# luna-kmu 使用手冊

適用版本：0.1.0

`kmu` 是 Thales Luna HSM 的金鑰管理命令列工具：產生與管理金鑰、加解密、匯出入金鑰材料，以及與 payShield TMD 協商金鑰。

每個指令都是單次執行，不保留 session。完整的參數清單隨時可由 `kmu <指令> --help` 取得。

# 目錄

1. 快速開始 ..... 4
取得執行檔 ..... 4
環境需求 ..... 4
第一個指令 ..... 4
全域選項 ..... 4
結束代碼 ..... 5
2. 連線與認證 ..... 6
PIN 的四種來源 ..... 6
角色 ..... 6
驗證憑證 ..... 6
3. 查詢指令 ..... 7
列出 slot ..... 7
列出物件 ..... 7
顯示單一物件的屬性 ..... 7
顯示支援的 mechanism ..... 8
4. 產生金鑰 ..... 9
對稱金鑰 ..... 9
非對稱金鑰對 ..... 9
屬性旗標 ..... 10
明文金鑰分量 ..... 10
5. 物件管理 ..... 11
建立資料物件 ..... 11
修改屬性 ..... 11
屬性讀寫 ..... 11
摘要與 KCV ..... 12
刪除 ..... 12

KCV 的四種方法 ..... 12
6. 密碼運算 ..... 14
加解密 ..... 14
金鑰衍生 ..... 14
7. 匯出與匯入 ..... 16
公鑰 ..... 16
對稱金鑰與私鑰（以另一把金鑰包裹） ..... 16
私鑰的密碼保護（PKCS#8 PBES2） ..... 17
TR-31 key block ..... 17
明文分量匯入 ..... 17
檔案編碼轉換 ..... 17
8. 與 payShield TMD 協商金鑰 ..... 19
9. 疑難排解 ..... 20
CKR_KEY_NOT_WRAPPABLE ..... 20
CKR_OPERATION_NOT_ALLOWED（digestkey） ..... 20
KCV 與其他工具對不上 ..... 20
CKR_ECC_UNKNOWN_CURVE ..... 20
PKCS#8 檔 OpenSSL 讀不開 ..... 20
在管線或排程裡輸入分量與密碼 ..... 20
找不到 PKCS#11 library ..... 21

# 1. 快速開始

## 取得執行檔

kmu 是單一執行檔，不需要安裝，也不依賴額外的 runtime：

| 平台 | 檔名 |
|---|---|
| Linux x86-64 | kmu |
| Windows x86-64 | kmu.exe |

複製到任何目錄即可執行。放進 PATH 之後就能直接打 kmu，以下範例都假設如此；否則以完整路徑呼叫。

## 環境需求

- Luna Client 已安裝，/etc/Chrystoki.conf（Windows 是 C:\Program Files\SafeNet\LunaClient\crystoki.ini）設定完成 —— kmu 從這裡的 Chrystoki2 區段讀出 PKCS#11 library 的位置
- 設定檔不在預設位置時，用 ChrystokiConfigurationPath 指向它**所在的目錄**（指向設定檔本身或指向 .so／.dll 都會回報錯誤）
- 或以 --library 直接指定 PKCS#11 library 的路徑，這會跳過設定檔

## 第一個指令

```
# 看看有哪些 slot
kmu listslot

# 把 PIN 放進環境變數（不要寫在命令列上）
export HSM_PIN='your-pin'

# 列出 slot 0 的物件
kmu list --slot 0 --pin-env HSM_PIN
```

## 全域選項

| 選項 | 說明 |
|---|---|
| --library <路徑> | PKCS#11 library。未指定時依 ChrystokiConfigurationPath 解析 |
| --json | 以 JSON 輸出，供腳本使用 |
| -h, --help | 顯示說明（每個子指令都有） |
| -V, --version | 顯示版本與產品網址 |

```
kmu --help              # 指令總覽
kmu export --help       # 單一指令的完整參數
kmu help export         # 同上
```

說明的第一行是標題，依作業系統的語系顯示：中文環境（語言碼 zh，繁簡皆同）是「Luna KMU 命令列版 版權所有 (c) 雲杉科技」，其餘語系是英文的「Luna KMU CLI Copyright (c) Spruce Technology Co., Ltd. Taiwan.」。第二行是產品網址 https://sprucets.com/tools/luna-kmu/，`-V` 也會印出它。 Linux 依序看 `LANGUAGE`、`LC_ALL`、`LC_MESSAGES`、`LANG`；Windows 依使用者的顯示語言。其餘說明文字一律是中文。

## 結束代碼

| 碼 | 意義 |
|---|---|
| 0 | 成功 |
| 1 | 執行錯誤（物件不存在、HSM 回錯等） |
| 2 | 參數錯誤（含 PKCS#11 library 的設定錯誤） |
| 3 | 連線或登入失敗 |

## 2. 連線與認證

### PIN 的四種來源

四種擇一（同時給兩種會被當成參數錯誤），都沒給才互動輸入：

```
kmu list --slot 0 --ped                    # 1. PED 認證，不送 PIN
kmu list --slot 0 --pin-env HSM_PIN        # 2. 從環境變數讀（建議）
kmu list --slot 0 --pin-file /run/pin.txt  # 3. 從檔案讀（去除尾端換行）
kmu list --slot 0 --pin 'your-pin'         # 4. 直接給（不建議）
kmu list --slot 0                          # 5. 互動式輸入（不顯示輸入內容）
```

> **`--pin` 會留在 shell 歷史與行程清單中。** 正式環境請用 `--pin-env` 或 `--pin-file`。PIN 不會出現在任何輸出或錯誤訊息中。
>
> stdin 不是終端機時，互動輸入改從 stdin 讀一行（提示寫到 stderr）。**PIN 是空的就不送出**，直接以結束代碼 2 結束 —— 空 PIN 送到 HSM 也會算一次登入失敗。

### 角色

預設以 Crypto Officer 登入。加 `--crypto-user` 改以 Crypto User 登入。

### 驗證憑證

```
kmu login --slot 0 --pin-env HSM_PIN
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer
```

連線、登入、印出成功訊息後結束，不留下任何狀態。

```
kmu logout
```

輸出：

```
單次執行模式不持有 session：每個指令各自登入、執行完即結束，因此不需要登出。
```

`logout` 是明確的 no-op，只印出訊息就結束 —— kmu 是單次執行，不持有 session，沒有東西可登出。

---

## 3. 查詢指令

### 列出 slot

```
kmu listslot        # 亦可寫成 kmu listslots（別名）
```

輸出：

```
Slot list :
Slot  Label  Model         Firmware  Software  Serial
0     10     LunaSA 7.9.1  7.9.3     7.9.1     1482011006884
1     1      LunaSA 7.9.1  7.9.3     7.9.1     1482011006875
2     6      LunaSA 7.9.1  7.9.3     7.9.1     1482011006880
```

### 列出物件

```
kmu list --slot 0 --pin-env HSM_PIN
kmu list --slot 0 --pin-env HSM_PIN --limit 20     # 只看前 20 筆
kmu list --slot 0 --pin-env HSM_PIN --json         # 給腳本用
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

handle=1417  Class=Secret Key   Label=mykek
handle=1925  Class=Private Key  Label=myrsa
handle=1365  Class=Public Key   Label=myrsa
handle=1866  Class=Secret Key   Label=mykey

 4 objects found
```

### 顯示單一物件的屬性

```
kmu getattribute --slot 0 --pin-env HSM_PIN --handle 1866
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

The list of attributes for handle 1866 are :

Class=Secret Key
Token=true
Private=true
Modifiable=true
Label=mykey
ObjectSize=264 bytes
KeyType=aes (0x0000001F)
id=
StartDate=
EndDate=
Local=true
Derive=true
Sensitive=true
AlwaysSensitive=true
Extractable=true
```

```
NeverExtractable=false
Encrypt=true
Decrypt=true
Sign=true
Verify=true
Derive=true
…（其餘為 Luna 專屬屬性）
```

輸出涵蓋 class、key type、label、能力旗標、指紋、專屬屬性等，實際項目依物件的型別而不同。

### 顯示支援的 **mechanism**

```
kmu getcapabilities --slot 0
```

輸出：

```
Supported key generation mecanisms :
-> aes              Minimum key size : 16       Maximum key size : 32
-> sm4              Key size : 128
-> des              Key size : 64
-> des2             Key size : 128
-> des3             Key size : 192
-> genericsecret        Minimum key size : 64       Maximum key size : 4096
-> hmac             Minimum key size : 64       Maximum key size : 4096
-> rsa-pkcs         Minimum key size : 256      Maximum key size : 8192
…（其餘型別同樣格式）
```

## 4. 產生金鑰

### 對稱金鑰

```
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type aes --key-size 32 --label mykey
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Key successfully generated, handle is : 1311, label is : mykey
```

| --key-type | 允許的 --key-size（位元組） |
|---|---|
| aes、sm4 | 16、24、32 |
| des | 8 |
| des2 | 16 |
| des3 | 24 |
| genericsecret、hmac | 1–512 |

### 非對稱金鑰對

```
# RSA（--mech 可選 pkcs、prime、aux）。--key-size 是模數的「位元數」，
# 不是位元組 — 給 256 會得到 256 位元的 RSA，而且不會報錯
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type rsa --key-size 2048 --label myrsa

# 更長的 RSA
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type rsa --key-size 4096 --label myrsa40
96

# EC 家族：以 --curve 指定曲線
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type ecdsa --curve prime256v1 --label my
ec
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type ecdsa --curve secp384r1 --label myp
384
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type ecdsa --curve secp256k1 --label myk
1
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type eddsa --curve ed25519 --label myed
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type eddsa --curve ed448 --label myed448
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type montgomery --curve x25519 --label m
yx

# 後量子：以 --parameter-set 指定（也接受簡短形式 65／768，非英數字元會被忽略）
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type ml-dsa --parameter-set ml-dsa-65 --
label mypqc
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type ml-kem --parameter-set ml-kem-768 -
-label mykem

# DSA／DH 需要 domain parameters
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type dsa --label mydsa \
    --prime <p 的十六進位> --subprime <q> --base <g>
```

```
# HSS／LMS：--lms-type 與 --lmots-type 以逗號分隔，每層各一個
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type hss --label myhss \
    --hss-levels 2 --lms-type 5,5 --lmots-type 4,4
```

金鑰對會印出兩個 handle：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Key pair successfully generated, private key handle : 1405, public key handle : 639
```

支援的型別：rsa、dsa、dh、dh-x9.42、ecdsa、eddsa、montgomery、 sm2、ml-dsa、ml-kem、hss／lms。

公私鑰可用 --label-public 與 --label-private 分別命名；只給 --label 時兩者共用。

## 屬性旗標

以下旗標**未指定時一律為 true**：

--token、--private-object、--sensitive、--extractable、--modifiable、 --encrypt、--decrypt、--sign、--verify、--wrap、--unwrap、--derive

值只接受小寫的 true／false（1／0 會被當成參數錯誤）：

```
kmu generatekey ... --sign false --wrap true
```

旗標只套用到該物件有意義的屬性。以金鑰對為例：公鑰不帶 CKA_SENSITIVE、CKA_EXTRACTABLE，--sign、--decrypt、--unwrap 只作用在私鑰； --private-object 則公私鑰都套用，所以預設產生的公鑰也是 Private=true。

## 明文金鑰分量

以 N 份分量 XOR 組成金鑰。逐份輸入（不顯示輸入內容），最後以 pci 方法顯示組出來的金鑰的 KCV（sm4、genericsecret、hmac 沒有對應的 KCV 方法，不顯示）。每份分量的長度必須等於 --key-size；不能與 --json 併用：

```
kmu generatekey --slot 0 --pin-env HSM_PIN --key-type aes --key-size 32 \
    --label combined --clear-components 3
```

## 5. 物件管理

### 建立資料物件

```
kmu createdo --slot 0 --pin-env HSM_PIN --label mydata --application app --value 41424344
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Data Object successfully generated, handle value is 1902,
```

### 修改屬性

未指定的屬性不受影響。

```
kmu setattribute --slot 0 --pin-env HSM_PIN --handle 1866 --sign false --label mykey
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Attempt to update attributes on object 1866 ...
Updating attribute CKA_SIGN ... Success. Value = false

2 attributes updated
```

--label 這類文字屬性改成功時不另印一行，但會算進最後的總數。

**不是每個屬性建立後都改得動：**

- --extractable 只能從 true 改成 false，改回 true 會失敗。
- --private-object 只能在建立物件時決定（generatekey、import 等）。建立後用 setattribute 改一律失敗，改成相同的值也一樣（Luna 韌體 7.9.3 實測）。
- --token 與 --sensitive 不在 setattribute 會送出的屬性裡（與上游相同），指定了**也不會有任何效果**。
- AlwaysSensitive、NeverExtractable 由 HSM 維護，沒有任何指令能改。

每個屬性各自送出、各自回報成功或失敗，--json 的 results 逐項列出。

### 屬性讀寫

CKA_ID 與 CKA_VALUE 以二進位讀寫，其餘以文字。

```
kmu readattribute  --slot 0 --pin-env HSM_PIN --handle 1866 --attribute id --output-file
id.bin
kmu writeattribute --slot 0 --pin-env HSM_PIN --handle 1866 --attribute id --input-file i
d.bin
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Successfull: 4 bytes written in file : id.bin
```

### 摘要與 KCV

```
kmu computekcv --slot 0 --pin-env HSM_PIN --handle 1866 --method pci
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

KCV of the key 1866 is equal to : 620853
```

`digestkey` 對金鑰本身算摘要，用法相同：

```
kmu digestkey --slot 0 --pin-env HSM_PIN --handle 1866 --hash sha256
```

這個指令受 partition 政策管制，政策關著時一律失敗（見第 9 章）。

### 刪除

預設先顯示要刪的物件並要求確認；非互動環境（含 `--json`）必須加 `--yes`。

```
kmu delete --slot 0 --pin-env HSM_PIN --handle 1902
kmu delete --slot 0 --pin-env HSM_PIN --handle 1902 --yes    # 略過確認
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Object with handle 1902 deleted
```

### KCV 的四種方法

| `--method` | AES | DES 系列 | generic secret |
|---|---|---|---|
| `pkcs11` | ECB 加密全零區塊 | ECB 加密全零區塊 | — |
| `pci` | AES-CMAC（全零輸入） | 同 `pkcs11` | — |
| `gp` | AES-CMAC，輸入為全 `0x01` | 同 `pkcs11` | — |
| `hmac-sha-256` | — | — | HMAC-SHA-256 |

一律取結果的**前 3 個位元組**。DES 系列三種方法的結果相同；SM4 不支援任何方法。

> **同一把金鑰用不同方法會得到不同的值。** 與其他工具比對 KCV 時務必確認雙方用的是同一種方法。

**computekcv 會暫時改動物件。** 金鑰若缺少所需的 CKA_SIGN 或 CKA_ENCRYPT，會暫時開啟該屬性、算完還原 —— 否則算不出 KCV。中途失敗也會還原；--json 會回報動過哪個屬性。

## 6. 密碼運算

### 加解密

```
kmu encrypt --slot 0 --pin-env HSM_PIN --handle 1866 --algo aes_cbc_pad \
    --input-file plain.bin --output-file cipher.bin
kmu decrypt --slot 0 --pin-env HSM_PIN --handle 1866 --algo aes_cbc_pad \
    --input-file cipher.bin --output-file plain.bin
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Encrypt successfull: 16 bytes written in file : cipher.bin
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Decrypt successfull : 5 bytes written in file : plain2.bin
```

**演算法**（15 種）：aes_ecb、aes_cbc、aes_cbc_pad、aes_cbc_pad_ipsec、 aes_cfb8、aes_cfb128、aes_ofb、aes_kw、aes_kwp、aes_gcm、 rsa_pkcs、rsa_oaep_sha256、rsa_oaep_sha384、rsa_oaep_sha512、rsa_oaep

```
# --format 同時決定輸入與輸出：bin（原樣）或 text（大寫十六進位）
kmu encrypt ... --format text

# 需要 IV 的模式有預設值，--iv 可覆寫
kmu encrypt ... --algo aes_cbc --iv 000102030405060708090A0B0C0D0E0F

# GCM 的額外參數
kmu encrypt ... --algo aes_gcm --aad 41424344 --tag-bits 128

# rsa_oaep 的雜湊不寫在名稱裡，要用 --hash（sha1／sha256／sha384／sha512）
kmu encrypt ... --algo rsa_oaep --hash sha256
```

### 金鑰衍生

```
kmu derive --slot 0 --pin-env HSM_PIN --handle 1866 --mech sha256 \
    --key-type aes --key-size 32 --label derived
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Key successfully derived: handle is : 1271, label is : derived
```

**機制**（12 種）：sha1、sha224、sha256、sha384、sha512、 sha3-224、sha3-256、sha3-384、sha3-512、luna-kdf、luna-nist-kdf、 aes-encrypt-ecb

```
# Luna KDF 需要額外參數
kmu derive ... --mech luna-nist-kdf --kdf-type aes-cmac --kdf-scheme scheme1 \
```

```
    --kdf-counter 1 --kdf-label 41424344 --kdf-context 50515253

# 以資料加密衍生。--kdf-data 的長度要夠衍生出 --key-size 那麼長的金鑰，
# 例如衍生 32 位元組的 AES 金鑰就要給 32 位元組的資料。
kmu derive ... --mech aes-encrypt-ecb \
    --kdf-data 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
```

`--kdf-type`：tdes-cmac、aes-cmac、hmac-sha1...hmac-sha512 `--kdf-scheme`：scheme1...scheme4、scp03、hid

## 7. 匯出與匯入

公鑰

```
kmu export --slot 0 --pin-env HSM_PIN --handle 1365 --format pkcs8 --output-file pub.pem
kmu import --slot 0 --pin-env HSM_PIN --key-class public --key-type ecdsa \
    --label mypub --input-file pub.pem --format pkcs8
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Export public key : Key (handle=1365) successfully exported.
451 bytes written in file : pub.pem
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Public key successfully imported: handle is : 490, label is : mypub
```

格式：`bin`（SubjectPublicKeyInfo 的 DER）、`text`（同上的十六進位）、`pkcs8`（`-----BEGIN PUBLIC KEY-----` 的 PEM）。

支援的型別：RSA、DSA、DH／X9.42-DH、ECDSA、SM2、Ed25519、Ed448、X25519、**X448**、ML-DSA、ML-KEM、HSS。

`export` 走哪條路徑由物件的 class 決定，**沒有 `--key-class`**。

對稱金鑰與私鑰（以另一把金鑰包裹）

```
kmu export --slot 0 --pin-env HSM_PIN --handle 1866 --format bin \
    --algo aes_kwp --key 1417 --output-file key.bin
kmu import --slot 0 --pin-env HSM_PIN --key-class secret --key-type aes \
    --label mykey --input-file key.bin --format bin --algo aes_kwp --key 1417
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Export : Key (handle=1866) successfully wrapped with key (handle=1417).
40 bytes written in file : key.bin
```

**可用於包裹的演算法**（12 種）：加解密那 15 種扣掉 `aes_cfb8`、`aes_cfb128`、`aes_ofb` ——串流模式不可用於包裹。

> **私鑰匯出受 partition 政策管制**。若 `Allow private key wrapping` 為 `0`，會回 `CKR_KEY_NOT_WRAPPABLE`。以 `lunacm` 的 `partition showpolicies` 查詢。

### 私鑰的密碼保護（PKCS#8 PBES2）

```
kmu export --slot 2 --pin-env HSM_PIN --handle 1925 --format pkcs8 \
    --algo pbkdf2_aes256_cbc --prf hmac-sha1 --key-password 'secret' --output-file prv.pe
m

kmu import --slot 2 --pin-env HSM_PIN --key-class private --key-type rsa \
    --label myprv --input-file prv.pem --format pkcs8 --key-password 'secret'
```

演算法：`pbkdf2_aes128_cbc`、`pbkdf2_aes192_cbc`、`pbkdf2_aes256_cbc`。 `--salt` 與 `--iv` 未給時由 HSM 產生亂數；`--iteration` 預設 10000。

匯入時 salt／迭代次數／PRF／IV **全部從檔案讀**，不需要（也沒有）那些參數。

產出的檔案與 OpenSSL 互通：

```
openssl pkey -in prv.pem -passin pass:secret -noout -text
```

> **密碼的大小寫有意義**，`Passw0rd` 與 `passw0rd` 是兩個不同的密碼。

### TR-31 key block

```
kmu export --slot 0 --pin-env HSM_PIN --handle 1866 --format tr31 --key 1548 \
    --tr31-usage K0 --tr31-mode B --tr31-export E --output-file block.txt
kmu import --slot 0 --pin-env HSM_PIN --key-class secret --key-type aes \
    --label imported --input-file block.txt --format tr31 --key 1548
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Export : Key (handle=1866) successfully wrapped with key (handle=1548).
144 bytes written in file : block.txt
```

僅支援版本 `D`（AES Key Derivation Binding）與 AES 金鑰。`--key` 是 KBPK （Key Block Protection Key）的 handle。

> 產出 TR-31 需要金鑰明文短暫經過主機記憶體（MAC 涵蓋未加密的內容，得在加密前計算），用完立即清除。

### 明文分量匯入

```
kmu import --slot 0 --pin-env HSM_PIN --key-class secret --key-type aes \
    --key-size 32 --clear-components 3 --label combined
```

只支援 `aes`、`des`、`des2`、`des3`，分量數 2 – 16。逐份輸入（不顯示輸入內容）、逐份顯示 KCV 並要求確認。

### 檔案編碼轉換

```
kmu convert --input-file a.bin --output-file a.txt --in-form bin --out-form text
kmu convert --input-file a.txt --output-file a.bin --in-form text --out-form bin
```

輸出：

```
convert :
26 bytes written in file : a.txt
```

不連線 HSM。`--in-form` 與 `--out-form` 只接受 `bin` 與 `text`，且必須不同（`txt` 是 `text` 的別名，兩者視為相同）。

---

## 8. 與 payShield TMD 協商金鑰

```
kmu remotemzmk --slot 0 --pin-env HSM_PIN --key-type aes --key-size 32 \
    --label mzmk --input-file tmd_pub.txt
```

輸出：

```
Authentication with slot [0] : 10 ... Success -> Connected as Crypto Officer

Key successfully derived: handle is : 1601, label is : mzmk
MZMK Key check value : 62727F

Successfull: the file MZMKdata_2026_09_16_08_33.csv has been generated. You can upload in
 TMD.
```

一次 ECDH 協商，兩端各自算出同一把 MZMK：

1. 讀 TMD 匯出的 EC 公鑰（SubjectPublicKeyInfo 的十六進位文字）
2. 在同一條曲線上產生一對 session 金鑰
3. `CKM_ECDH1_DERIVE` + `CKD_SHA256_KDF`（X9.63）派生 MZMK
4. 在**輸入檔所在的目錄**產生 `MZMKdata_<年>_<月>_<日>_<時>_<分>.csv`

金鑰從未離開任何一端 —— CSV 裡只有公鑰與 KCV，都不是祕密。把 CSV 上傳到 TMD，兩邊比對 KCV 即可確認協商成功。

`--key-type` 只接受 `aes`（16／24／32 位元組）與 `des`（只能 24，會建立成 DES3）。

---

## 9. 疑難排解

**CKR_KEY_NOT_WRAPPABLE**

partition 政策不允許匯出該類金鑰。查詢：

```
printf 'slot set -slot 0\npartition showpolicies\nexit\n' | lunacm | grep "private key wrapping"
```

`Allow private key wrapping : 0` 表示該 partition 禁止私鑰匯出（同時會出現的 `Enable private key wrapping` 那行是 HSM 的能力，不是 partition 的政策）。**一次只查一個 slot** —— 把多個 slot 串在同一個 pipeline 會讓 `slot set` 失效。

**CKR_OPERATION_NOT_ALLOWED（digestkey）**

partition 政策 `Allow digest key` 為 `0`，與金鑰的 `CKA_SENSITIVE` 無關。

### KCV 與其他工具對不上

先確認雙方用同一種方法。`pkcs11` 是 ECB 加密全零區塊，`pci` 對 AES 是 AES-CMAC —— 同一把金鑰兩者的值不同，兩個都正確。

**CKR_ECC_UNKNOWN_CURVE**

曲線名稱 kmu 認得，但這台 HSM 不支援該曲線。用 `kmu getcapabilities --slot N` 看這個 slot 實際支援哪些 mechanism。（名稱 kmu 不認得時，會在送到 HSM 之前就以「不認得的曲線」回報；`prime256v1`／`secp256r1`、`secp384r1`、`secp256k1`、 `ed25519`、`ed448`、`x25519`、`x448`、`sm2` 等常見名稱都認得。）

### PKCS#8 檔 OpenSSL 讀不開

先確認讀取指令正確：**`openssl pkcs8 -in enc.pem -nocrypt` 不會解密** —— 不加 `-topk8` 時 `-nocrypt` 的意思是「輸入未加密」。用：

```
openssl pkey -in enc.pem -passin pass:yourpass -noout -text
```

若仍失敗且密碼含大寫字母，試一次全小寫的密碼 —— 有些工具會在派生金鑰前把密碼轉成小寫，那樣產生的檔案要用小寫密碼才開得了。

### 在管線或排程裡輸入分量與密碼

stdin 不是終端機時，金鑰分量、PIN、PKCS#8 密碼都改讀一般的一行，可正常腳本化：

```
printf 'y\n%s\ny\n%s\ny\n' "$C1" "$C2" | kmu import ... --clear-components 2
```

排程裡 stdin 常是空的（例如 `</dev/null`），這時讀到的 PIN 是空字串，kmu 會以「PIN 是空的」結束而不登入。PIN 仍建議用 `--pin-env` 或 `--pin-file` 提供。

### 找不到 PKCS#11 library

依序檢查：

1. `/etc/Chrystoki.conf`（Windows：`C:\Program Files\SafeNet\LunaClient\crystoki.ini`）存在且可讀，`Chrystoki2` 區段裡有 `LibUNIX64`（Windows：`LibNT64` 或 `LibNT`）。只認 64-bit 的鍵 —— `LibUNIX`／`LibNT32` 是 32-bit 版，不會被採用。
2. 設定檔在別處時，`ChrystokiConfigurationPath` 要指向它**所在的目錄**。
3. 以上都不成立時用 `--library <路徑>` 直接指定。

環境變數**有設**卻讀不到設定檔時，兩個平台都會當場回報錯誤。設定檔讀得到但沒有上述的鍵，或沒設環境變數而預設位置的設定檔缺席時：

- Linux 退回 `dlopen("libCryptoki2_64.so")`；若該檔不在 loader 的搜尋路徑上，錯誤會延到載入時才出現。
- Windows 沒有退路，當場回報錯誤。

設定類的錯誤以結束代碼 2 回報。