

- 目標對象
- 使用身分驗證
- 使用政策管理存取權
- AWS 付款密碼編譯如何與 IAM 搭配使用
- AWS 付款密碼編譯身分型政策範例
- AWS 付款密碼編譯的資源型政策
- AWS 付款密碼編譯的多方核准
- 對 AWS 付款密碼編譯身分和存取權進行故障診斷

目標對象

您的使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 對 AWS 付款密碼編譯身分和存取權進行故障診斷)
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 AWS 付款密碼編譯如何與 IAM 搭配使用)
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 AWS 付款密碼編譯身分型政策範例)

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

對於程式設計存取， AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的 API 請求的AWS 第 4 版簽署程序。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分具有對所有 AWS 服務 和 資源的完整存取權。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取 。

IAM 群組會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過從使用者切換到 IAM 角色 （主控台） 或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS ， 會評估這些政策。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的在受管政策與內嵌政策之間選擇。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。您必須在資源型政策中指定主體。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF和 Amazon VPC 是支援 ACLs的服務範例。如需進一步了解 ACL，請參閱《Amazon Simple Storage Service 開發人員指南》中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《 IAM 使用者指南》中的 IAM 實體許可界限。
- 服務控制政策 (SCP) — 為 AWS Organizations中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的資源控制政策 (RCP)。
- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參《IAM 使用者指南》中的工作階段政策。

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

AWS 付款密碼編譯如何與 IAM 搭配使用

在您使用 IAM 管理 AWS 付款密碼編譯的存取權之前，您應該了解哪些 IAM 功能可與 AWS 付款密碼編譯搭配使用。若要全面了解 AWS 付款密碼編譯和其他 AWS 服務如何與 IAM 搭配使用，請參閱《IAM 使用者指南》中的與 IAM AWS 搭配使用的 服務。

主題

- AWS 付款密碼編譯身分型政策
- 以 AWS 付款密碼編譯標籤為基礎的授權

AWS 付款密碼編譯身分型政策

透過 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。 AWS 付款密碼編譯支援特定動作、資源和條件索引鍵。若要了解您在 JSON 政策中使用的所有元素，請參閱 IAM 使用者指南中的 JSON 政策元素參考。

動作

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策會使用動作來授予執行相關聯動作的許可。

AWS 付款密碼編譯中的政策動作在動作之前使用下列字首：`payment-cryptography:`。例如，若要授予某人執行 AWS 付款密碼編譯 `VerifyCardData` API 操作的許可，請在其政策中包含 `payment-cryptography:VerifyCardData`動作。政策陳述式必須包含 `Action` 或 `NotAction` 元素。 AWS 付款密碼編譯會定義自己的一組動作，描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
```

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

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定以字詞開頭的所有動作 `List`（例如 `ListKeys`和 `ListAliases`），請包含下列動作：

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

若要查看 AWS 付款密碼編譯動作的清單，請參閱《IAM 使用者指南》中的AWS 付款密碼編譯定義的動作。

Resources

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

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

payment-cryptography 金鑰資源具有下列 ARN：

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

如需 ARNs 格式的詳細資訊，請參閱 Amazon Resource Name (ARNs) AWS 和服務命名空間。

例如，若要在陳述式中指定 `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h` 執行個體，請使用以下 ARN：

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

若要指定屬於特定帳戶的所有金鑰，請使用萬用字元 (*)：

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

有些 AWS 付款密碼編譯動作無法在特定資源上執行，例如用於建立金鑰的動作。在這些情況下，您必須使用萬用字元 (*)。

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

若要在單一陳述式中指定多個資源，請使用逗號，如下所示：

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

範例

若要檢視 AWS 付款密碼編譯身分型政策的範例，請參閱 AWS 付款密碼編譯身分型政策範例。

以 AWS 付款密碼編譯標籤為基礎的授權

您可以將標籤連接至 AWS 付款密碼編譯資源，或在請求中將標籤傳遞至 AWS 付款密碼編譯。如需根據標籤控制存取，請使用 payment-cryptography:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件索引鍵，在政策的條件元素中，提供標籤資訊。

AWS 付款密碼編譯身分型政策範例

根據預設，IAM 使用者和角色沒有建立或修改 AWS 付款密碼編譯資源的許可。他們也無法使用 AWS 管理主控台 AWS CLI或 AWS API 執行任務。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 作業的所需許可。管理員接著必須將這些政策連接至需要這些許可的 IAM 使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立政策。

主題

- 政策最佳實務
- 使用 AWS 付款密碼編譯主控台
- 允許使用者檢視他們自己的許可

- 能夠存取 AWS 付款密碼學的所有層面
- 能夠使用指定的金鑰呼叫 APIs
- 能夠明確拒絕資源

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 AWS 付款密碼編譯資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 CloudFormation。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱《IAM 使用者指南》中的 IAM 安全最佳實務。

使用 AWS 付款密碼編譯主控台

若要存取 AWS 付款密碼編譯主控台，您必須擁有一組最低許可。這些許可必須允許您列出和檢視 AWS 帳戶中 AWS 付款密碼編譯資源的詳細資訊。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (IAM 使用者或角色) 而言，主控台就無法如預期運作。

為了確保這些實體仍然可以使用 AWS 付款密碼編譯主控台，請將下列 AWS 受管政策連接至實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合您嘗試執行之 API 操作的動作就可以了。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台或使用 或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
    "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": "*"
        }
    ]
}
```

能夠存取 AWS 付款密碼學的所有層面

> **Warning**
>
> 此範例提供廣泛的許可，不建議使用。請改為考慮最低權限的存取模型。

在此範例中，您想要授予 AWS 您帳戶中的 IAM 使用者存取所有 AWS 付款密碼編譯金鑰，以及呼叫所有 AWS 付款密碼編譯 API 的能力，包括 `ControlPlane`和 `DataPlane`操作。

JSON

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

能夠使用指定的金鑰呼叫 APIs

在此範例中，您想要授予 AWS 帳戶中的 IAM 使用者存取其中一個 AWS 付款密碼編譯金鑰，`arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`然後在兩個 APIs `GenerateCardValidationData`和 中使用此資源`VerifyCardValidationData`。相反地，IAM 使用者將無法存取在其他操作上使用此金鑰，例如 `DeleteKey`或 `ExportKey`

資源可以是 金鑰，字首為 `key`或 別名，字首為 `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"
               ]
            }
      ]
   }
```

能夠明確拒絕資源

> **Warning**
>
> 仔細考慮授予萬用字元存取的影響。請改為考慮最低權限模型。

在此範例中，您想要允許 AWS 帳戶中的 IAM 使用者存取任何 AWS 付款密碼編譯金鑰，但想要拒絕某個特定金鑰的許可。除了拒絕陳述式中指定的金鑰之外，使用者可以存取 GenerateCardData VerifyCardData和所有金鑰。

JSON

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

```
            "Action": [
                "payment-cryptography:VerifyCardValidationData",
                "payment-cryptography:GenerateCardValidationData"
            ],
            "Resource": [
              "arn:aws:payment-cryptography:us-east-2:111122223333:key/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
              "payment-cryptography:GenerateCardValidationData"
            ],
            "Resource": [
              "arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h"
            ]
        }
    ]
}
```

AWS 付款密碼編譯的資源型政策

以資源為基礎的政策是您連接到資源的 JSON 政策文件，例如 AWS 付款密碼編譯金鑰。在資源型政策中，您可以指定誰可以存取金鑰，以及他們可以對其執行的動作。您可以使用資源型政策：

- 將單一金鑰的存取權授予多個使用者和角色。
- 將存取權授予其他 AWS 帳戶中的使用者或角色。

主題

- 考量事項
- 管理以資源為基礎的政策
- 資源型政策範例

當您將資源型政策連接至 AWS 付款密碼編譯金鑰時， AWS 付款密碼編譯會使用 IAM 政策評估邏輯來判斷指定的委託人是否獲授權執行請求的動作。若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的委託人。跨帳戶存取需要兩個政策：

1. 以資源為基礎的政策（金鑰擁有者的帳戶）— 金鑰擁有者使用 PutResourcePolicy 授予呼叫者帳戶或 IAM 主體的存取權。
2. 身分型政策（來電者的帳戶）— 發起人的 IAM 管理員也必須在發起人的 IAM 政策中允許 AWS 付款密碼編譯動作（例如 payment-cryptography:EncryptData)。

這兩個政策都必須允許 動作。如果其中一個遺失，則使用 拒絕跨帳戶請求AccessDeniedException。

如果以資源為基礎的政策授予相同帳戶中的委託人存取權，則不需要其他以身分為基礎的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與以資源為基礎的原則有何差異。

> **資源政策控制平面操作**
>
> 以資源為基礎的政策不適用於資源政策控制平面操作GetResourcePolicy，例如 PutResourcePolicy、 和 DeleteResourcePolicy。這可防止潛在鎖定案例，其中資源政策可能會拒絕修改或移除政策本身的能力。存取這些控制平面操作僅受 IAM 身分型政策的約束。

考量事項

將資源型政策與 AWS 付款密碼編譯搭配使用時，請記住下列事項。

- AWS 付款密碼編譯會自動強制不公開存取金鑰。您無法建立將存取權授予匿名或公有主體的資源型政策。所有對 AWS Payment Cryptography 金鑰的存取都需要經過驗證的 AWS 主體，而且一律會封鎖公開存取。
- 每個金鑰套用以資源為基礎的政策。每個 AWS 付款密碼編譯金鑰最多可連接一個以資源為基礎的政策。
- 資源型政策不適用於別名。當您依索引鍵別名參考索引鍵時，會評估連接至基礎索引鍵的資源政策。
- 以資源為基礎的政策目前不適用於使用多區域金鑰複寫建立的唯讀複本區域金鑰。資源政策只能連接到主要區域金鑰。
- 資源型政策中的 Resource元素必須與政策所連接之金鑰的 ARN 相符"*"或完全相符。"*" 建議使用 ，因為它允許在多個金鑰之間重複使用相同的政策文件。
- 資源政策管理 APIs(PutResourcePolicy、 GetResourcePolicy和 DeleteResourcePolicy) 僅限於 AWS 帳戶 擁有金鑰的 。只有金鑰擁有者帳戶中的主體才能管理資源政策。

管理以資源為基礎的政策

您可以使用 AWS CLI 或 AWS API 來管理 AWS 付款密碼編譯金鑰的資源型政策。若要使用此命令，請以您自己的資訊取代範例命令中的*########*。

連接以資源為基礎的政策

使用 PutResourcePolicy API 動作或 put-resource-policy CLI 命令，將資源型政策連接至金鑰。如果政策已存在，命令會取代它。

下列範例會將資源型政策從 JSON 檔案連接至金鑰。

```
aws payment-cryptography put-resource-policy \
    --resource-arn arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h \
    --policy file://policy.json
```

擷取以資源為基礎的政策

使用 GetResourcePolicy API 動作或 get-resource-policy CLI 命令來擷取連接到金鑰的資源型政策。

下列範例會擷取連接至金鑰的資源型政策。

```
aws payment-cryptography get-resource-policy \
    --resource-arn arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h
```

回應會傳回政策文件：

```
{
    "Policy": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
                },
                "Action": [
                    "payment-cryptography:EncryptData",
                    "payment-cryptography:DecryptData"
```

```
                ],
                "Resource": "*"
            }
        ]
    }
}
```

刪除以資源為基礎的政策

使用 DeleteResourcePolicy API 動作或 delete-resource-policy CLI 命令，從金鑰中移除資源型政策。

下列範例會刪除連接至金鑰的資源型政策。

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

資源型政策範例

授予金鑰的跨帳戶存取權

下列資源型政策授予另一個 AWS 帳戶中的角色許可，以使用 AWS 付款密碼編譯金鑰進行密碼編譯操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
            },
            "Action": [
                "payment-cryptography:GenerateCardValidationData",
                "payment-cryptography:VerifyCardValidationData"
            ],
            "Resource": "*"
        }
    ]
}
```

將不同的許可授予不同的帳戶

下列資源型政策示範如何將不同許可授予不同帳戶中的委託人。在此範例中，一個帳戶中的 3DS 存取控制伺服器 (ACS) 可以產生卡片驗證資料，而不同帳戶中的付款授權服務只能驗證 3DS 加密。

```
{
    "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": "*"
        }
    ]
}
```

AWS 付款密碼編譯的多方核准

AWS 付款密碼編譯與來自 Amazon Web Services Organizations 的多方核准 (MPA) 整合，透過分散式核准程序協助保護關鍵操作。使用 MPA，您可以要求多個信任的個人在執行特定 AWS 付款密碼編譯操作之前核准這些操作。

主題

- 概觀

- 受保護的操作
- 先決條件
- 啟用和停用 MPA
- 開始使用
- 範例：匯入已啟用 MPA 的根憑證
- AWS CloudTrail MPA 事件的記錄
- 檢查請求狀態和處理失敗

概觀

多方核准需要一組信任的個人核准，才能繼續操作，為敏感的 AWS 付款密碼編譯操作新增額外的安全層。如果單一登入資料遭到入侵，這有助於防止未經授權的變更，並防止單一個人單方面進行變更。

核准團隊是組織中一組核准者，您指定核准或拒絕受保護的操作請求。核准程序完全由您組織的核准者管理。沒有 AWS 人員參與核准或拒絕請求。

為受保護的操作啟用 MPA 時，會發生下列情況：

1. 請求者啟動受保護的操作。
2. MPA 會建立核准工作階段並通知核准團隊成員。
3. 核准團隊成員會檢閱請求，並透過 MPA 入口網站核准或拒絕請求。
4. 一旦達到必要的核准最低閾值，操作就會繼續。如果核准團隊拒絕請求，或在達到核准閾值之前允許的工作階段時間過期，則不會執行操作。在這兩種情況下，請求者都必須提交新的請求，才能重試操作。

> **Note**
>
> 在啟用 MPA 的情況下匯入根 CA 憑證時， `RequesterComment` 參數是強制性的。此註解包含在傳送給核准團隊的核准通知中，提供請求的內容。

受保護的操作

AWS 付款密碼編譯支援下列操作的 MPA：

- ImportKey 使用RootCertificatePublicKey金鑰材料 — 匯入根公有金鑰憑證是一項關鍵操作，因為根憑證會使用 TR-34 等非對稱金鑰交換，為所有後續金鑰匯入和匯出建立信任錨點。此操作需要多方核准，有助於確保沒有任何個人可以單方面建立或變更 AWS 付款密碼編譯金鑰的信任根。

先決條件

您必須先完成下列先決條件，才能搭配 AWS 付款密碼編譯使用 MPA：

- 在 Amazon Web Services Organizations 環境中設定 MPA。如需說明，請參閱什麼是多方核准？ 多方核准使用者指南中的 。
- 建立至少一個具有所需核准者的核准團隊。
- 使用 與包含您的 AWS 付款密碼編譯金鑰 AWS 帳戶 的 共用核准團隊 AWS Resource Access Manager。
- 組織中的管理帳戶必須選擇加入多方核准。

啟用和停用 MPA

設定核准團隊之後，您可以將團隊與您的帳戶建立關聯，以啟用 MPA for AWS Payment Cryptography。您也可以取消團隊關聯來停用 MPA，但取消關聯需要目前關聯的核准團隊核准。

啟用 MPA

使用 AssociateMpaTeam API 動作或 associate-mpa-team CLI 命令，將核准團隊與您的 AWS 付款密碼帳戶建立關聯。建立關聯後，受保護的操作需要團隊核准才能繼續。

```
aws payment-cryptography associate-mpa-team \
    --team-arn arn:aws:mpa:us-east-1:111122223333:team/my-approval-team
```

停用 MPA

使用 DisassociateMpaTeam API 動作或 disassociate-mpa-team CLI 命令移除核准團隊關聯。取消團隊關聯本身是受保護的操作，需要目前關聯的核准團隊核准。

```
aws payment-cryptography disassociate-mpa-team \
    --team-arn arn:aws:mpa:us-east-1:111122223333:team/my-approval-team
```

> **Important**
>
> 停用 MPA 需要目前相關聯核准團隊的核准。這可確保沒有任何一個人可以單方面移除多方核准保護。

> **Note**
>
> associate-mpa-team 和 的 `--requester-comment` 參數是選用的disassociate-mpa-team。

開始使用

若要開始使用 MPA for AWS Payment Cryptography，請參閱多方核准使用者指南以取得詳細的設定指示，包括如何建立核准團隊、設定核准政策和管理核准工作階段。

範例：匯入已啟用 MPA 的根憑證

啟用 MPA 且核准團隊與 `ImportKey`的操作相關聯後`RootCertificatePublicKey`，匯入請求需要核准才能繼續。

1. 請求者呼叫 `import-key` 來匯入根公有金鑰憑證。若要使用此命令，請以您自己的資訊取代範例命令中的*########*。

```
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"
```

   回應會傳回 `KeyState` 設定為 的金鑰CREATE_IN_PROGRESS，表示請求正在等待核准。回應也包含`MpaStatus`有關核准工作階段的詳細資訊：

```
{
```

```
    "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. 由於已啟用 MPA，請求不會立即完成。反之， AWS 付款密碼編譯會建立核准工作階段，並傳回表示核准待定的回應。
3. 核准團隊成員會收到通知，並透過 MPA 入口網站檢閱請求。一旦所需的核准者數目核准請求，匯入操作就會繼續，並匯入根憑證。

AWS CloudTrail MPA 事件的記錄

啟用 MPA 時，當核准工作階段完成AWS CloudTrail時， AWS 付款密碼編譯會將服務事件記錄到 。這些事件會記錄核准程序的結果，包括請求是否已核准或失敗。您可以使用這些日誌來稽核 MPA 活動，並追蹤受保護操作的狀態。

MPA 相關的 CloudTrail 事件包含 中的下列欄位`serviceEventDetails`：

- `keyArn` — 受 操作影響之金鑰的 ARN。
- `operation` — 請求的受保護操作。
- `mpaSessionArn` — MPA 核准工作階段的 ARN。
- `sessionStatus` — 核准工作階段的結果 (`APPROVED` 或 `FAILED`)。

核准的請求

下列範例顯示 MPA 團隊核准之ImportKey請求的 CloudTrail 事件：


```
{
    "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"
    }
}
```


失敗的請求

下列範例顯示 ImportKey請求的 CloudTrail 事件遭到拒絕或逾時：


```
{
    "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"
    }
}
```

若要進一步了解 AWS CloudTrail，請參閱AWS CloudTrail 《 使用者指南》 。

檢查請求狀態和處理失敗

您可以呼叫 來檢查待定 MPA 請求的狀態GetKey。回應包含具有目前核准工作階段詳細資訊MpaStatus的欄位。若要使用此命令，請以您自己的資訊取代範例命令中的########。

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

當請求待核准時，回應會顯示KeyState為 CREATE_IN_PROGRESS和 MpaStatus.Status 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"
        }
    }
}
```

一旦必要的核准者數目核准請求，KeyState會移至 CREATE_COMPLETE並MpaStatus.Status移至 APPROVED。金鑰現在可供使用：

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

如果核准團隊拒絕請求，或在達到核准閾值之前工作階段過期，則 KeyState的變更CREATE_FAILED和 MpaStatus.Status 變更為 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"
        }
    }
}
```

狀態的金鑰CREATE_FAILED無法用於密碼編譯操作。若要重試匯入，您必須提交新的ImportKey請求，這會建立新的核准工作階段。

對 AWS 付款密碼編譯身分和存取權進行故障診斷

主題將新增至本節，因為已識別 AWS 付款密碼編譯特有的 IAM 相關問題。如需 IAM 主題的一般疑難排解內容，請參閱《IAM 使用者指南》中的疑難排解一節。

監控 AWS 付款密碼編譯

監控是維護 AWS 付款密碼編譯和其他 AWS 解決方案的可靠性、可用性和效能的重要部分。AWS 提供下列監控工具來監看 AWS 付款密碼編譯、報告錯誤，並在適當時自動採取動作：

- Amazon CloudWatch AWS 會即時監控您的 AWS 資源和您在 上執行的應用程式。您可以收集和追蹤指標、建立自訂儀板表，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以讓 CloudWatch 追蹤特定 APIs的用量，或在接近 AWS 付款密碼編譯配額時通知您。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- Amazon CloudWatch Logs 可讓您監控、存放和存取來自 Amazon EC2 執行個體、CloudTrail 及其他來源的日誌檔案。CloudWatch Logs 可監控日誌檔案中的資訊，並在達到特定閾值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南。
- AWS CloudTrail 會擷取由您的帳戶或代表 AWS 您的帳戶發出的 API 呼叫和相關事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、呼叫的端點、使用的 資源 （ 金鑰 ）、呼叫的來源 IP 地址，以及呼叫的時間。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》。

主題

- 使用 記錄 AWS 付款密碼編譯 API 呼叫 AWS CloudTrail

使用 記錄 AWS 付款密碼編譯 API 呼叫 AWS CloudTrail

AWS 付款密碼編譯已與 服務整合 AWS CloudTrail，此服務提供使用者、角色或 AWS 服務在 AWS 付款密碼編譯中採取之動作的記錄。CloudTrail 會將 AWS 付款密碼編譯的所有 API 呼叫擷取為事件。擷取的呼叫包括從 主控台進行的呼叫，以及針對 API 操作的程式碼呼叫。如果您建立線索，則可以將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 AWS 付款密碼編譯的事件。如果您未設定線索，仍然可以在 CloudTrail 主控台的事件歷史記錄中檢視最新的管理 （ 控制平面 ） 事件。您可以使用 CloudTrail 所收集的資訊，判斷對 AWS 付款密碼編譯提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱「AWS CloudTrail 使用者指南」。

主題

- AWS CloudTrail 中的付款密碼編譯資訊

- CloudTrail 中的控制平面事件
- CloudTrail 中的資料事件
- 了解 AWS 付款密碼編譯控制平面日誌檔案項目
- 了解 AWS 付款密碼編譯資料平面日誌檔案項目

AWS CloudTrail 中的付款密碼編譯資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當活動在 AWS 付款密碼編譯中發生時，該活動會與事件歷史記錄中的其他服務 AWS 事件一起記錄在 CloudTrail 事件中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷史記錄檢視事件。

若要持續記錄您 AWS 帳戶中的事件，包括 AWS 付款密碼編譯的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案傳送到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌檔案
- 從多個帳戶接收 CloudTrail 日誌檔案

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 是否使用根或 AWS Identity and Access Management (IAM) 使用者登入資料提出請求。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

CloudTrail 中的控制平面事件

CloudTrail 會記錄 AWS 付款密碼編譯操作，例如 CreateKey、ImportKey、DeleteKey、ListKeys、TagResource 和所有其他控制平面操作。

CloudTrail 中的資料事件

資料事件提供有關在資源上執行或在資源中執行的資源操作的資訊，例如加密承載或翻譯接腳。資料事件是 CloudTrail 預設不會記錄的大量活動。您可以使用 CloudTrail API 或主控台，為 AWS 付款密碼編譯資料平面事件啟用資料事件 APIs 動作記錄。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的記錄資料事件。

使用 CloudTrail，您必須使用進階事件選取器來決定記錄和記錄哪些 AWS 付款密碼編譯 API 活動。若要記錄 AWS 付款密碼編譯資料平面事件，您必須包含 資源類型 `AWS Payment Cryptography key`和 `AWS Payment Cryptography alias`。設定此選項後，您可以藉由選取要記錄的特定資料事件 (例如使用 `eventName` 篩選條件追蹤 `EncryptData` 事件)，進一步調整記錄偏好設定。如需詳細資訊，請參閱 AWS CloudTrail API 參考中的 `AdvancedEventSelector`。

> **Note**
>
> 若要訂閱 AWS 付款密碼編譯資料事件，您必須使用進階事件選取器。我們建議您訂閱金鑰和別名事件，以確保您收到所有事件。

AWS 付款密碼編譯資料事件：

- `DecryptData`
- `EncryptData`
- `GenerateCardValidationData`
- `GenerateMac`
- `GeneratePinData`
- `ReEncryptData`
- `TranslatePinData`
- `VerifyAuthRequestCryptogram`
- `VerifyCardValidationData`
- `VerifyMac`
- `VerifyPinData`

資料事件需支付額外的費用。如需詳細資訊，請參閱 AWS CloudTrail 定價。

了解 AWS 付款密碼編譯控制平面日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

下列範例顯示示範 AWS 付款密碼編譯`CreateKey`動作的 CloudTrail 日誌項目。

```
      {
    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={
    }
  }
}
```

下列範例顯示 CloudTrail 日誌項目，示範啟用多區域金鑰複寫的 AWS 付款密碼編譯。

```
{
    "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"
}
```

了解 AWS 付款密碼編譯資料平面日誌檔案項目

資料平面事件可以選擇性地設定和運作，類似於控制平面日誌，但通常磁碟區較高。由於某些輸入和輸出對 AWS 付款密碼編譯資料平面操作的敏感性質，您可能會找到具有「*** 敏感資料修訂 ***」訊息的特定欄位。這無法設定，旨在防止敏感資料出現在日誌或追蹤中。

下列範例顯示示範 AWS 付款密碼編譯`EncryptData`動作的 CloudTrail 日誌項目。

```
      {
        "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": {
```

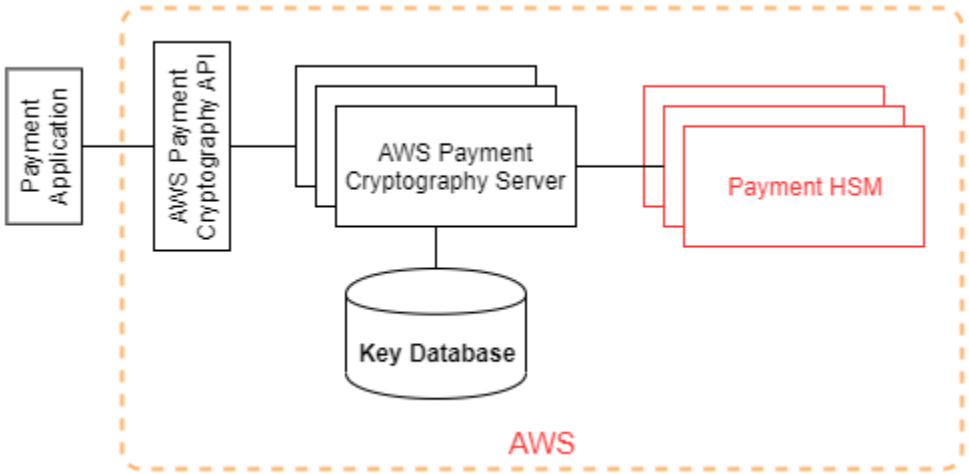
```
                            "type": "Role",
                            "principalId": "TESTXECZ5U9M4LGF2N6Y5",
                            "arn": "arn:aws:iam::111122223333:role/Admin",
                            "accountId": "111122223333",
                            "userName": "Admin"
                        },
                        "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"
                }
            }
        ]
    }
```

密碼編譯詳細資訊

AWS 付款密碼編譯提供 Web 界面來產生和管理付款交易的密碼編譯金鑰。 AWS 付款密碼編譯提供標準金鑰管理服務和付款交易密碼編譯，以及可用於集中管理和稽核的工具。本文件提供您可以在 AWS 付款密碼編譯中使用的密碼編譯操作的詳細說明，協助您評估 服務所提供的功能。

AWS 付款密碼包含多個界面 （包括透過 AWS CLI、AWS 開發套件和 的 RESTful API AWS 管理主控台)，以請求 PCI PTS HSM 驗證硬體安全模組的分散式機群的密碼編譯操作。



AWS 付款密碼編譯是一種分層服務，由面向 Web 的 AWS 付款密碼編譯主機和 HSMs層組成。這些分層主機的分組會形成 AWS 付款密碼編譯堆疊。所有對 AWS 付款密碼編譯的請求都必須透過 Transport Layer Security 通訊協定 (TLS) 提出，並在 AWS 付款密碼編譯主機上終止。服務主機只允許具有密碼套件的 TLS 提供完美的轉送私密性。此服務會使用所有其他 AWS API 操作可用的相同 IAM 登入資料和政策機制來驗證和授權您的請求。

AWS 付款密碼編譯伺服器透過私有、非虛擬網路連線至基礎 HSM。服務元件和 HSM 之間的連線會使用交互 TLS (mTLS) 進行身分驗證和加密。

主題

- 設計目標
- 基礎
- 內部操作
- 客戶操作

設計目標

AWS 付款密碼編譯旨在滿足下列要求：

- 值得信任 — 金鑰的使用受到您定義和管理的存取控制政策的保護。沒有機制可匯出純文字 AWS 付款密碼編譯金鑰。密碼編譯金鑰的機密性至關重要。若要對 HSM 執行管理動作，需要有多名 Amazon 員工擁有對以仲裁為基礎之存取控制項的角色特定存取權。任何 Amazon 員工都無法存取 HSM 主要（或主要）金鑰或備份。主金鑰無法與不屬於 AWS 付款密碼編譯區域的 HSMs 同步。所有其他金鑰都受到 HSM 主金鑰的保護。因此，客戶 AWS 付款密碼編譯金鑰在客戶帳戶中操作的 AWS 付款密碼編譯服務之外無法使用。
- 低延遲和高輸送量： AWS 付款密碼編譯在延遲和輸送量層級提供密碼編譯操作，適用於管理付款密碼編譯金鑰和處理付款交易。
- 耐久性 — 密碼編譯金鑰的耐久性旨在與 AWS 中最高耐久性服務相同。單一密碼編譯金鑰可與付款終端機、EMV 晶片卡或其他安全密碼編譯裝置 (SCD) 共用，這些裝置已使用多年。
- 獨立區域 — AWS 為需要限制不同區域中的資料存取或需要遵守資料駐留要求的客戶提供獨立區域。金鑰用量可在 AWS 區域內隔離。
- 隨機數字的安全來源 — 由於強式密碼編譯取決於真正不可預測的隨機數字產生， AWS 因此付款密碼編譯提供高品質且經過驗證的隨機數字來源。 AWS 付款密碼編譯的所有金鑰產生都使用 PCI PTS HSM 列出的 HSM，以 PCI 模式操作。
- Audit： AWS Payment Cryptography 會在透過 Amazon CloudWatch 提供的 CloudTrail 日誌和服務日誌中記錄密碼編譯金鑰的使用和管理。您可以使用 CloudTrail 日誌來檢查密碼編譯金鑰的使用，包括由您已共用金鑰的帳戶使用金鑰。 AWS 第三方評估人員會根據適用的 PCI、卡片品牌和區域付款安全標準稽核付款密碼編譯。AWS Artifact 提供證明和共同責任指南。
- Elastic： AWS Payment Cryptography 會根據您的需求向外擴展和向內擴展。 AWS 付款密碼編譯提供隨需付款密碼編譯，而不是預測和保留 HSM 容量。 AWS Payment Cryptography 負責維護 HSM 的安全性和合規性，以提供足夠的容量來滿足客戶的尖峰需求。

基礎

本章中的主題說明 AWS 付款密碼編譯的基本概念及其使用位置。他們也介紹 服務的基本元素。

主題

- 密碼編譯基本元素
- 熵和隨機數字產生
- 對稱金鑰操作

- 非對稱金鑰操作
- 金鑰儲存
- 使用對稱金鑰匯入金鑰
- 使用非對稱金鑰匯入金鑰
- 金鑰匯出
- 每個交易衍生的唯一金鑰 (DUKPT) 通訊協定
- 金鑰階層

密碼編譯基本元素

AWS 付款密碼編譯使用可參數的標準密碼編譯演算法，讓應用程式可以實作其使用案例所需的演算法。一組密碼編譯演算法由 PCI、ANSI X9、EMVco 和 ISO 標準定義。所有密碼編譯都是由以 PCI 模式執行的 PCI PTS HSM 標準清單 HSMs 執行。

熵和隨機數字產生

AWS 付款密碼編譯金鑰產生是在 AWS 付款密碼編譯 HSMs 上執行。HSMs 實作隨機數字產生器，符合所有支援金鑰類型和參數的 PCI PTS HSM 需求。

對稱金鑰操作

支援 ANSI X9 TR 31、ANSI X9.24 和 PCI PIN Annex C 中定義的對稱金鑰演算法和金鑰強度：

- 雜湊函數 — 來自輸出大小大於 2551 的 SHA2 和 SHA3 系列的演算法。除了與預先 PCI PTS POI v3 終端機的回溯相容性之外。
- 加密和解密 — 金鑰大小大於或等於 128 位元的 AES，或金鑰大小大於或等於 112 位元的 TDEA (2 個金鑰或 3 個金鑰）。
- 使用 AES 的訊息驗證碼 (MACs) CMAC 或 GMAC，以及具有已核准雜湊函數且金鑰大小大於或等於 128 的 HMAC。

AWS 付款密碼編譯針對 HSM 主金鑰、資料保護金鑰和 TLS 工作階段金鑰使用 AES 256。

注意：某些列出的函數會在內部使用，以支援標準通訊協定和資料結構。如需特定動作支援的演算法，請參閱 API 文件。

非對稱金鑰操作

支援 ANSI X9 TR 31、ANSI X9.24 和 PCI PIN Annex C 中定義的非對稱金鑰演算法和金鑰強度：

- 核准的金鑰建立機制 — 如 NIST SP800-56A (ECC/FCC2 型金鑰協議）、NIST SP800-56B (IFC 型金鑰協議） 和 NIST SP800-38F (AES 型金鑰加密/包裝） 所述。

AWS 付款密碼編譯主機僅允許使用 TLS 搭配提供完美轉送私密的密碼套件來連線至服務。

注意：某些列出的函數會在內部使用，以支援標準通訊協定和資料結構。如需特定動作支援的演算法，請參閱 API 文件。

金鑰儲存

AWS 付款密碼編譯金鑰受 HSM AES 256 主金鑰保護，並存放在加密資料庫中的 ANSI X9 TR 31 金鑰區塊中。資料庫會複寫至 AWS 付款密碼編譯伺服器上的記憶體內資料庫。

根據 PCI PIN 安全規範附件 C，AES 256 金鑰與下列相同或更強：

- 3 金鑰 TDEA
- RSA 15360 位元
- ECC 512 位元
- DSA、DH 和 MQV 15360/512

使用對稱金鑰匯入金鑰

AWS 付款密碼編譯支援匯入具有對稱金鑰或公有金鑰的加密法和金鑰區塊，其對稱金鑰加密金鑰 (KEK) 與受保護的金鑰一樣強或強。

使用非對稱金鑰匯入金鑰

AWS 付款密碼編譯支援匯入具有對稱金鑰或公有金鑰的加密法和金鑰區塊，該金鑰受到私有金鑰加密金鑰 (KEK) 的保護，該金鑰對於匯入的保護金鑰同樣強大或更強大。提供用於解密的公有金鑰必須具有其真實性和完整性，並由客戶信任的 授權單位提供的憑證所確保。

AWS Payment Cryptography 提供的公有 KEK 具有憑證授權機構 (CA) 的身分驗證和完整性保護，並證明符合 PCI PIN Security 和 PCI P2PE Annex A。

金鑰匯出

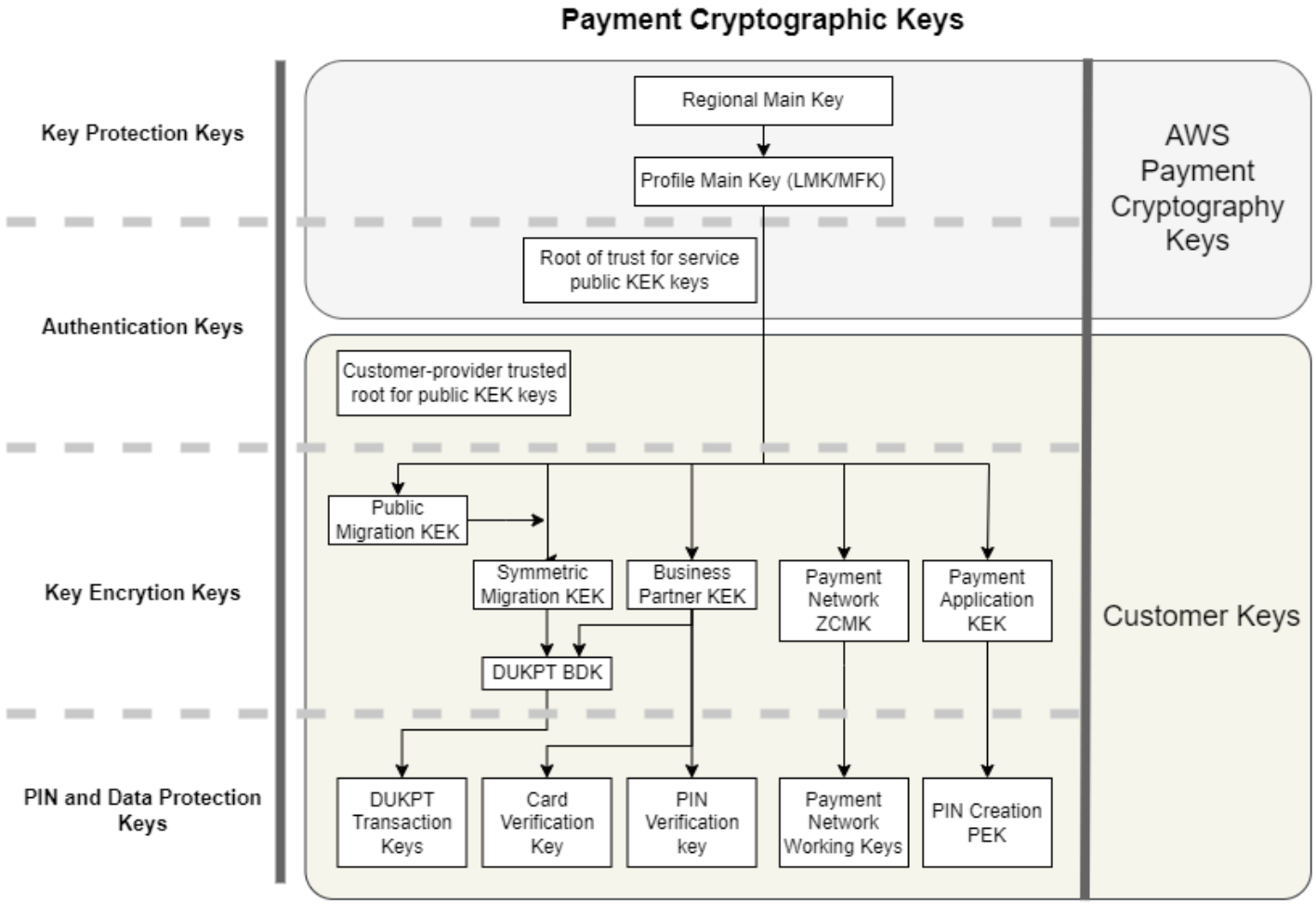
您可以使用適當的 KeyUsage 來匯出和保護金鑰，且其強度與要匯出的金鑰相同或更強。

每個交易衍生的唯一金鑰 (DUKPT) 通訊協定

AWS 付款密碼編譯支援 TDEA 和 AES 基礎衍生金鑰 (BDK)，如 ANSI X9.24-3 所述。

金鑰階層

AWS Payment Cryptography 金鑰階層可確保金鑰一律受到與保護的金鑰相同或更強的金鑰保護。



AWS 付款密碼編譯金鑰用於服務內的金鑰保護：

金鑰	說明
區域主索引鍵	保護用於密碼編譯處理的虛擬 HSM 映像或設定檔。此金鑰僅存在於 HSM 和安全備份中。
設定檔主索引鍵	最上層客戶金鑰保護金鑰，傳統上稱為本機主金鑰 (LMK) 或客戶金鑰的主檔案金鑰 (MFK)。此金鑰僅存在於 HSM 和安全備份中。設定檔會根據付款使用案例的安全標準，定義不同的 HSM 組態。
AWS 付款密碼編譯公有金鑰加密金鑰 (KEK) 金鑰的信任根	受信任的根公有金鑰和憑證，用於驗證和驗證 AWS Payment Cryptography 提供的公有金鑰，用於使用非對稱金鑰匯入和匯出金鑰。

客戶金鑰會依用於保護其他金鑰的金鑰和保護付款相關資料的金鑰分組。以下是這兩種類型的客戶金鑰範例：

金鑰	說明
客戶提供的公有 KEK 金鑰受信任根	您提供的公有金鑰和憑證作為信任根，用於驗證和驗證您使用非對稱金鑰為金鑰匯入和匯出提供的公有金鑰。
金鑰加密金鑰 (KEK)	KEK 僅用於加密其他金鑰，以便在外部金鑰存放區與 AWS 付款密碼編譯、業務合作夥伴、付款網路或組織內的不同應用程式之間進行交換。
每個交易衍生的唯一金鑰 (DUKPT) 基礎衍生金鑰 (BDK)	BDKs用於為每個付款終端機建立唯一金鑰，並將交易從多個終端機轉譯為單一收單銀行或收單機構的運作金鑰。PCI Point-to-Point加密 (P2PE) 所需的最佳實務是，不同的 BDKs用於不同的終端機模型、金鑰注入或初始化服務，或其他分段，以限制損害 BDK 的影響。
付款網路區域控制主金鑰 (ZCMK)	ZCMK 也稱為區域金鑰或區域主金鑰，由付款網路提供以建立初始工作金鑰。

金鑰	說明
DUKPT 交易金鑰	針對 DUKPT 設定的付款終端機會衍生終端機和交易的唯一金鑰。接收交易的 HSM 可以從終端機識別符和交易序號判斷金鑰。
卡片資料準備金鑰	EMV 發行者主金鑰、EMV 卡片金鑰和驗證值，以及卡片個人化資料檔案保護金鑰，可用來建立個別卡片的資料，以供卡片個人化供應商使用。發行銀行或發行者也會使用這些金鑰和密碼編譯驗證資料，做為授權交易的一部分來驗證卡片資料。
卡片資料準備金鑰	EMV 發行者主金鑰、EMV 卡片金鑰和驗證值，以及卡片個人化資料檔案保護金鑰，可用來建立個別卡片的資料，以供卡片個人化供應商使用。發行銀行或發行者也會使用這些金鑰和密碼編譯驗證資料，做為授權交易的一部分來驗證卡片資料。
付款網路工作金鑰	通常稱為發行者工作金鑰或取得者工作金鑰，這些是加密傳送至付款網路或從付款網路接收之交易的金鑰。這些金鑰經常由網路輪換，通常是每天或每小時。這些是 PIN/扣款交易的 PIN 加密金鑰 (PEK)。
個人識別號碼 (PIN) 加密金鑰 (PEK)	建立或解密 PIN 區塊的應用程式會使用 PEK 來防止儲存或傳輸純文字 PIN。

內部操作

本主題說明 服務實作的內部需求，以保護全球分散式和可擴展的付款密碼編譯和金鑰管理服務的客戶金鑰和密碼編譯操作。

主題

- HSM 保護
- 一般金鑰管理

- 客戶金鑰的管理
- 通訊安全性
- 日誌記錄和監控

HSM 保護

HSM 規格和生命週期

AWS 付款密碼編譯使用商用 HSMs機群。HSMs 經過 FIPS 140-2 第 3 級驗證，並使用 PCI 安全標準委員會核准的 PCI PTS 裝置清單中列出的韌體版本和安全政策，以符合 PCI HSM v3 規範。PCI PTS HSM 標準包含對 HSM 硬體的製造、運送、部署、管理和銷毀的額外要求，這些要求對於付款安全性和合規性至關重要，但 FIPS 140 未予以解決。

第三方評估人員會驗證 HSM 是否建立模型、韌體、組態、生命週期實體管理、變更控制、操作員存取控制、主金鑰管理，以及與 HSMs 和 HSM 操作相關的所有 PCI PIN 和 P2PE 要求。

所有 HSMs都以 PCI 模式操作，並使用 PCI PTS HSM 安全政策進行設定。只有支援 AWS 付款密碼編譯使用案例所需的函數才會啟用。 AWS 付款密碼編譯不提供列印、顯示或傳回純文字 PINs碼。

HSM 裝置實體安全性

只有具有由製造商在交付前的 AWS 付款密碼編譯憑證授權機構 (CA) 簽署的裝置金鑰的 HSMs 才能供服務使用。 AWS 付款密碼編譯是製造商 CA 的子 CA，它是 HSM 製造商和裝置憑證的信任根。製造商的 CA 已證明符合 PCI PIN Security Annex A 和 PCI P2PE Annex A。製造商會驗證所有具有由 AWS Payment Cryptography CA 簽署之裝置金鑰的 HSM 都已運送到 AWS 指定的接收者。

根據 PCI PIN Security 的要求，製造商會透過與 HSM 運送不同的通訊管道提供序號清單。在將 HSM 安裝到 AWS 資料中心的過程中，會在每個步驟檢查這些序號。最後， AWS 付款密碼編譯運算子會根據製造商清單驗證已安裝的 HSM 清單，再將序號新增至允許接收 AWS 付款密碼編譯金鑰的 HSM 清單。

HSMs隨時處於安全儲存或雙重控制狀態，其中包括：

- 從製造商運送到 AWS 機架組件設施。
- 在機架組件期間。
- 從機架組件設施到資料中心的寄件。
- 接收並安裝至資料中心安全處理室。HSM 機架使用卡片存取控制鎖定、警示門感應器和攝影機強制執行雙重控制。

- 在操作期間。
- 停用和銷毀期間。

維護和監控每個 HSM chain-of-custody，其中包含個別責任。

HSM 初始化

HSM 只有在透過序號、製造商安裝的裝置金鑰和韌體檢查總和驗證其身分和完整性之後，才會初始化為 AWS 付款密碼編譯機群的一部分。驗證 HSM 的真實性和完整性後，即會進行設定，包括啟用 PCI 模式。然後建立 AWS 付款密碼編譯區域主金鑰和設定檔主金鑰，HSM 可供服務使用。

HSM 服務和修復

HSM 具有可服務元件，不需要違反裝置的密碼編譯界限。這些元件包括冷卻風扇、電源供應器和電池。如果 HSM 機架內的 HSM 或其他裝置需要 服務，則在整個機架開啟期間都會維持雙重控制。

HSM 停用

由於 HSM end-of-life或故障而停用。HSM 會在從機架移除之前邏輯上歸零，如果正常運作，則會在 AWS 資料中心的安全處理室內銷毀。它們永遠不會傳回給製造商進行修復、用於其他用途，或是在銷毀之前從安全處理室移除。

HSM 韌體更新

HSM 韌體更新會在需要時套用，以維持與 PCI PTS HSM 和 FIPS 140-2 （或 FIPS 140-3) 列出的版本一致，如果更新與安全性相關，或者確定客戶可以從新版本中的功能中受益。 AWS 付款密碼編譯 HSMs執行off-the-shelf韌體，符合 PCI PTS HSM 列出的版本。新的韌體版本會經過 PCI 或 FIPS 認證韌體版本的完整性驗證，然後在推展到所有 HSMs之前測試功能。

運算子存取

在正常操作期間從 HSM 收集的資訊不足以識別問題或規劃變更的罕見情況下，操作員可以對 HSM 進行非主控台存取以進行故障診斷。執行下列步驟：

- 開發和核准疑難排解活動，並排定非主控台工作階段。
- HSM 會從客戶處理服務中移除。
- 主金鑰會在雙重控制下刪除。
- 允許操作員非主控台存取 HSM，以在雙重控制下執行核准的故障診斷活動。

- 終止非主控台工作階段後，會在 HSM 上執行初始佈建程序，傳回標準韌體和組態，然後同步主金鑰，然後再將 HSM 傳回給客戶。
- 工作階段的記錄會記錄在變更追蹤中。
- 從工作階段取得的資訊將用於規劃未來的變更。

所有非主控台存取記錄都會審查程序合規性，以及 HSM 監控、non-console-access管理程序或操作員訓練的潛在變更。

一般金鑰管理

區域中的所有 HSM 都會同步至區域主金鑰。區域主金鑰會保護至少一個設定檔主金鑰。Profile 主金鑰可保護客戶金鑰。

所有主金鑰都是由 HSM 產生，並使用非對稱技術透過對稱金鑰分佈分佈到 ，並與 ANSI X9 TR 34 和 PCI PIN Annex A 保持一致。

產生

AES 256 位元主金鑰是使用 PCI PTS HSM 隨機數字產生器，在為服務 HSM 機群佈建的其中一個 HSM 上產生。

區域主金鑰同步

HSM 區域主金鑰由跨區域機群的 服務與 ANSI X9 TR-34 定義的機制同步，其中包括：

- 使用金鑰分佈主機 (KDH) 和金鑰接收裝置 (KRD) 金鑰和憑證的相互身分驗證，以提供公有金鑰的身分驗證和完整性。
- 憑證由符合 PCI PIN Annex A2 要求的憑證授權機構 (CA) 簽署，但適用於保護 AES 256 位元金鑰的非對稱演算法和金鑰強度除外。
- 分散式對稱金鑰的識別和金鑰保護與 ANSI X9 TR-34 和 PCI PIN Annex A1 一致，但適用於保護 AES 256 位元金鑰的非對稱演算法和金鑰強度除外。

區域主金鑰是為 HSMs 已透過下列方式為區域進行驗證和佈建：

- 主要金鑰會在 區域中的 HSM 上產生。該 HSM 被指定為金鑰分佈主機。
- 區域中所有佈建HSMs 都會產生 KRD 身分驗證字符，其中包含 HSM 的公有金鑰和無法重播的身分驗證資訊。

- KDH 驗證 HSM 接收金鑰的身分和許可後，KRD 字符會新增至 KDH 允許清單。
- KDH 為每個 HSM 產生可驗證的主金鑰字符。權杖包含 KDH 身分驗證資訊和加密的主金鑰，只能在其建立的 HSM 上載入。
- 每個 HSM 都會收到為其建置的主要金鑰字符。驗證 HSM 自己的身分驗證資訊和 KDH 身分驗證資訊之後，主金鑰會由 KRD 私有金鑰解密，並載入主金鑰。

如果單一 HSM 必須與區域重新同步：

- 它使用韌體和組態重新驗證和佈建。
- 如果該區域是新的：
  - HSM 會產生 KRD 身分驗證字符。
  - KDH 會將權杖新增至其允許清單。
  - KDH 會產生 HSM 的主要金鑰字符。
  - HSM 會載入主索引鍵。
  - HSM 可供 服務使用。

這可確保：

- 只有針對區域內的 AWS 付款密碼編譯處理進行驗證的 HSM 才能接收該區域的主金鑰。
- 只有來自 AWS 付款密碼編譯 HSM 的主金鑰才能分發到機群中的 HSM。

區域主金鑰輪換

區域主金鑰會在加密期間到期時輪換、在不太可能發生可疑金鑰洩露的情況下輪換，或在決定影響金鑰安全性的服務變更之後輪換。

新的區域主金鑰會如同初始佈建一樣產生和分發。儲存的設定檔主索引鍵必須翻譯為新的區域主索引鍵。

區域主金鑰輪換不會影響客戶處理。

設定檔主金鑰同步

設定檔主索引鍵受到區域主索引鍵的保護。這會將設定檔限制為特定區域。

設定檔主索引鍵會相應地佈建：

- 在同步區域主金鑰的 HSM 上產生設定檔主金鑰。
- 設定檔主金鑰會使用設定檔組態和其他內容來儲存和加密。
- 描述檔用於具有區域主索引鍵之區域中任何 HSM 的客戶密碼編譯函數。

設定檔主索引鍵輪換

設定檔主金鑰會在加密期間到期時、可疑金鑰遭到入侵後，或經判定會影響金鑰安全性的服務變更後輪換。

輪換步驟：

- 產生新的設定檔主金鑰，並以待定主金鑰的形式分發，就像初始佈建一樣。
- 背景程序會將客戶金鑰材料從已建立的設定檔主金鑰轉譯為待定的主金鑰。
- 當所有客戶金鑰都已使用待定金鑰加密時，待定金鑰會提升為設定檔主金鑰。
- 背景程序會刪除受過期金鑰保護的客戶金鑰材料。

設定檔主要金鑰輪換不會影響客戶處理。

保護

金鑰僅依賴金鑰階層進行保護。保護主金鑰對於防止遺失或危及所有客戶金鑰至關重要。

區域主金鑰只能從備份還原至 HSM 驗證並佈建給服務。這些金鑰只能存放為來自特定 HSM 之特定 KDH 的可交互驗證加密主金鑰字符。

設定檔主金鑰會以設定檔組態和區域加密的內容資訊存放。

客戶金鑰存放在金鑰區塊中，受到設定檔主金鑰的保護。

所有金鑰僅存在於 HSM 中，或由其他相同或更強密碼編譯強度金鑰所保護的儲存。

耐久性

即使在通常會導致中斷的極端情況下，也必須提供交易密碼編譯和商業函數的客戶金鑰。 AWS 付款密碼編譯會跨可用區域和 AWS 區域使用多層級備援模型。客戶在付款密碼編譯操作方面需要比 服務更高的可用性和耐用性，因此應實作多區域架構。

HSM 身分驗證和主金鑰字符已儲存，可用於還原主金鑰或與新的主金鑰同步，以防必須重設 HSM。權杖會封存，並在需要時僅在雙重控制下使用。

操作員存取 HSM 主金鑰

主金鑰僅存在於由 服務管理的 HSM 中，並在安全的 AWS 設施中受到保護。主金鑰無法從任何 HSM 匯出或同步到製造商未初始化供服務使用的 HSM。AWS 運算子無法以任何形式取得主索引鍵，這些索引鍵可以載入非由 服務管理的 HSM。

客戶金鑰的管理

AWS客戶信任是我們的首要任務。您可以完全控制您匯入的金鑰，或在 AWS 帳戶下的 服務中建立的金鑰。您仍需負責設定對金鑰的存取。

AWS Payment Cryptography 是服務提供者，可代表客戶使用 HSMs 和管理金鑰，類似於長期的付款服務提供者。此服務對 HSM 實體和邏輯安全性負有完整責任。金鑰管理責任由服務與客戶共同承擔，因為客戶必須提供由 建立或匯入服務之金鑰的正確資訊，該服務會使用這些資訊來強制執行正確的金鑰使用和管理。AWS 資料隔離保護可用來確保洩露屬於某個 AWS 帳戶的金鑰，不會洩露屬於另一個帳戶的金鑰。

AWS Payment Cryptography 完全負責服務所管理金鑰的 HSM 實體合規和金鑰管理。這需要擁有和管理 HSM 主金鑰，以及保護由 AWS 付款密碼編譯管理的客戶金鑰。

客戶金鑰空間分隔

AWS 付款密碼編譯會針對所有金鑰使用強制執行金鑰政策，包括將主體限制為擁有金鑰的帳戶，除非金鑰明確與另一個帳戶共用。

AWS 帳戶可在客戶或應用程式之間提供完整的環境隔離，類似於不同資料中心的非雲端實作。每個帳戶都提供隔離的存取控制、聯網、運算資源、資料儲存、用於資料保護和付款交易的密碼編譯金鑰，以及所有 AWS 資源。Organizations 和 Control Tower 等 AWS 服務可啟用個別應用程式帳戶的企業管理，類似於企業資料中心內的籠子或房間。

運算子存取客戶金鑰

由服務管理的客戶金鑰受到分割區主金鑰的保護，只能由擁有客戶帳戶或擁有者專門為金鑰共用設定的帳戶使用。AWS 運算子無法使用 AWS 手動運算子存取機制管理的服務手動存取，以客戶金鑰匯出或執行金鑰管理或密碼編譯操作。

實作客戶金鑰管理和使用的服務程式碼，受限於 AWS PCI DSS 評估所評估的 AWS 安全程式碼實務。

備份與復原

區域 服務內部存放的金鑰和金鑰資訊會備份至 加密的封存 AWS。封存需要 的雙重控制 AWS 才能還原。

金鑰區塊

所有金鑰都存放在 ANSI X9.143 格式的金鑰區塊中並進行處理。

金鑰可以從密碼編譯或其他 ImportKey 支援的金鑰區塊格式匯入服務。同樣地，如果金鑰可匯出，則可以匯出至金鑰匯出設定檔支援的其他金鑰區塊格式或密碼編譯。

金鑰使用

金鑰使用僅限於 服務設定的 KeyUsage。對於請求的密碼編譯操作，服務將失敗任何具有不當金鑰使用方式、使用模式或演算法的請求。

金鑰交換關係

PCI PIN Security 和 PCI P2PE 要求共用加密 PINs或卡資料的金鑰的組織，包括用於共用這些金鑰的金鑰交換金鑰 (KEK)，不要與任何其他組織共用相同的金鑰。最佳實務是對稱金鑰只會在 2 個方之間共用，用於單一用途，包括在同一個組織內。這可將強制取代受影響金鑰的可疑金鑰入侵的影響降至最低。

即使商業案例需要在超過 2 個方之間共用金鑰， 仍應將方數量保持在最低數量。

AWS 付款密碼編譯提供金鑰標籤，可用於追蹤和強制執行這些要求中的金鑰用量。

例如，透過為與該服務提供者共用的所有金鑰設定「KIF」=「POSStation」，即可識別不同金鑰注入設施的 KEK 和 BDK。另一個範例是使用「網路」=「PayCard」標記與付款網路共用的金鑰。標記可讓您建立存取控制並建立稽核報告，以強制執行和示範您的金鑰管理實務。

刪除金鑰

DeleteKey 會標記資料庫中的金鑰，以便在客戶可設定的期間之後刪除。在此期間之後，金鑰會無法復原地刪除。這是防止意外或惡意刪除金鑰的安全機制。標記為刪除的金鑰不適用於 RestoreKey 以外的任何動作。

刪除的金鑰會在刪除後保留在服務備份中 7 天。在此期間，它們無法還原。

屬於已關閉 AWS 帳戶的金鑰會標記為刪除。如果在達到刪除期間之前重新啟用帳戶，則會還原任何標記為刪除的金鑰，但會停用。您必須重新啟用它們，才能將其用於密碼編譯操作。

通訊安全性

外部

AWS 付款密碼編譯 API 端點符合 AWS 安全標準，包括 1.2 版或更新版本的 TLS，以及用於請求身分驗證和完整性的 Signature 第 4 版。

傳入 TLS 連線會在網路負載平衡器上終止，並透過內部 TLS 連線轉送至 API 處理常式。

內部 (Internal)

服務元件之間以及服務元件與其他 AWS 服務之間的內部通訊，受到 TLS 使用強式密碼編譯的保護。

HSM 位於只能從服務元件連線的私有、非虛擬網路上。HSM 和服務元件之間的所有連線都使用交互 TLS (mTLS) 保護，在 TLS 1.2 或更高版本。TLS 和 mTLS 的內部憑證是由 Amazon Certificate Manager 使用 AWS Private Certificate Authority 管理。監控內部 VPCs和 HSM 網路是否有非預期的活動和組態變更。

日誌記錄和監控

內部服務日誌包括：

- 服務進行之 AWS 服務呼叫的 CloudTrail 日誌
- 這兩個事件的 CloudWatch 日誌會直接記錄到 CloudWatch 日誌或來自 HSM 的事件
- 來自 HSM 和服務系統的日誌檔案
- 日誌封存

所有日誌來源都會監控和篩選敏感資訊，包括 金鑰。系統會對日誌進行系統性檢閱，以確保它們不包含敏感的客戶資訊。

只有完成任務角色所需的個人才能存取日誌。

所有日誌都會與 AWS 日誌保留政策保持一致。

客戶操作

AWS 根據 PCI 標準，付款密碼編譯對 HSM 實體合規負有完整責任。此服務也提供安全的金鑰存放區，並確保金鑰只能用於 PCI 標準允許的目的，並在建立或匯入期間由您指定。您負責設定金鑰屬性和存取權，以利用服務的安全性和合規功能。

主題

- 產生金鑰
- 匯入金鑰
- 匯出金鑰
- 刪除 金鑰
- 輪換 金鑰

產生金鑰

建立金鑰時，您可以設定服務用來強制合規使用金鑰的屬性：

- 演算法和金鑰長度
- Usage
- 可用性和過期

用於屬性型存取控制 (ABAC) 的標籤也應該在建立期間設定，以限制與特定合作夥伴或應用程式搭配使用的金鑰。請務必包含政策，以限制允許刪除或變更標籤的角色。

您應該確保決定可能使用和管理金鑰之角色的政策在建立金鑰之前已設定。

> **Note**
>
> CreateKey 命令上的 IAM 政策可用於強制執行和示範金鑰產生的雙重控制。

匯入金鑰

匯入金鑰時，服務會使用金鑰區塊中的密碼編譯繫結資訊來設定強制合規使用金鑰的屬性。設定基本金鑰內容的機制是使用以來源 HSM 建立且受到共用或非對稱 KEK 保護的金鑰區塊。這符合 PCI PIN 要求，並保留來源應用程式的用量、演算法和金鑰強度。

除了金鑰區塊中的資訊之外，匯入時還必須建立重要的金鑰屬性、標籤和存取控制政策。

使用密碼編譯匯入金鑰不會從來源應用程式傳輸金鑰屬性。您必須使用此機制適當地設定屬性。

金鑰通常會使用純文字元件交換，由金鑰託管人傳輸，然後載入在安全房間實作雙重控制的儀式。AWS 付款密碼編譯不支援此項目。API 將匯出具有憑證的公有金鑰，該憑證可由您自己的 HSM 匯入，以匯出可由服務匯入的金鑰區塊。可讓您使用自己的 HSM 載入純文字元件。

您應該使用金鑰檢查值 (KCV) 來驗證匯入的金鑰是否符合來源金鑰。

ImportKey API 上的 IAM 政策可用於強制執行和示範金鑰匯入的雙重控制。

匯出金鑰

與合作夥伴或內部部署應用程式共用金鑰可能需要匯出金鑰。使用金鑰區塊進行匯出，可維護具有加密金鑰材料的基本金鑰內容。

金鑰標籤可用來限制將金鑰匯出至共用相同標籤和值的 KEK。

AWS 付款密碼編譯不提供或顯示純文字金鑰元件。這需要金鑰託管人直接存取 PCI PTS HSM 或 ISO 13491 測試的安全密碼編譯裝置 (SCD) 以進行顯示或列印。您可以使用 SCD 建立非對稱 KEK 或對稱 KEK，以在雙控下進行純文字金鑰元件建立程序。

金鑰檢查值 (KCV) 應該用來驗證目的地 HSM 比對來源金鑰是否匯入。

刪除 金鑰

您可以使用刪除金鑰 API，在您設定的一段時間後排定要刪除的金鑰。在那之前，時間索引鍵是可復原的。金鑰一旦刪除，就會從服務中永久移除。

DeleteKey API 上的 IAM 政策可用於強制執行和示範金鑰刪除的雙重控制。

輪換 金鑰

可以使用金鑰別名來實作金鑰輪換的效果，方法是建立或匯入新金鑰，然後修改金鑰別名以參考新金鑰。根據您的管理實務，舊金鑰將被刪除或停用。

的配額 AWS Payment Cryptography

對於每個 AWS 服務，您的 AWS 帳戶有預設配額，先前稱為限制。除非另有說明，否則每個配額都是區域特定的。您可以請求提高某些配額，而其他配額無法提高。

名稱	預設	可調整	說明
別名	每個受支援的區域：2,000	是	您可以在目前區域中此帳戶中擁有的別名數目上限。
控制平面請求的合併速率	每個支援的區域：每秒 5 個	是	您可以在目前區域中的此帳戶中每秒提出的控制平面請求數目上限。此配額適用於所有合併的控制平面操作。
資料平面請求的合併速率（非對稱）	每個受支援的區域：每秒 20 個	是	對於資料平面操作，您可以在目前區域中的此帳戶中使用非對稱金鑰提出的每秒請求數上限。此配額適用於所有合併的資料平面操作。
資料平面請求的合併速率（對稱）	每個支援的區域：每秒 500	是	目前區域中，您在此帳戶中可以使用對稱金鑰進行的資料平面操作每秒請求數上限。此配額適用於所有合併的資料平面操作。
金鑰	每個受支援的區域：2,000	是	您可以在目前區域中此帳戶中擁有的金鑰數目上限，不包括已刪除的金鑰。

AWS 付款密碼編譯使用者指南的文件歷史記錄

下表說明 AWS 付款密碼編譯的文件版本。

變更	描述	日期
多區域金鑰複寫的非對稱金鑰支援	多區域金鑰複寫現在除了對稱金鑰之外，還支援非對稱付款密碼編譯金鑰，例如 RSA 和 ECC。	2026 年 7 月 27 日
新功能 - AS2805	支援演算法和流程以支援 AS2805 區域支援	2025 年 12 月 17 日
新功能 - 多區域金鑰複寫	使用多區域金鑰複寫，您可以將 AWS 付款密碼編譯金鑰複寫到多個 AWS 區域。	2025 年 9 月 10 日
新功能 - ECDH	在此版本中，ECDH 可用來建立共用 KEK 以進行進一步的金鑰交換。	2025 年 3 月 30 日
新的金鑰交換指引	為金鑰交換提供的新指引。也新增了常見 JCB 命令的資訊。	2025 年 1 月 31 日
新區域啟動	新增在歐洲（法蘭克福）、歐洲（愛爾蘭）、亞太區域（新加坡）和亞太區域（東京）推出新區域的端點	2024 年 7 月 31 日
適用於資料平面和動態金鑰的 CloudTrail	新增有關將 CloudTrail 用於資料平面（加密）操作的資訊，包括範例。還新增了有關將動態金鑰用於某些函數的資訊，以更好地支援不應匯入 AWS 付款密碼編譯的一次性或有限使用金鑰	2024 年 7 月 10 日

更新的範例	新增發卡的新範例	2024 年 7 月 1 日
功能版本	新增 VPC 端點 (PrivateLink) 和 iCVV 範例的相關資訊。	2024 年 5 月 30 日
功能版本	有關使用 RSA 和匯出 DUKPT IPEK/IK 金鑰之金鑰匯入/匯出的新功能新增的資訊。	2024 年 1 月 15 日
初始版本	初始版本的 AWS 付款密碼編譯使用者指南	2023 年 6 月 8 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。