Cryptography 101

Maojui

Outline

- Introduction & Tools
- Classical
- PRNG
- Modern
- Hash & MAC

Introduction

密碼學

- 加密演算法 ： 將資訊 (明文) 轉換成難以理解的資料 (密文) 的過程。
- 解密演算法 ：與上述相反的過程，由密文轉換回明文。

資訊安全 | Information Security

- Confidentiality（機密性）
- Integrity（完整性）
- Authentication（認證性）

Security Through Obscurity (STO)

- Nobody will ever find it.

~~Security Through Obscurity (STO)~~

- ~~Nobody will ever find it.~~
- Security experts have rejected this view as far back as 1851, and advise that obscurity should never be the only security mechanism.

柯克霍夫原則 | Kerckhoffs's principl

- 即使非數學上不可破解，系統也應在實質（實用）程度上無法破解。
- 即使落入敵人手中也不會造成困擾。
- 密匙必須易於溝通和記憶，且雙方可以容易的改變密匙。
- 系統應可以用於電訊。
- 系統應可以攜帶，不應需要兩個人或以上才能使用。
- 系統應容易使用，也無需記得長串的規則。

柯克霍夫原則 | Kerckhoffs's principl

- 不是說密碼學演算法都必須公開
- 但要確保即使公開也不會傷害安全性

Never write your own encryption algorithm

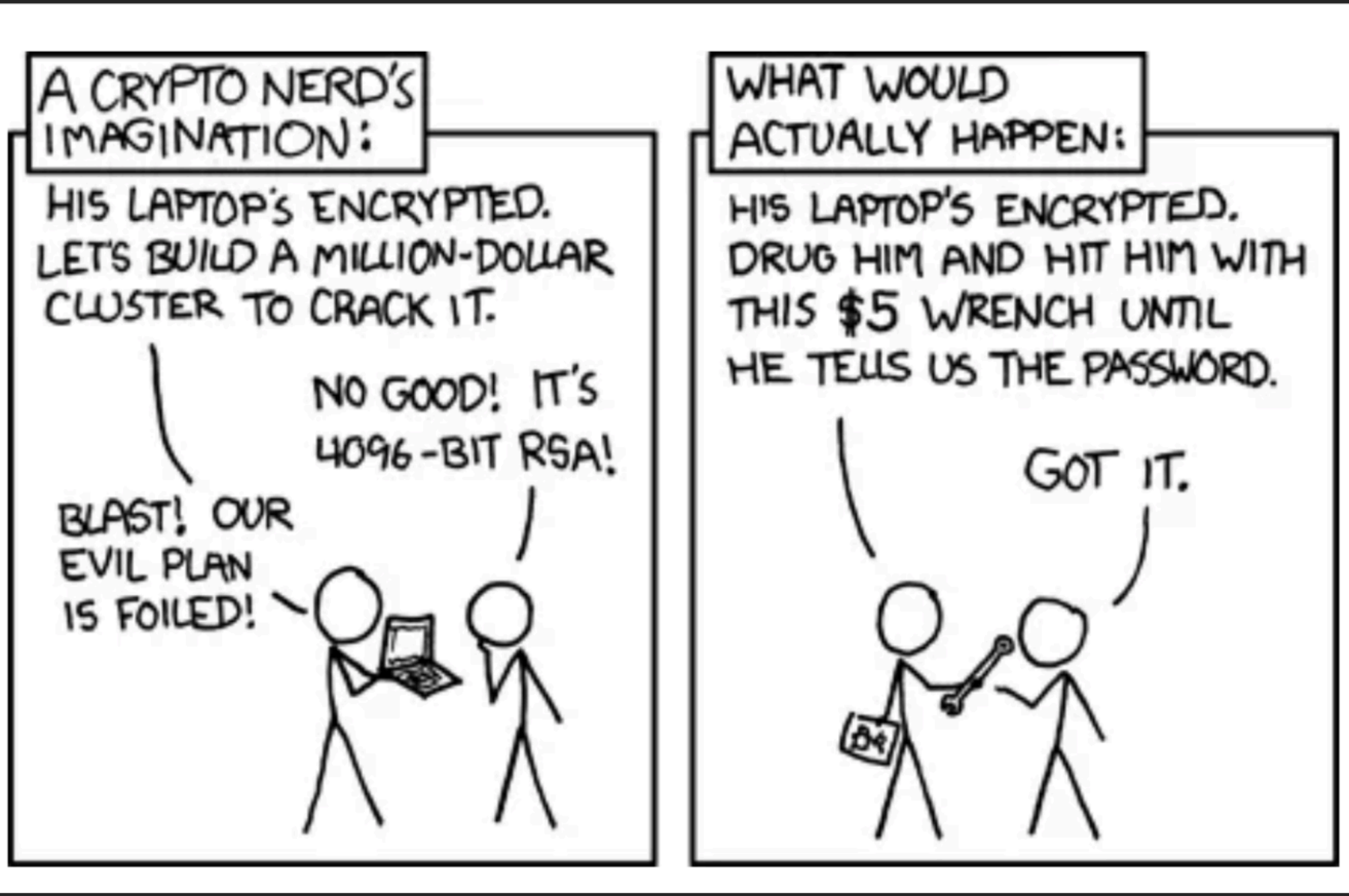

你有沒有想過在解密的人，真正追求的是什麼呢？

密碼不是靠暴力破解

而是 bypass


يا الله
麥當勞歡樂送!

開門 查水錶


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Danger

This is a “Hazardous Materials” module. You should **ONLY** use it if you’re 100% absolutely sure that you know what you’re doing because this module is full of land mines, dragons, and dinosaurs with laser guns.

```
>>> from cryptography.hazmat.backends import default_backend
>>> from cryptography.hazmat.primitives.asymmetric import rsa
>>> private_key = rsa.generate_private_key(
...     public_exponent=65537,
...     key_size=2048,
...     backend=default_backend()
... )
```

Cryptography : https://cryptography.io/en/latest/hazmat/primitives/

Tools

- Pwntools - Python’s Wonderful Networking Tools

- yafu - A factorization tool
- libnum - Number theory
- sympy - A library for symbolic mathematics.
- gmpy2 - Multiple-precision arithmetic & number theory
- Factordb - A large database of factor
- SageMath / CoCalc - Computer algebra system
- pyCrypto / cryptography - Crypto algorithms

Classical Cryptography

異或 | Exclusive or (XOR)

- XOR 是 Involutory function
- (A⊕B)⊕B = A

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

凱薩密碼 | Caesar cipher

- 加密 ： $E_k(m) = (m + k) \bmod 26$
- 解密 ： $D_k(c) = (c - k) \bmod 26$

凱薩密碼 | Caesar cipher

	2	11	0	18	18	8	2	0	11		8	18		1	14	17	4	3
明文	C	L	A	S	S	I	C	A	L		I	S		B	O	R	E	D

Key = 8

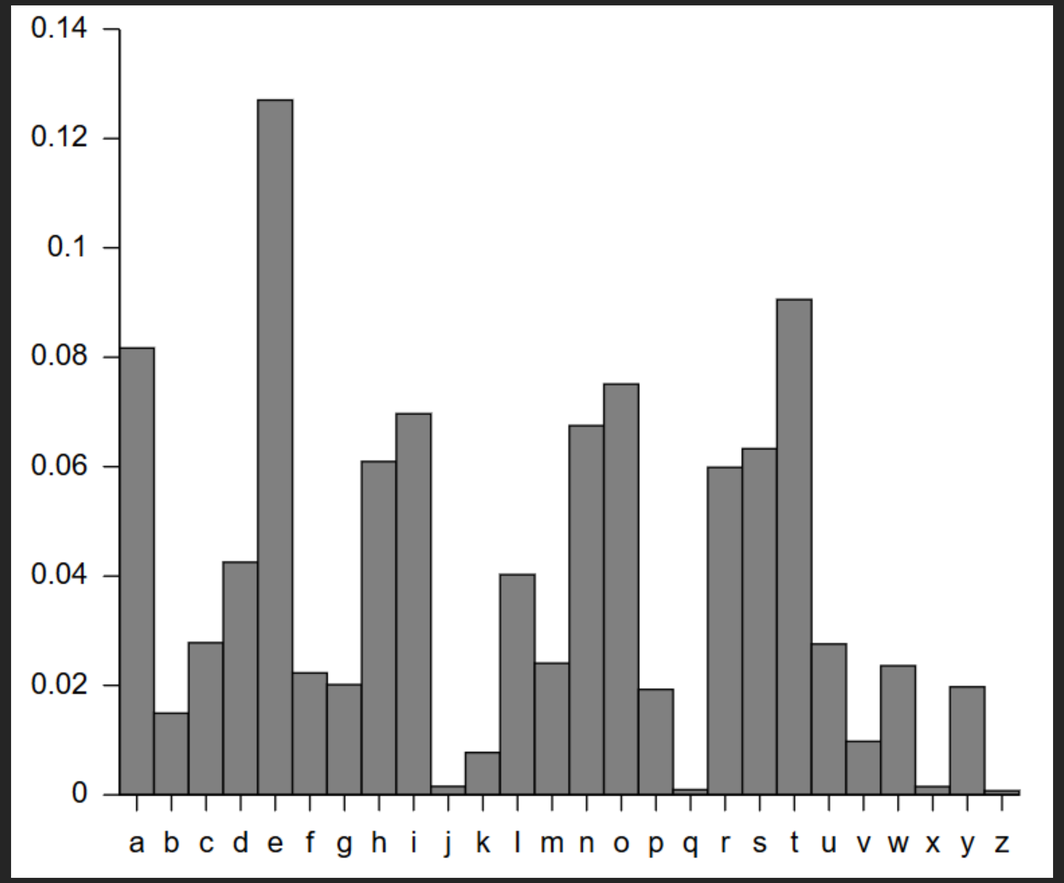
	10	19	8	0	0	16	10	8	19		16	0		9	22	25	12	11
明文	K	T	I	A	A	Q	K	I	T		Q	A		J	W	Z	M	L

維吉尼爾密碼 | Vigenère cipher

- 一堆 Caesar Cipher
- Key 從字母變單字

詞頻分析 | Frequency analysis

- 算出他的語言（Index of Coincidence）
- 從他的字母分布回推
- Tool : xortool, quipquip



一次性密碼本 | One time pad (OTP)

```
ONETIMEPAD = 14 13 04 19 08 12 04 15 00 03
```

```
UDCKSQUEOP = 20 03 02 10 18 16 20 04 14 15
```

```
IQGDACYTOS = 08 16 06 03 00 02 24 19 14 18
```

一次性密碼本 | One time pad (OTP)

- 用嚴密的演算法得到 **Truly random key** 來加密 (Caesar, XOR …)
- Key 只使用一次，用完即丟。
- 即使算出上一段的 Key 下一段仍然無法預測。

一次性密碼本 | One time pad (OTP)

- 安全的共享一個和明文一樣長的 `key ...` 或亂數演算法
- `truly random` 不易實現
- 一次性密碼本成本高，常常因多次使用造成問題 `QQ`

Many time pad

- $M_1 \oplus OTP = C_1$
- $C_1 \oplus OTP = M_1$
- $C_1 \oplus M_1 = OTP$????

Many time pad

- $C_1 \oplus OTP = M_1$
- $C_2 \oplus OTP = M_2$
- $C_3 \oplus OTP = M_3$
- $C_4 \oplus OTP = M_4$
- $C_5 \oplus OTP = M_5$
- $C_6 \oplus OTP = M_6$
- $C_7 \oplus OTP = M_7$
- $C_8 \oplus OTP = M_8$
- $C_9 \oplus OTP = M_9$

- 找到一個 OTP 讓 M1 ~ M9 看起來很正常

[LAB] N-Time pad

Pseudo Random Number Generator

LCG | Linear congruential generator

- $S_{i+1} = (aS_i + b) \bmod N$: Crack
- 變形 ：
  - 僅保留 high order bit
  - 僅保留 low order bit

XORShift

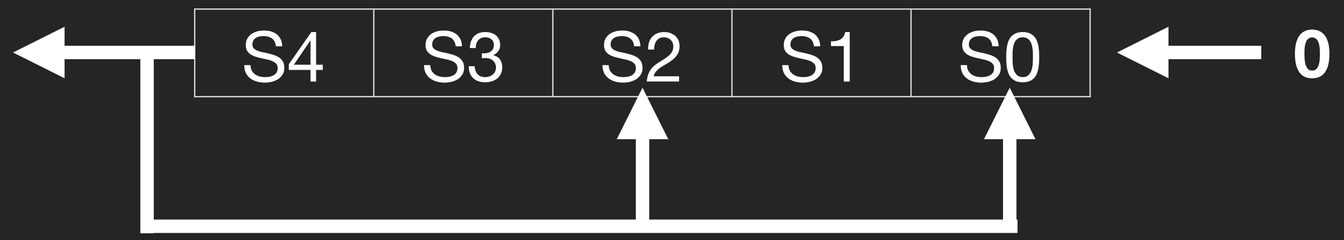
- 運算速度快，程式碼簡單
  - $x \oplus= x \ll a;$
  - $x \oplus= x \gg b;$
  - $x \oplus= x \ll c;$

- 分析 ：

  把 x 當作 bit vector, ⊕和≪ 可以寫成 GF(2) 下的矩陣乘法

LFSR | Linear feedback shift register

- 暫存器初始值：`seed`

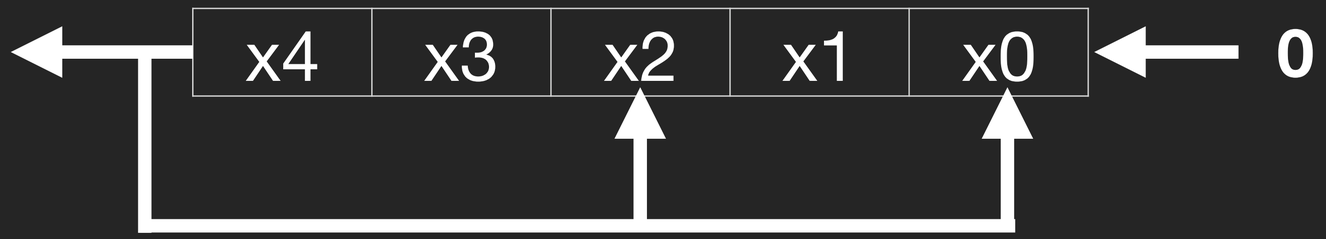


- 一次輸出一個 `bit`
  - 先 `shift` 一格
  - 輸出的 `bit` 回來 `xor` 指定的位置

- 分析 ：
  - 把 x 當作 `bit vector, ⊕` 和 `≪` 可以寫成 `GF(2)` 下的矩陣乘法

    [Berlekamp massey algorithm]

LFSR | Linear feedback shift register



k	LFSR coefficients 100101	$H_{25} = GF(2)[x]/p(x)$, $p(x) = x^5 + x^2 + 1$
0	$S[0] =$00001	$x^0 = 0 + 0 + 0 + 0 + 1$
1	$S[1] =$00010	$x^1 = 0 + 0 + 0 + x + 0$
2	$S[2] =$00100	$x^2 = 0 + 0 + x^2 + 0 + 0$
3	$S[3] =$01000	$x^3 = 0 + x^3 + 0 + 0 + 0$
4	$S[4] =$10000	$x^4 = x^4 + 0 + 0 + 0 + 0$
5	$S[5] =$00101	$x^5 = 0 + 0 + x^2 + 0 + 1$
6	$S[6] =$01010	$x^6 = 0 + x^3 + 0 + x + 0$
7	$S[7] =$10100	$x^7 = x^4 + 0 + x^2 + 0 + 0$
8	$S[8] =$01101	$x^8 = 0 + x^3 + x^2 + 0 + 1$
9	$S[9] =$11010	$x^9 = x^4 + x^3 + 0 + x + 0$
10	$S[10] =$10001	$x^{10} = x^4 + 0 + 0 + 0 + 1$
11	$S[11] =$00111	$x^{11} = 0 + 0 + x^2 + x + 1$
12	$S[12] =$01110	$x^{12} = 0 + x^3 + x^2 + x + 0$
13	$S[13] =$11100	$x^{13} = x^4 + x^3 + x^2 + 0 + 0$

MT19937 | Mersenne Twister

- 通過很多測試，週期長，沒專利，好用
- 各大程式語言預設亂數：`Matlab, Php, Python, R, Ruby, …`

- Wiki 傳送門

CSPRNG | Cryptographically secure pseudo-random number generator

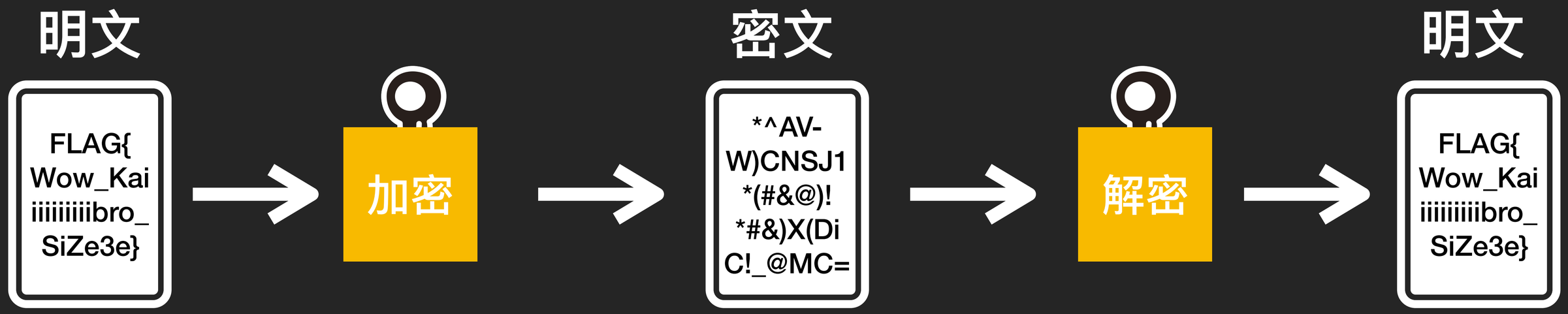
- 統計學偽隨機性
  - 0,1 出現的機率 ～1/2
- 密碼學安全偽隨機性
  - 沒有已知的多項式時間算法能以高於 1/2 的機率算出下一個 bit
- 真隨機性
  - 不可重現性

Modern Cryptogarphy

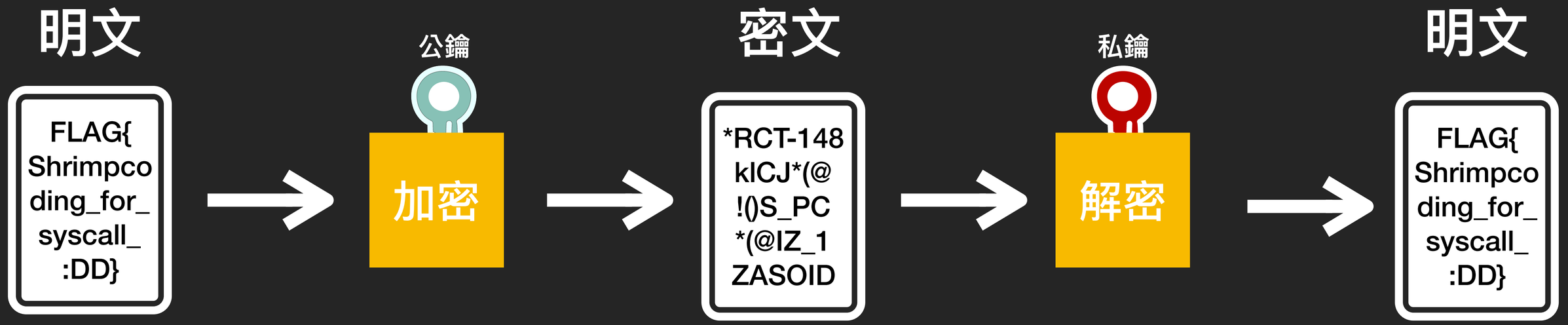
現代密碼學 | Modern Cryptography

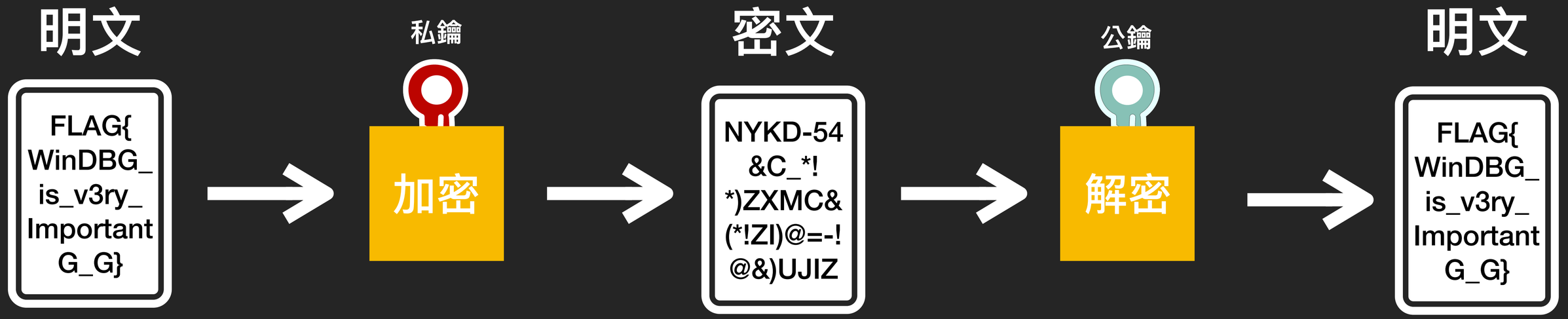
- 對稱式加密
  - 區塊密碼 (Block Ciphers)
  - 串流密碼 (Stream Ciphers)
- 非對稱式加密
  - …

對稱式加密 | Symmetric encryption

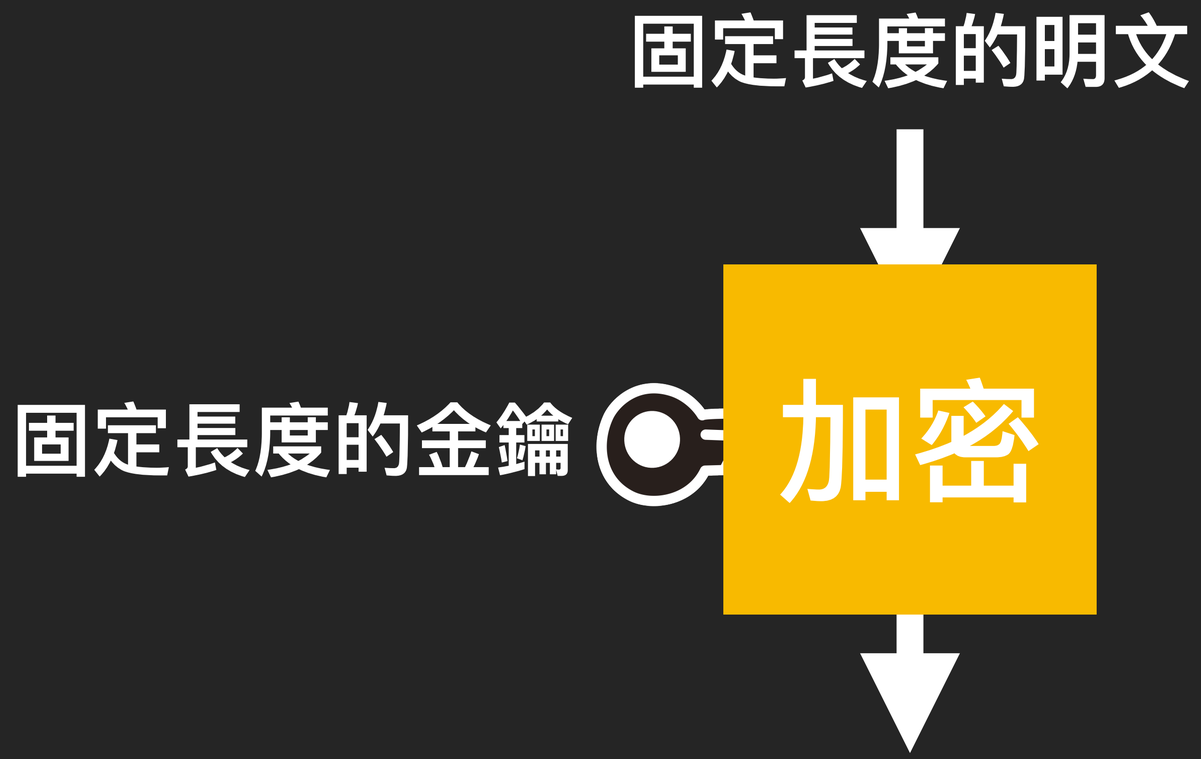


非對稱式加密 | Asymmetric encryption





區塊密碼 | Block cipher



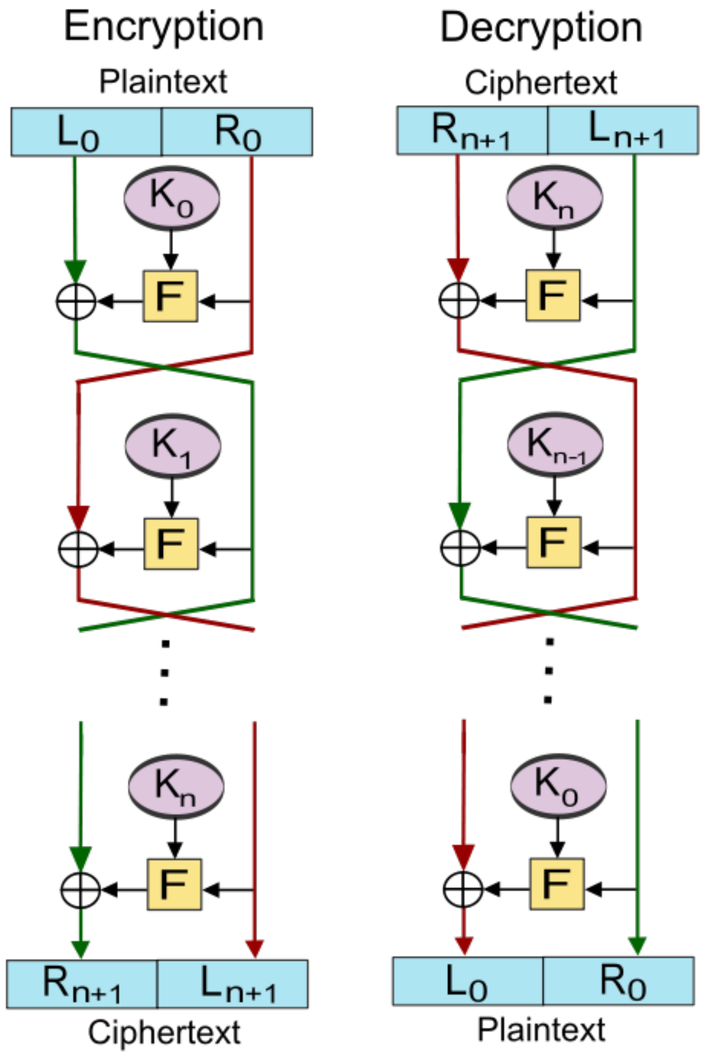
區塊密碼 | Block cipher

固定長度的密文

固定長度的明文

費斯妥結構 | Feistel structure

- 加密和解密操作非常相似
- 只需要逆轉 `Key` 的順序就可以加解密
- `F` 可以是任何一種純函數，不必是可逆的
- 例如：`SPN`、`Hash`、`Neural Network` …
- 應用：`FEAL, DES, Blowfish, TEA, …`



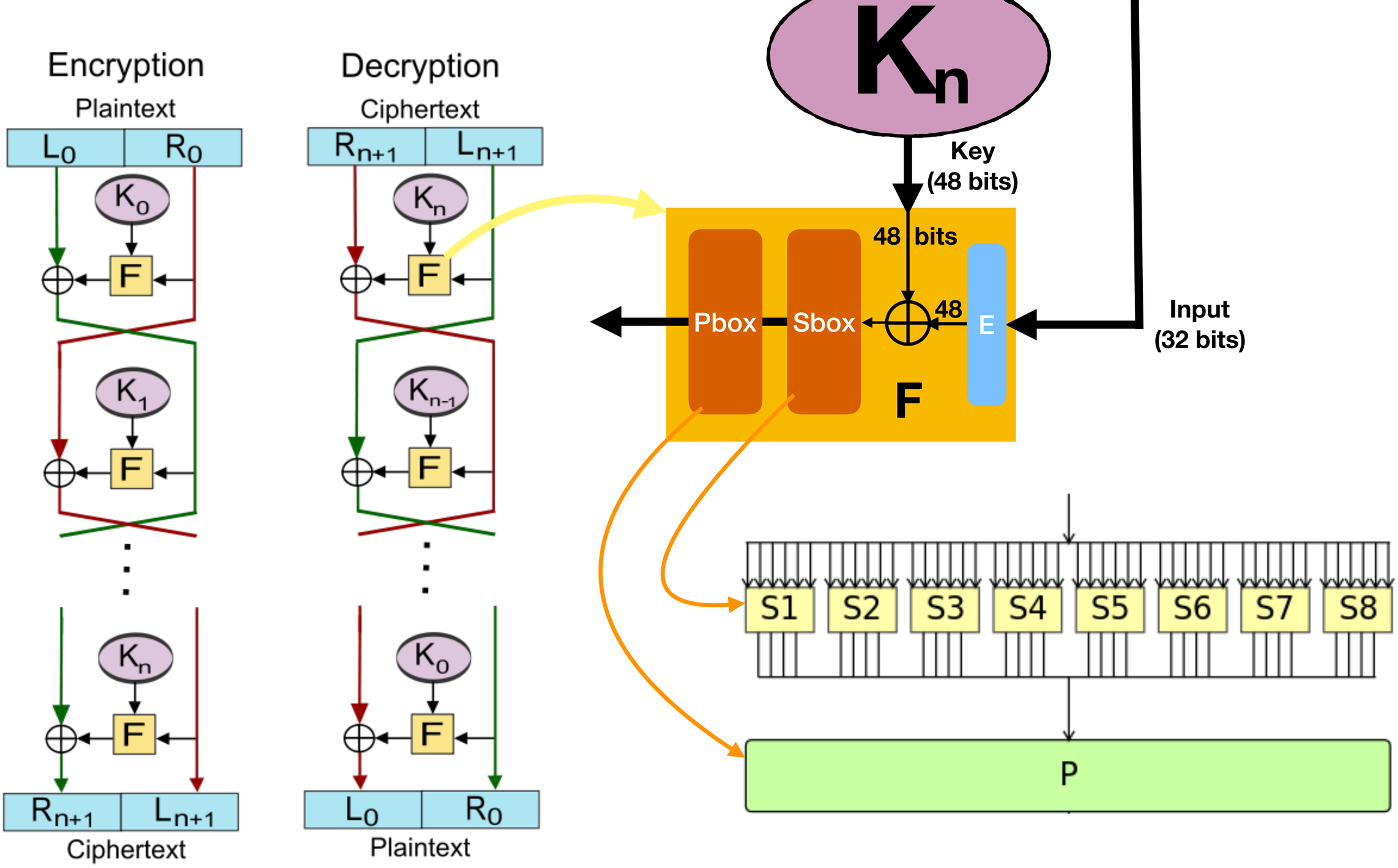
資料加密標準 | DES

- Block size : 64 bits (8 bytes)
- Keyspace : 56 bits

三重資料加密標準 | 3DES

- Block size : 64 bits (8 bytes)
- Keyspace : 168 bits

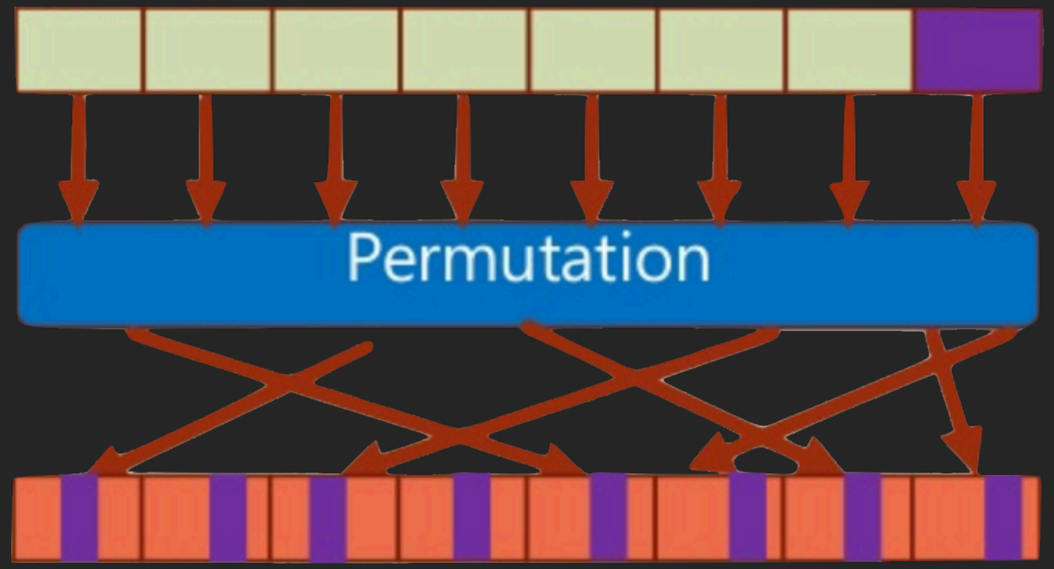
- Enc(p): $E_{K3}(D_{K2}(E_{K1}(p)))$
- Dec(c): $D_{K1}(E_{K2}(D_{K3}(c)))$


Encryption
Plaintext
L_0
R_0
K_0
F
K_1
F
K_n
F
R_{n+1}
L_{n+1}
Ciphertext
Decryption
Ciphertext
R_{n+1}
L_{n+1}
K_n
F
K_{n-1}
F
K_0
F
L_0
R_0
Plaintext
K_n
Key
(48 bits)
48 bits
Pbox
Sbox
48
E
F
Input
(32 bits)
S1
S2
S3
S4
S5
S6
S7
S8
P

Sbox | Substitution box

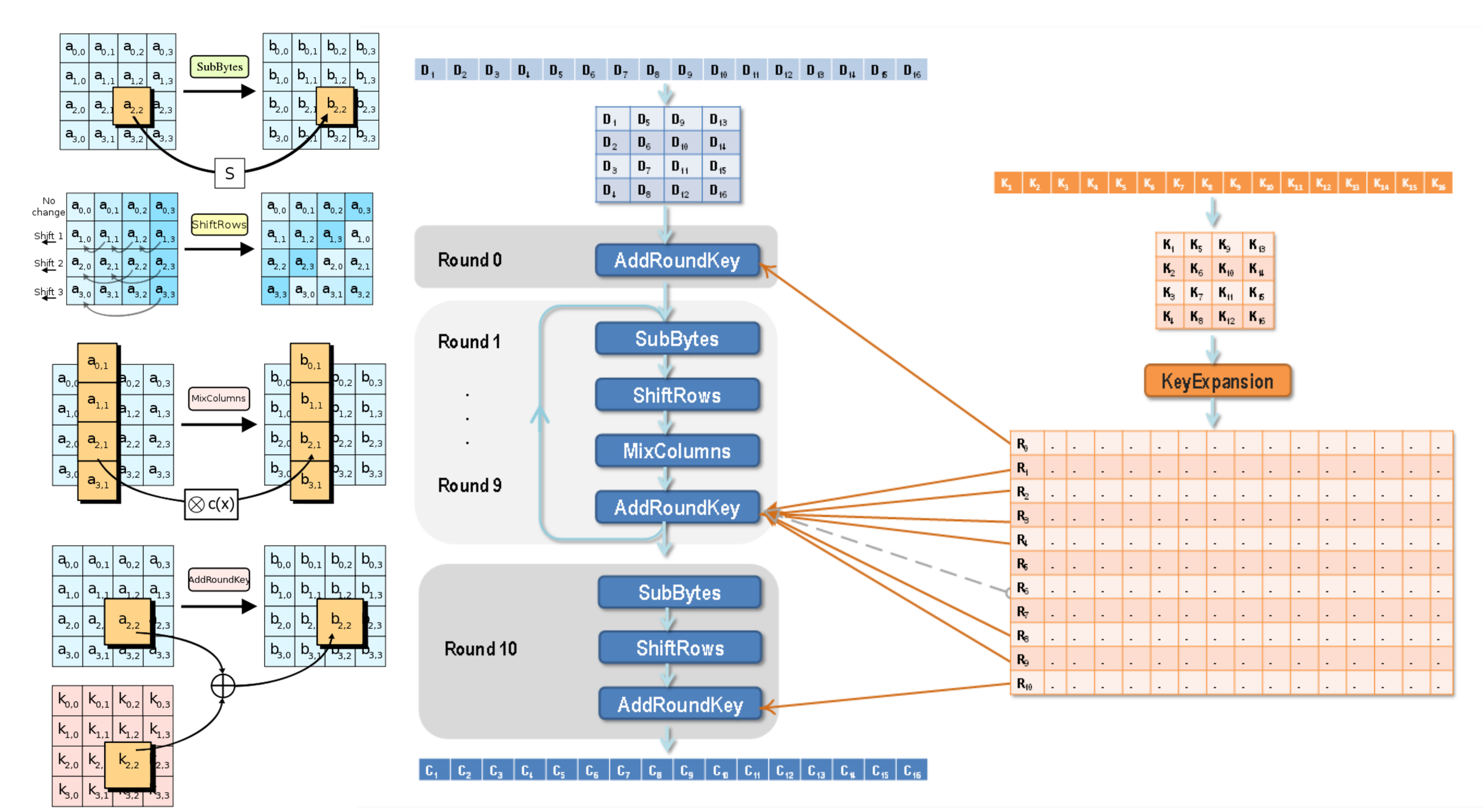
- `sbox = [14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7]`
- `Output = sbox[ input ]`

Pbox | Permutation box

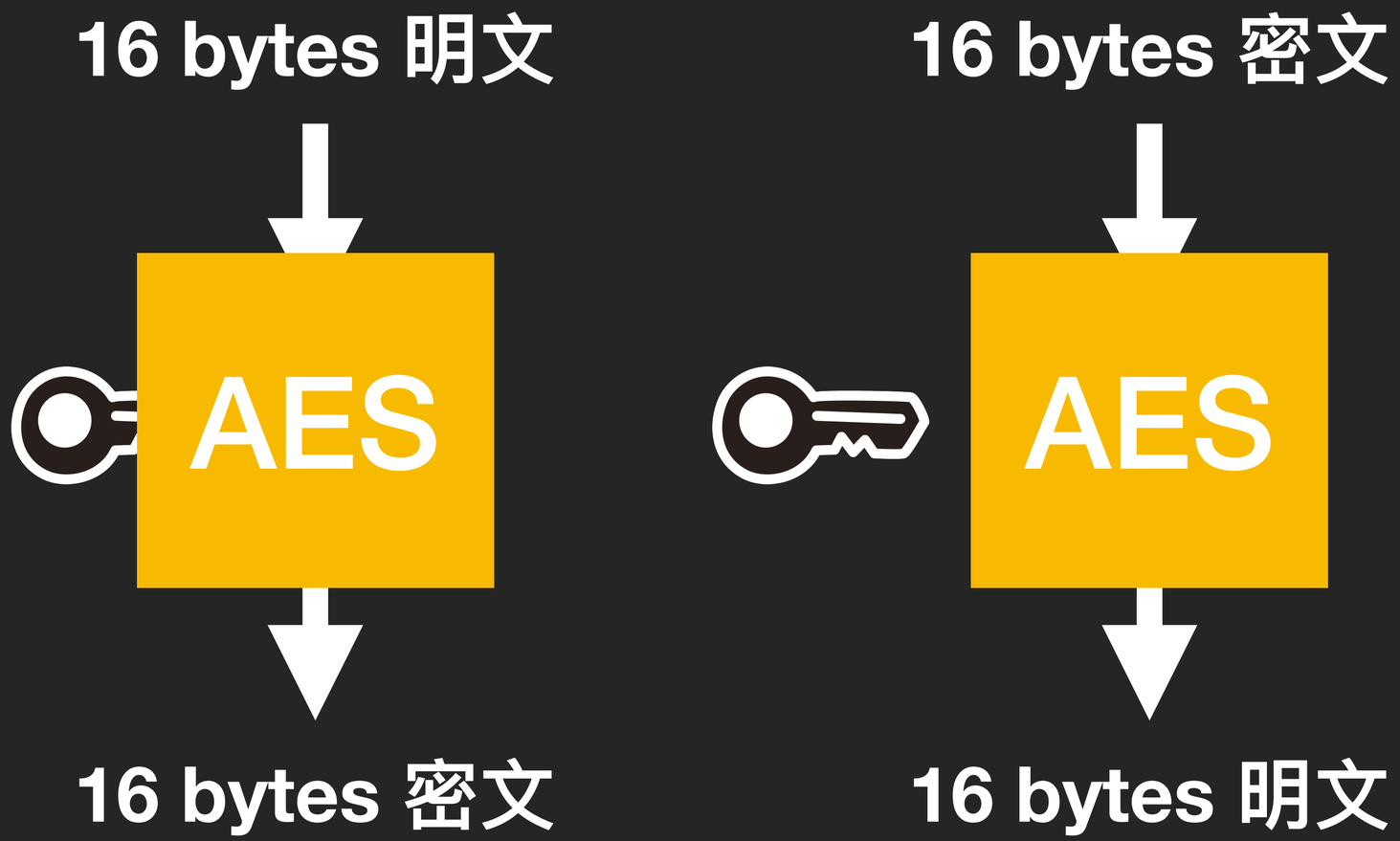


進階資料加密標準 | AES

- Block size : 128 bits (16 bytes)
- Keyspace : 128, 192, 256 bits

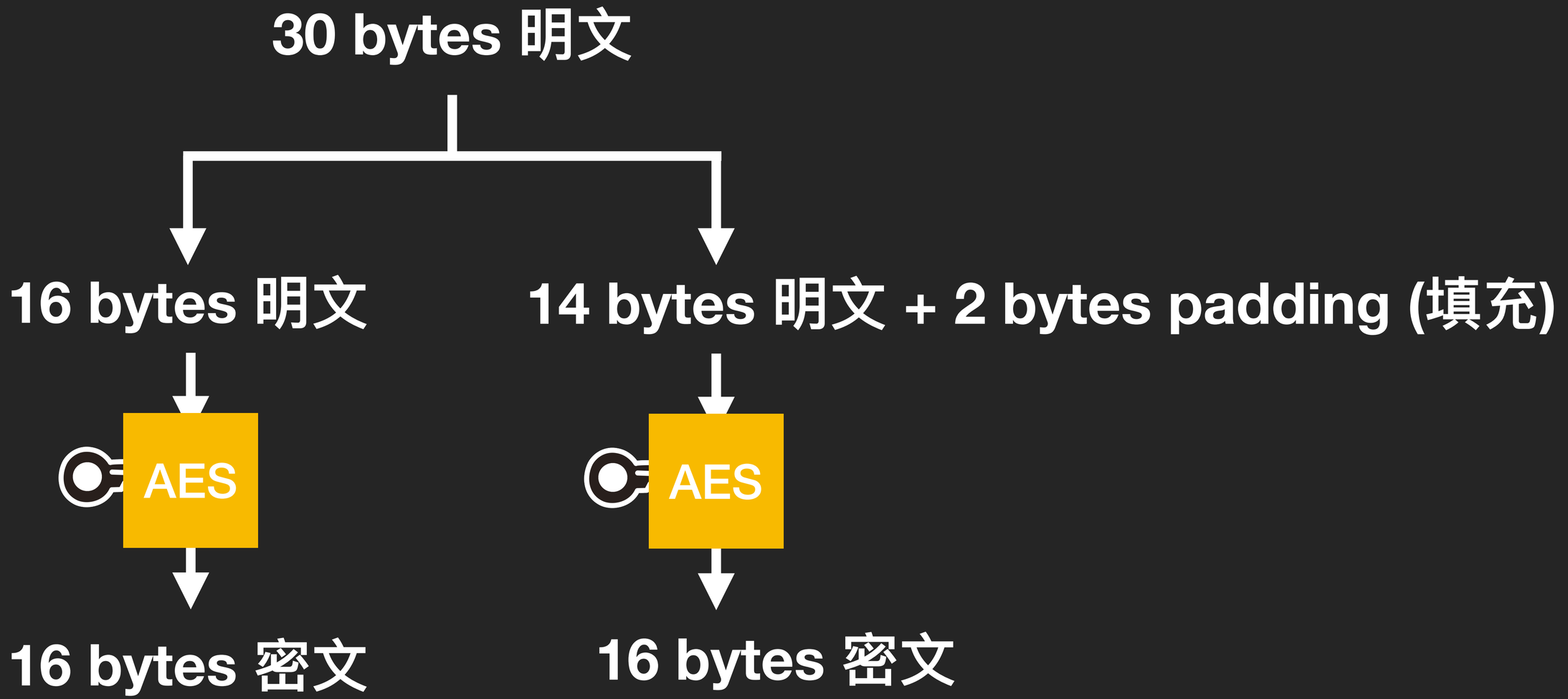

SubBytes
S
No change
Shift 1
Shift 2
Shift 3
ShiftRows
MixColumns
⊗ c(x)
AddRoundKey
KeyExpansion
Round 0
Round 1
Round 9
Round 10
SubBytes
ShiftRows
MixColumns
AddRoundKey
SubBytes
ShiftRows
AddRoundKey

區塊密碼 | Block cipher



一次只能 16 bytes 怎麼夠？

16 bytes傳三\xe5\xb0


30 bytes 明文
16 bytes 明文
14 bytes 明文 + 2 bytes padding (填充)
AES
AES
16 bytes 密文
16 bytes 密文

填充 | PKCS#7 Padding

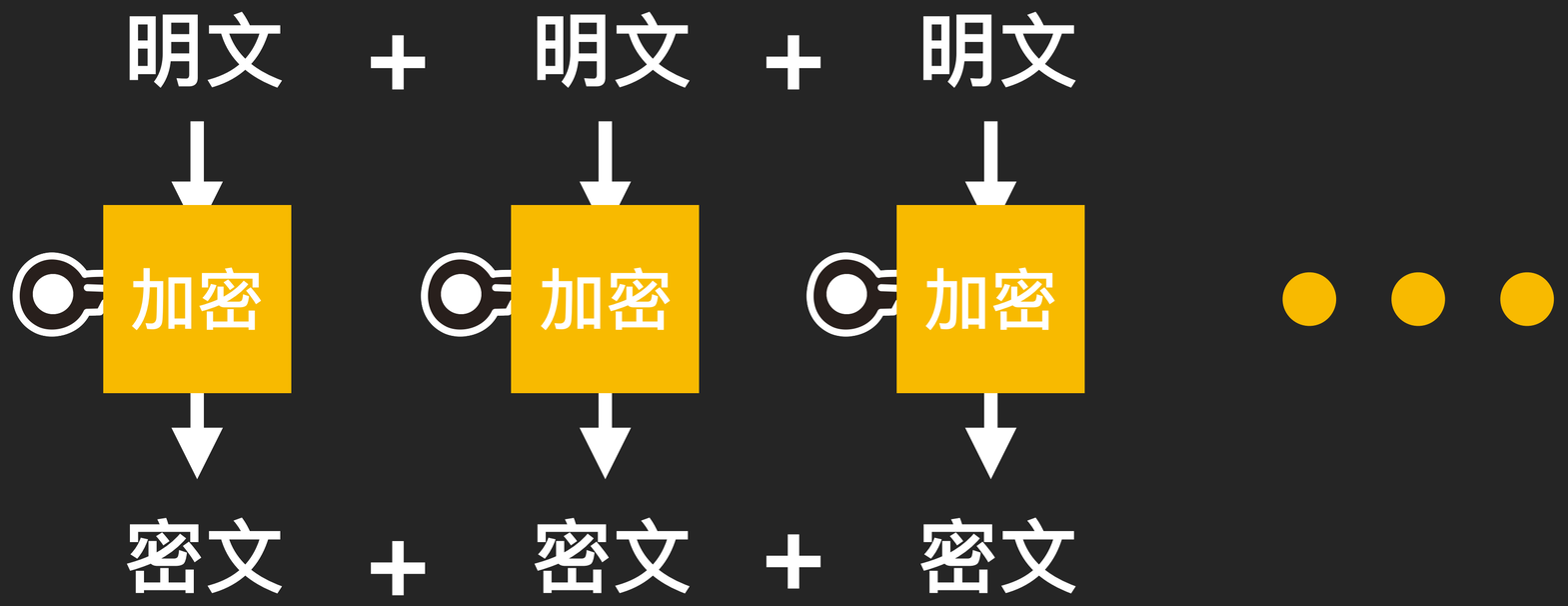
```
. . . . | FA CE B0 0C DE AD FF 01
. . . . | FA CE B0 0C DE AD 02 02
. . . . | FA CE B0 0C DE 03 03 02
. . . . | FA CE B0 0C 04 04 04 02
. . . . | FA CE B0 05 05 05 05 05
. . . . | FA CE 06 06 06 06 06 06
. . . . | FA 07 07 07 07 07 07 07
. . . . | 08 08 08 08 08 08 08 08
```

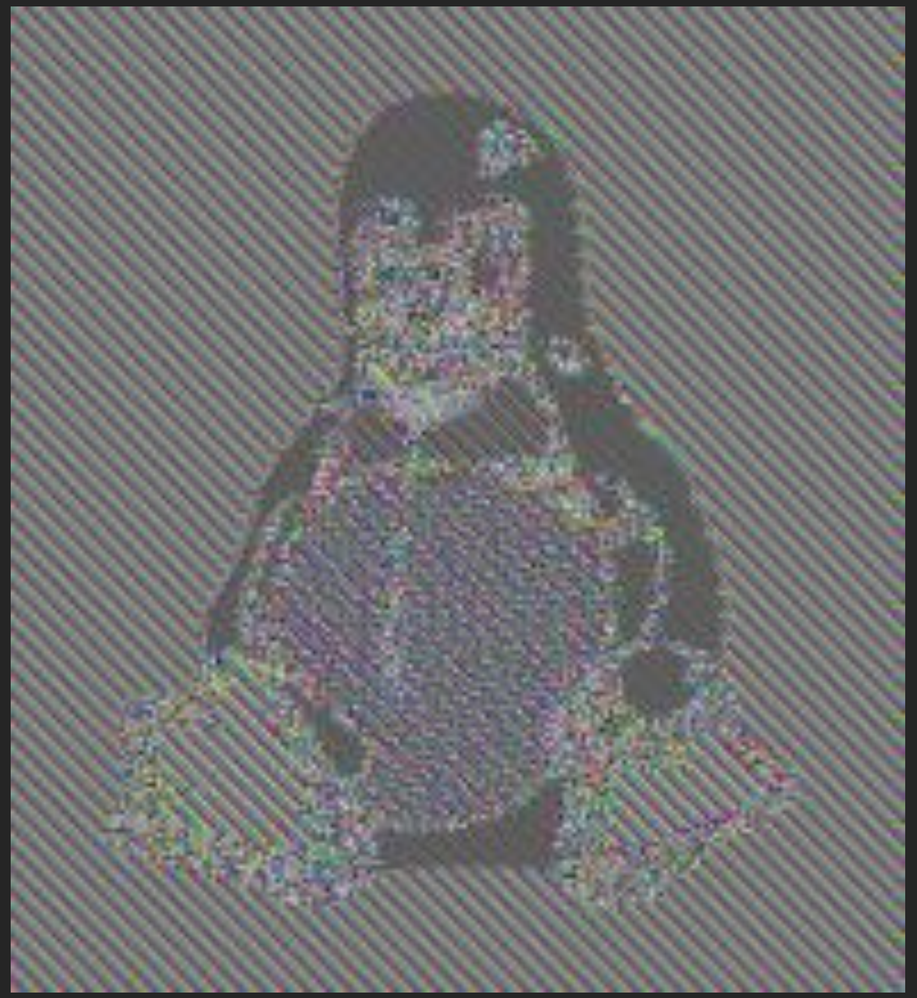
To avoid padding, you could use “Ciphertext stealing (CTS)”.

區塊加密法工作模式 | mode of operation

- ECB (Electronic codebook)
- CBC (Cipher-block chaining)

ECB | Electronic codebook



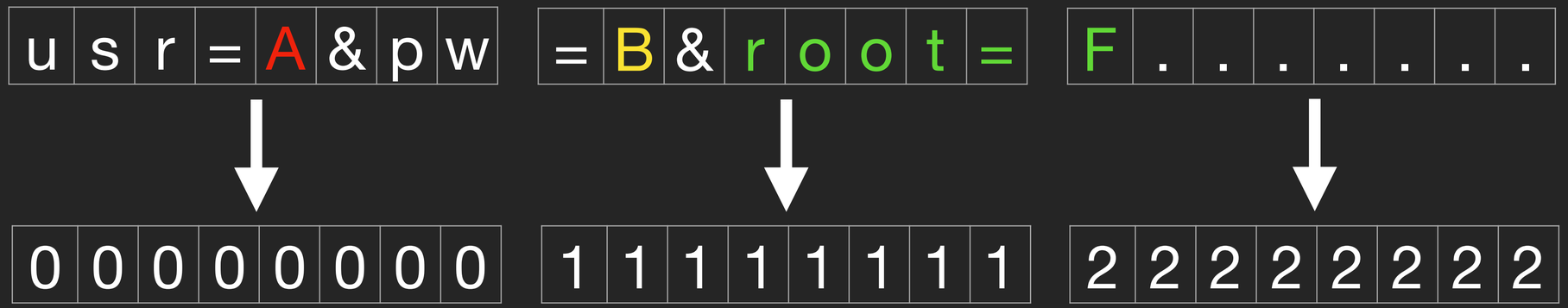


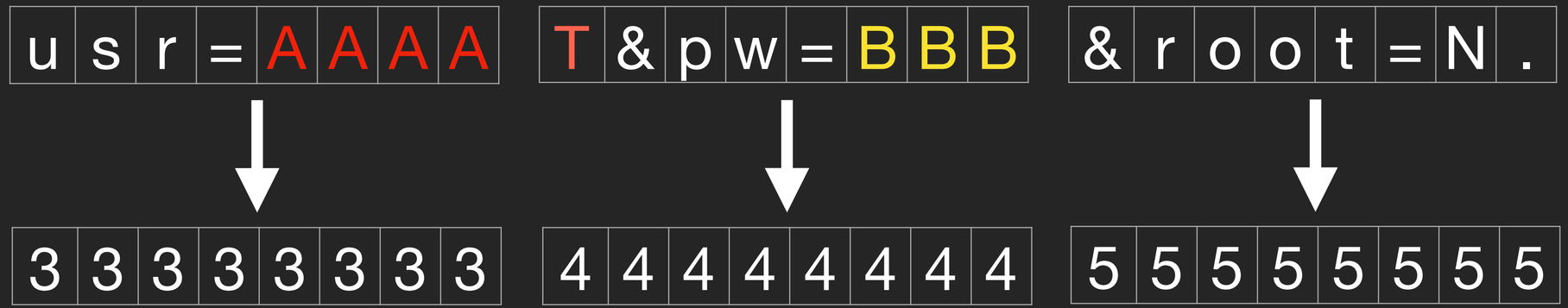
picoCTF{d0nt_r0ll_y0ur_0wn_aES}

Cut & Paste

- 研究表明：
- 漢字的順序並不定一能影響閱讀
- 比如你完看這句話才這發現裡的字全是亂的
- ECB 也一樣

Cut & Paste

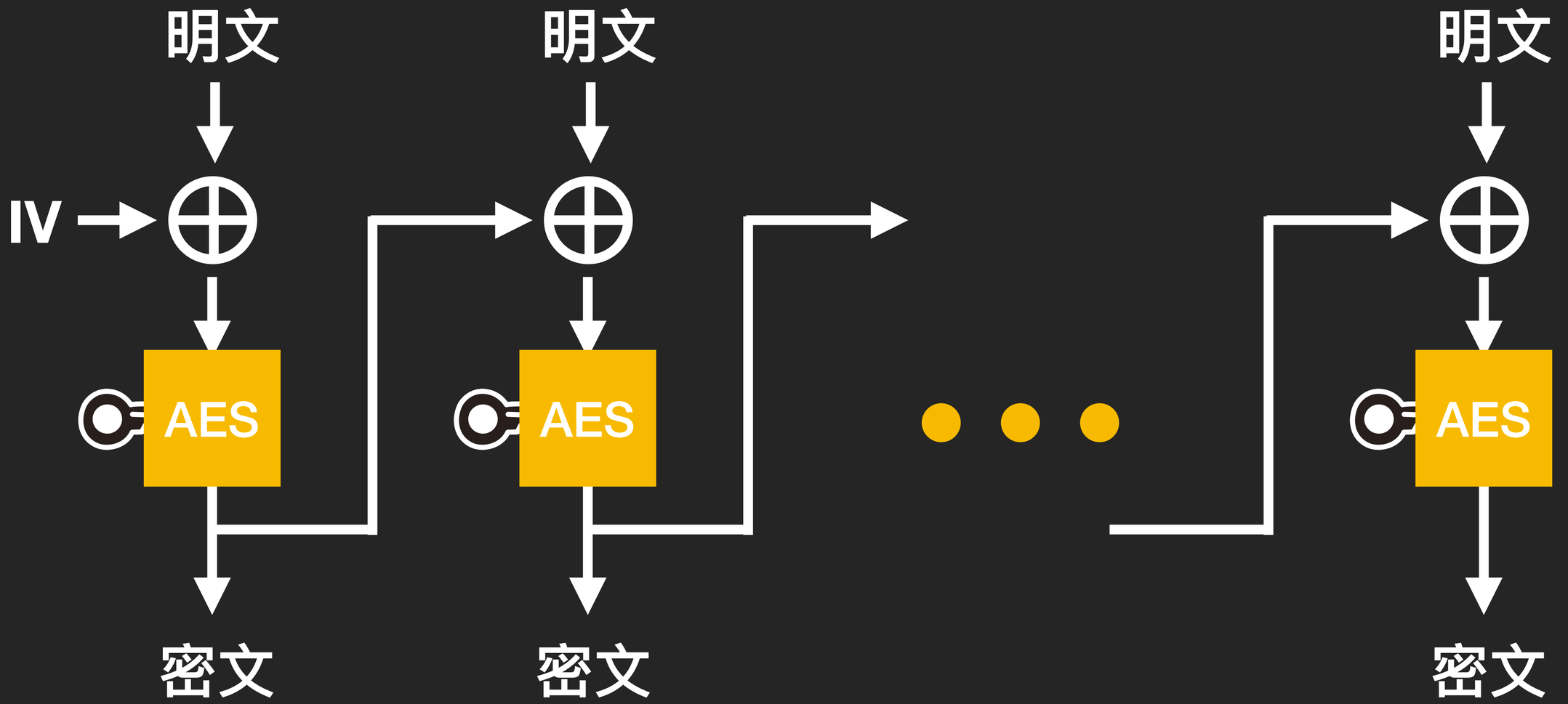

u s r = A & p w
= B & r o o t =
F . . . . . .
0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2


u s r = A A A A
T & p w = B B B
& r o o t = N .
3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5

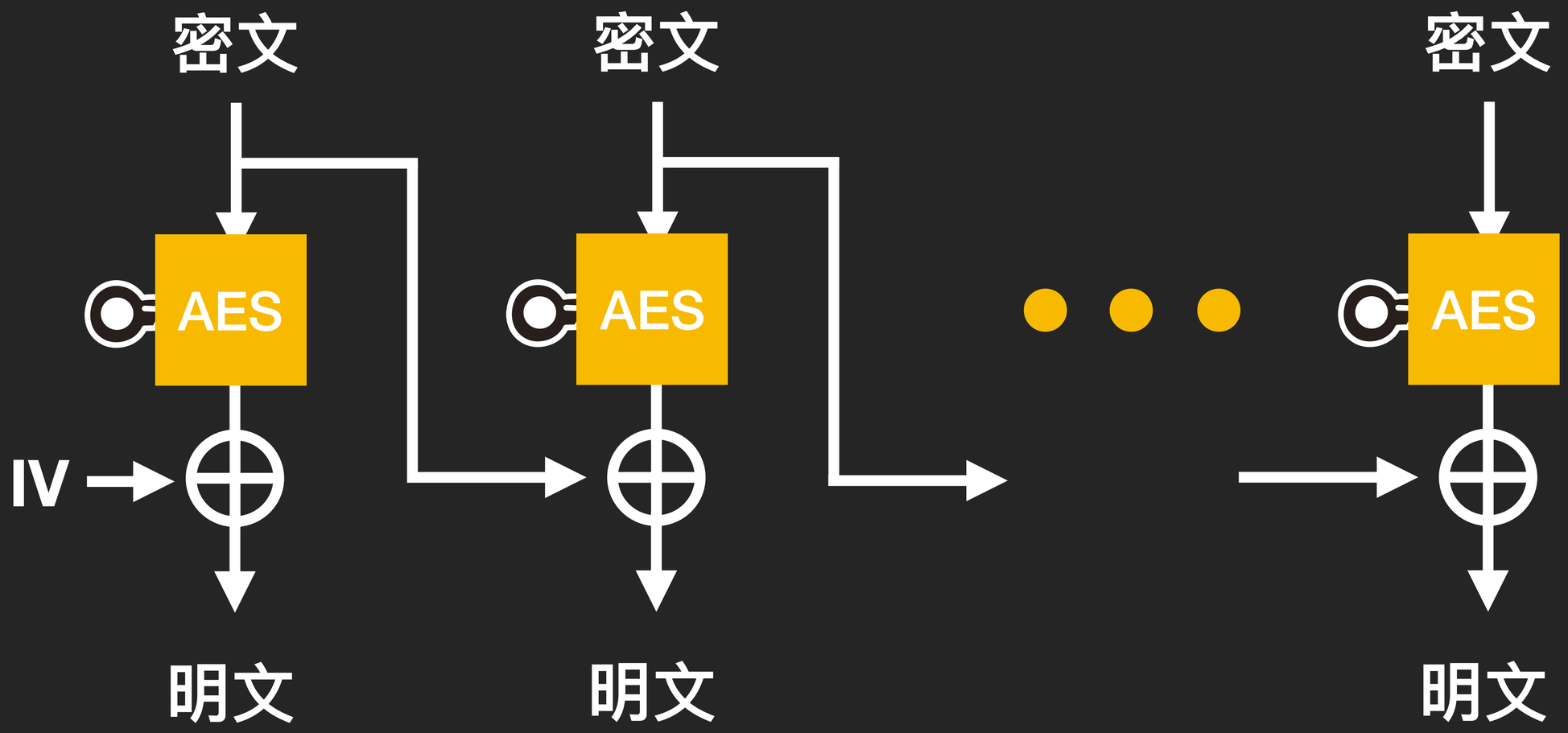
Cut & Paste

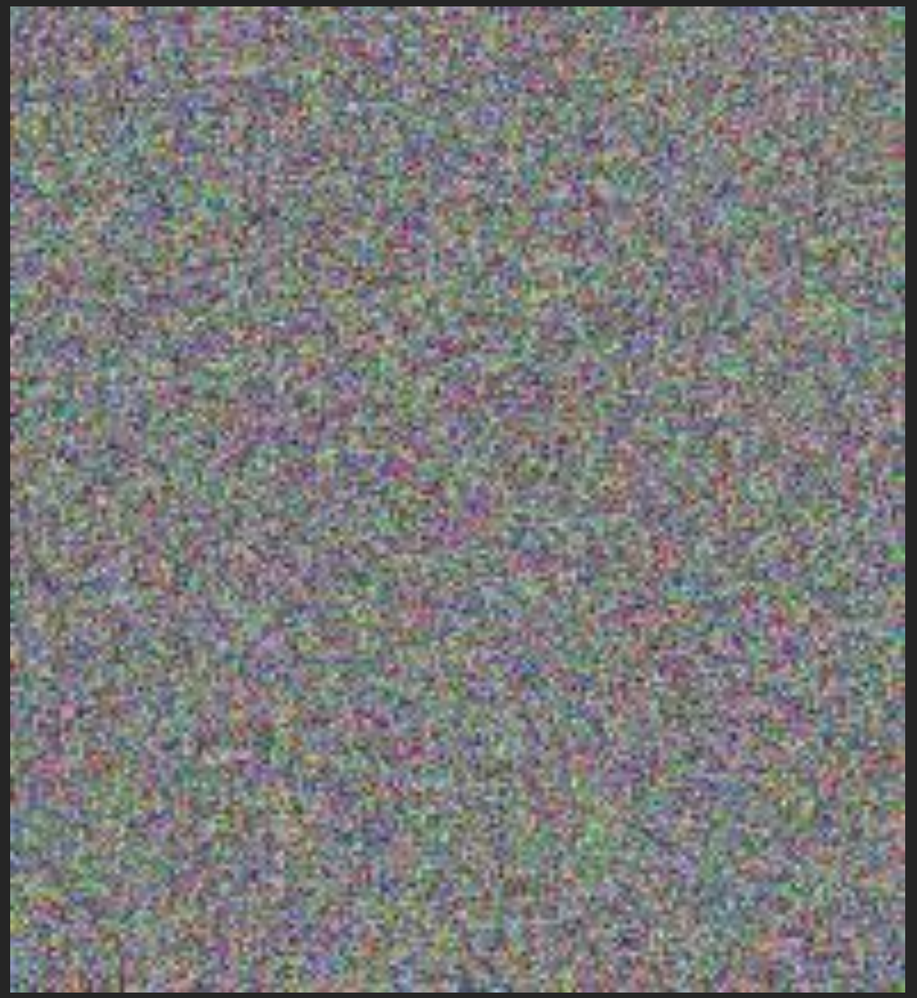

0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1
4 4 4 4 4 4 4 4
2 2 2 2 2 2 2 2
DES-ECB
Decryption
u s r = A & p w
= a & r o o t =
T & p w = a a a
F . . . . . .

CBC | Cipher-block chaining



CBC | Cipher-block chaining

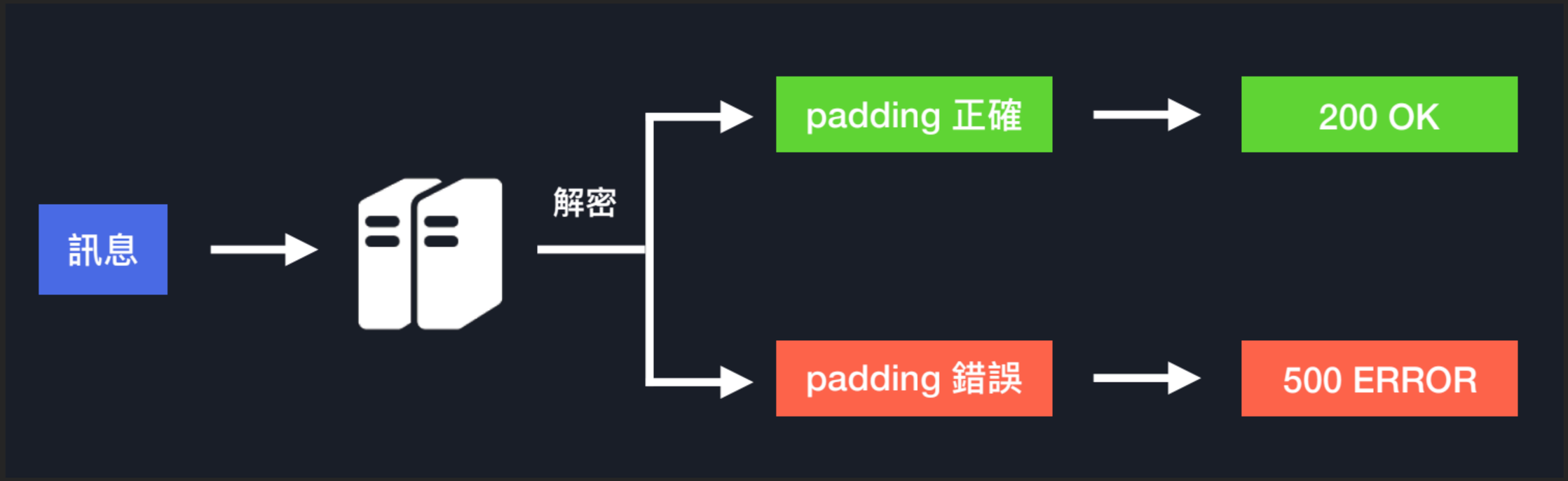




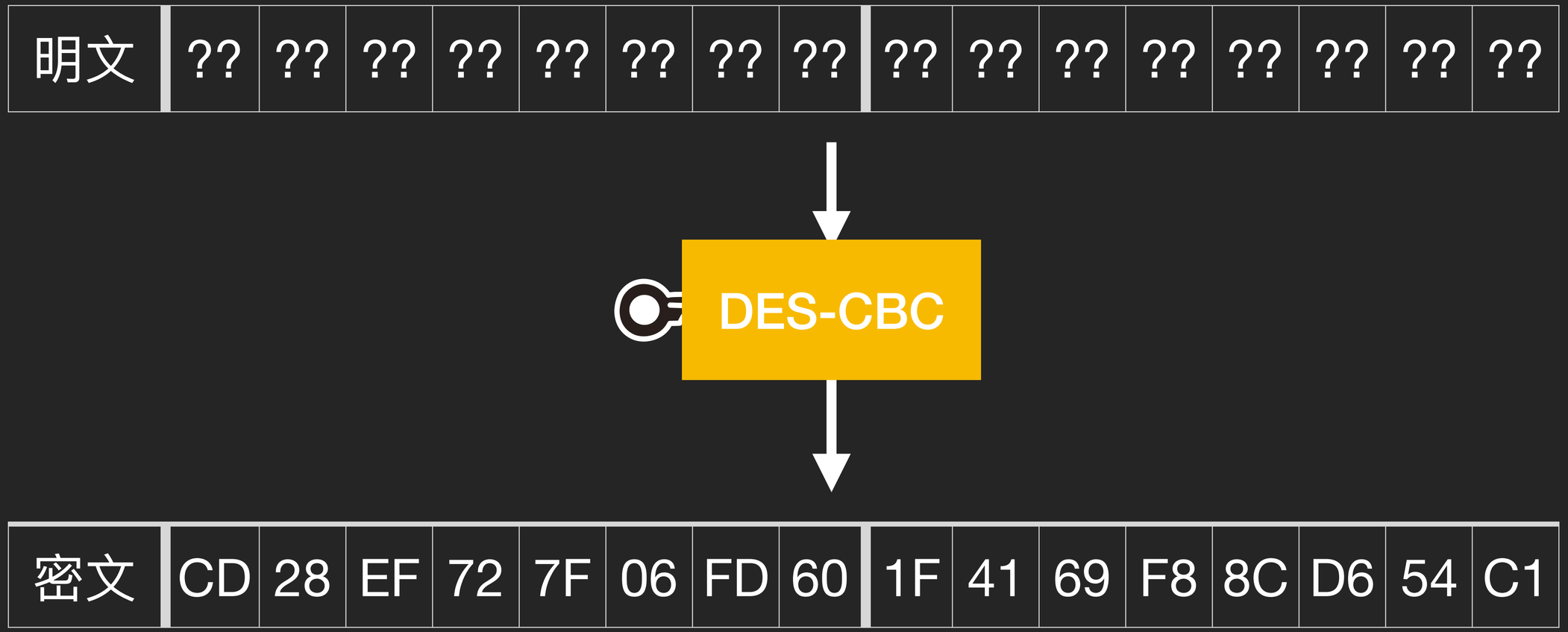
Padding Oracle Attack

- 攻擊對象：

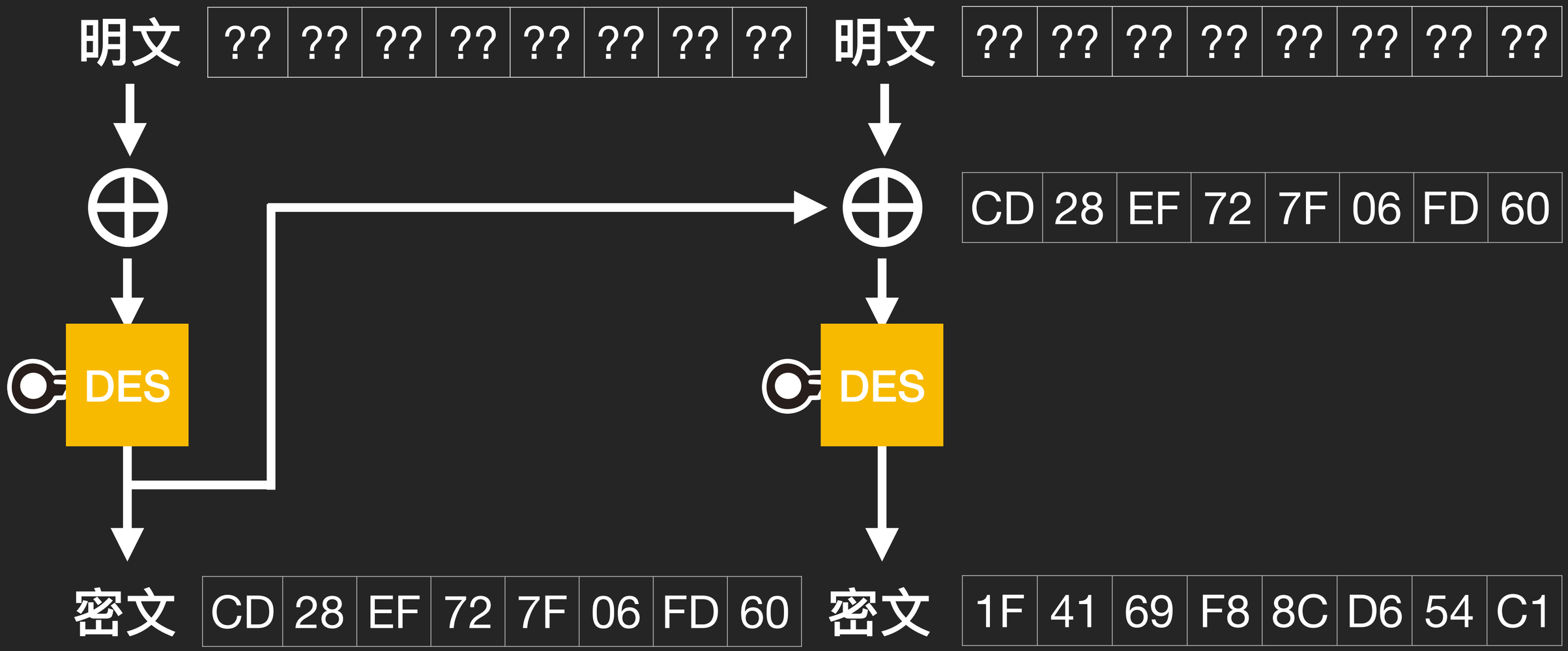
  使用 PKCS#7 作 Padding 的 CBC Block Cipher



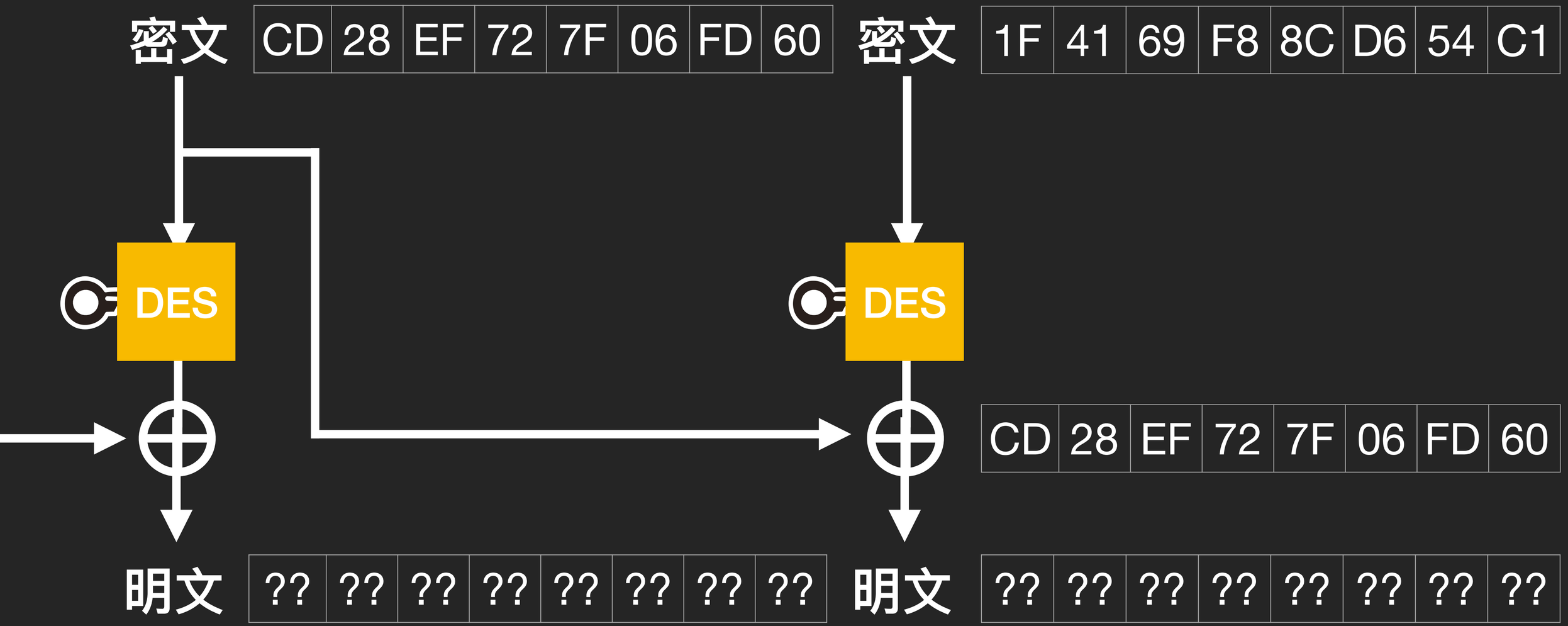
Padding Oracle Attack



Padding Oracle Attack

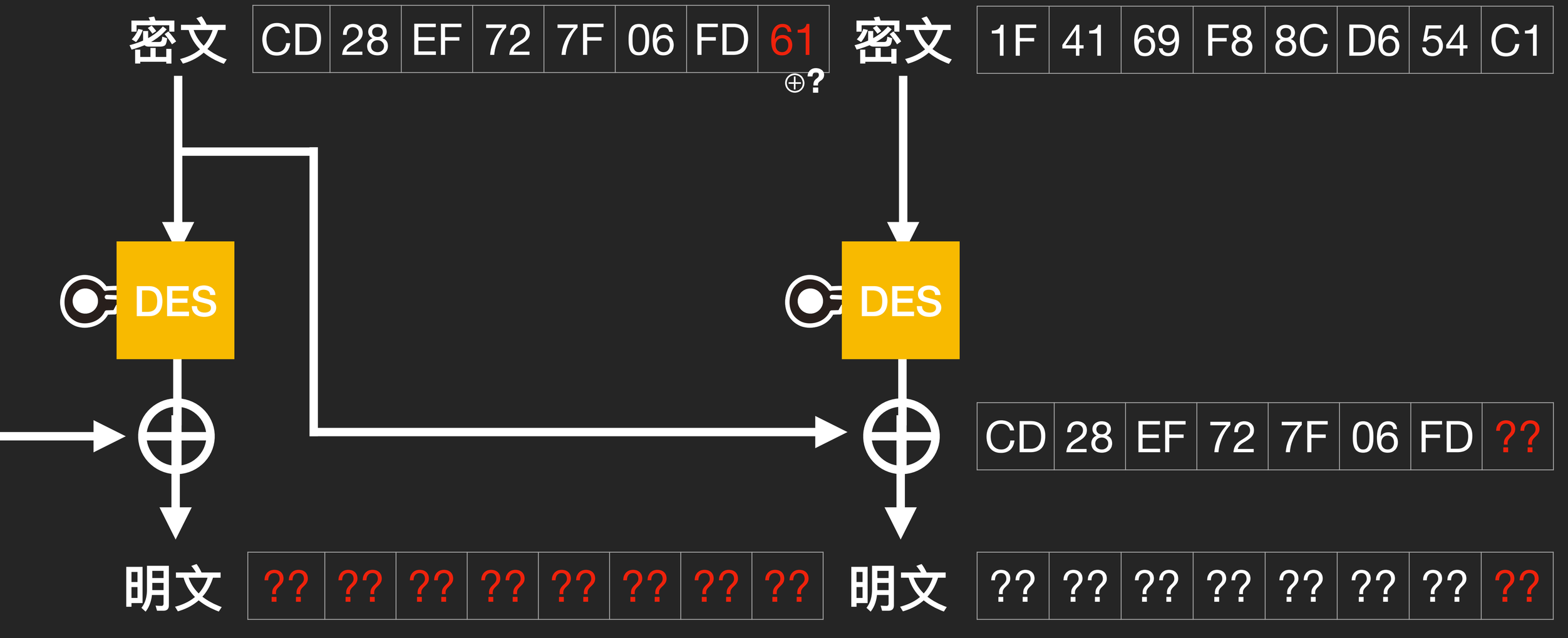


Padding Oracle Attack

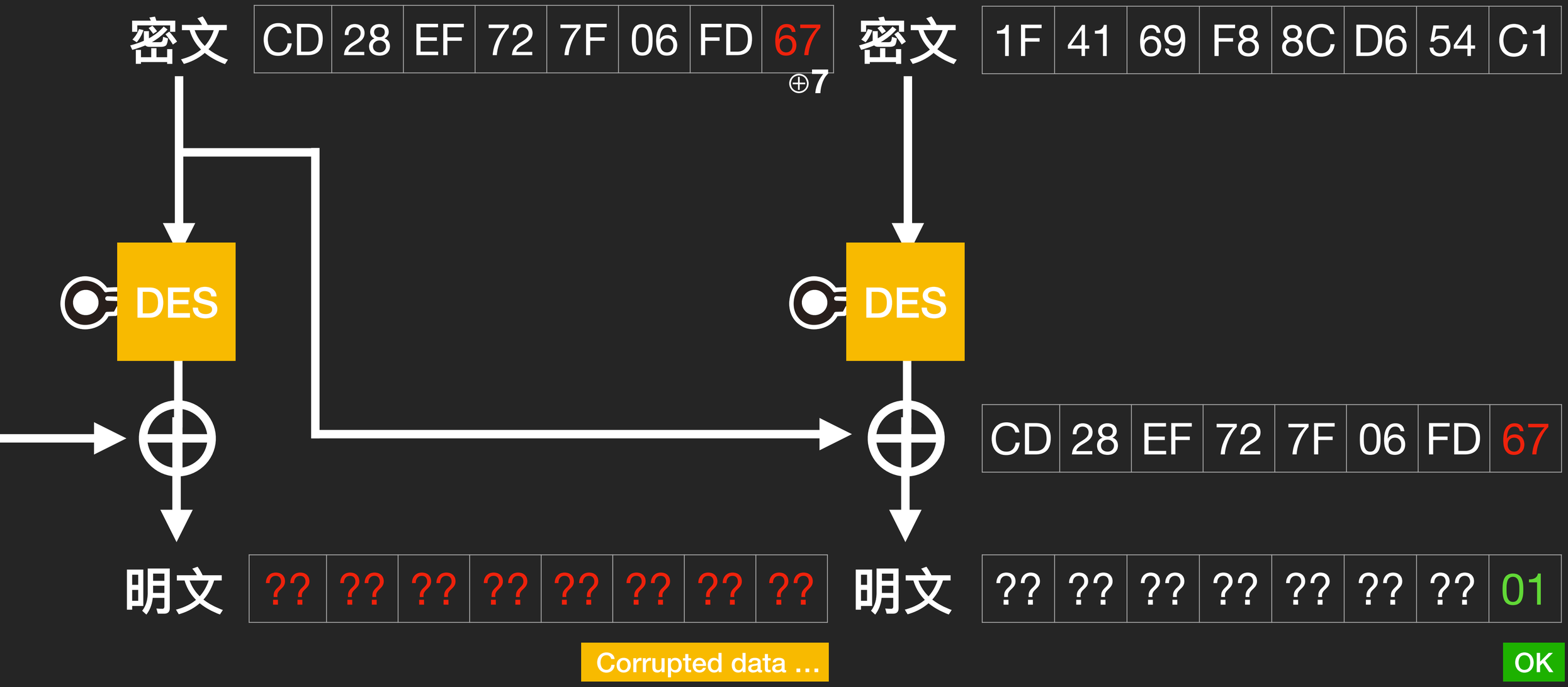


OK

Padding Oracle Attack



Padding Oracle Attack



Padding[-1] ⊕ 7 => 1

Padding[-1] ⊕ 7 => 1

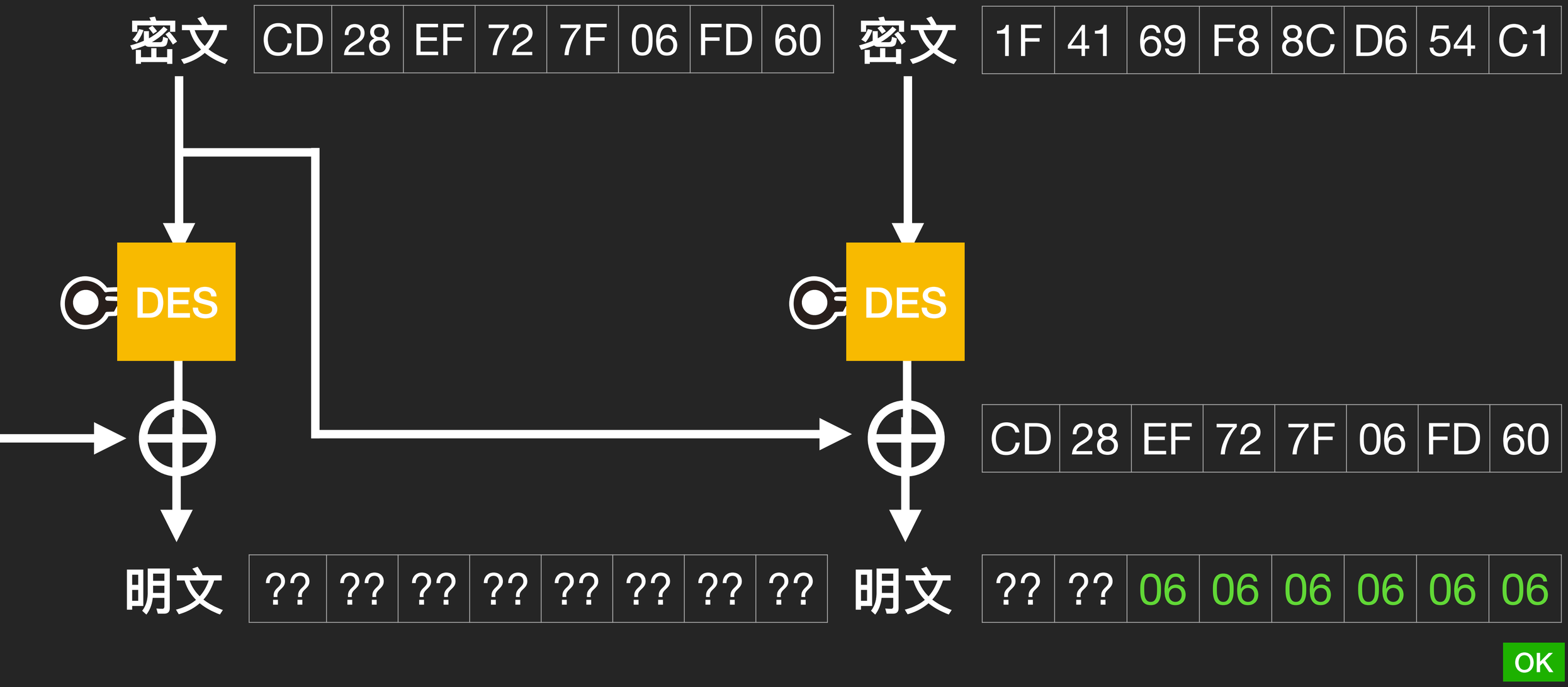
Padding[-1] = 6

Padding[-1] ⊕ 7 => 1

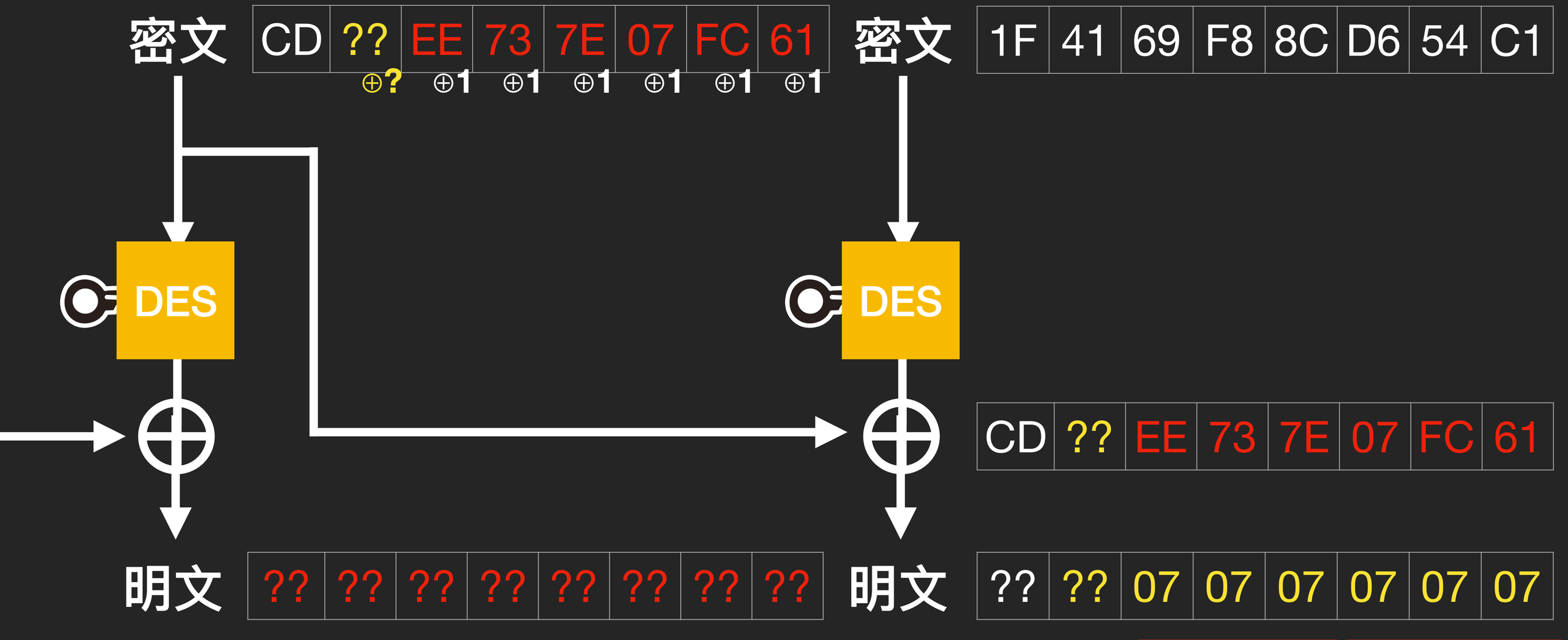
Padding[-1] = 6

Padding = 666666

Padding Oracle Attack



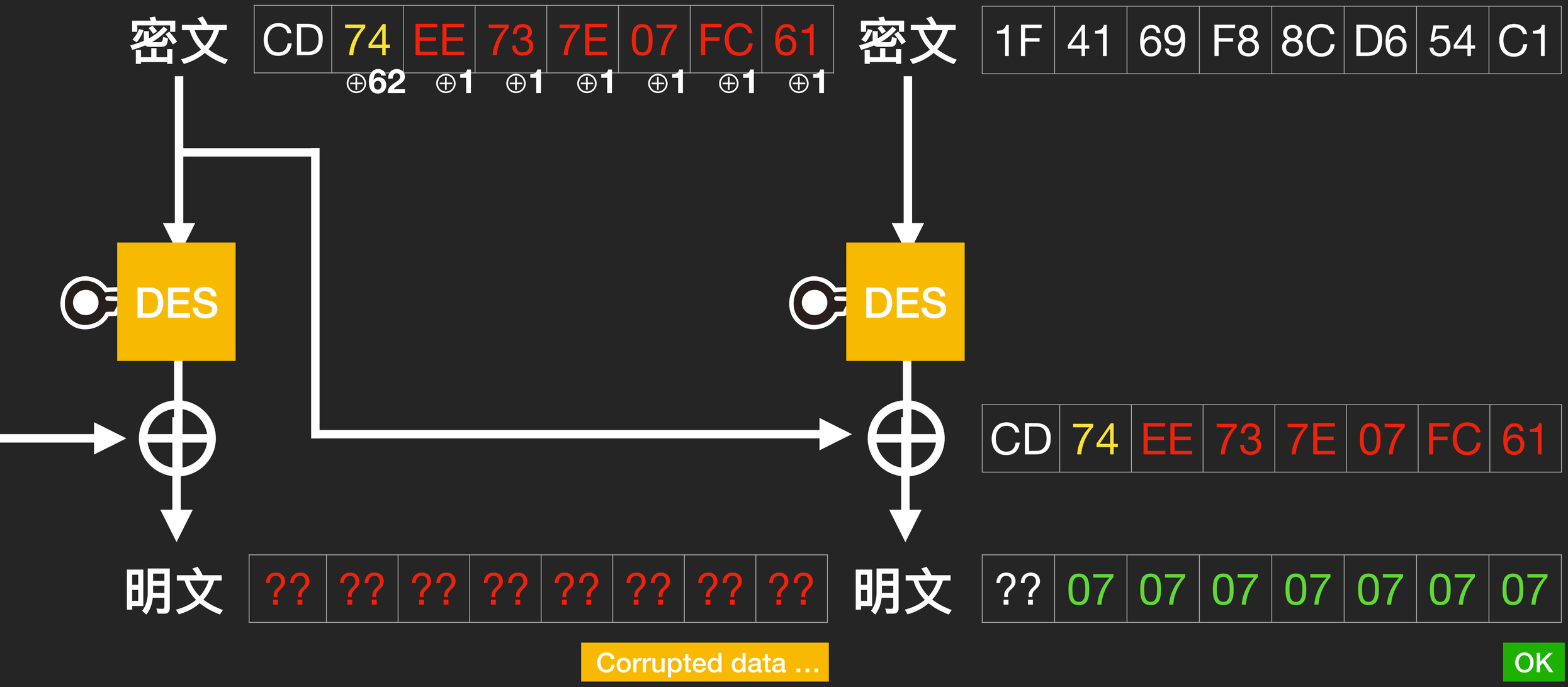
Padding Oracle Attack



Decrypt Fail …

Invalid Padding

Padding Oracle Attack



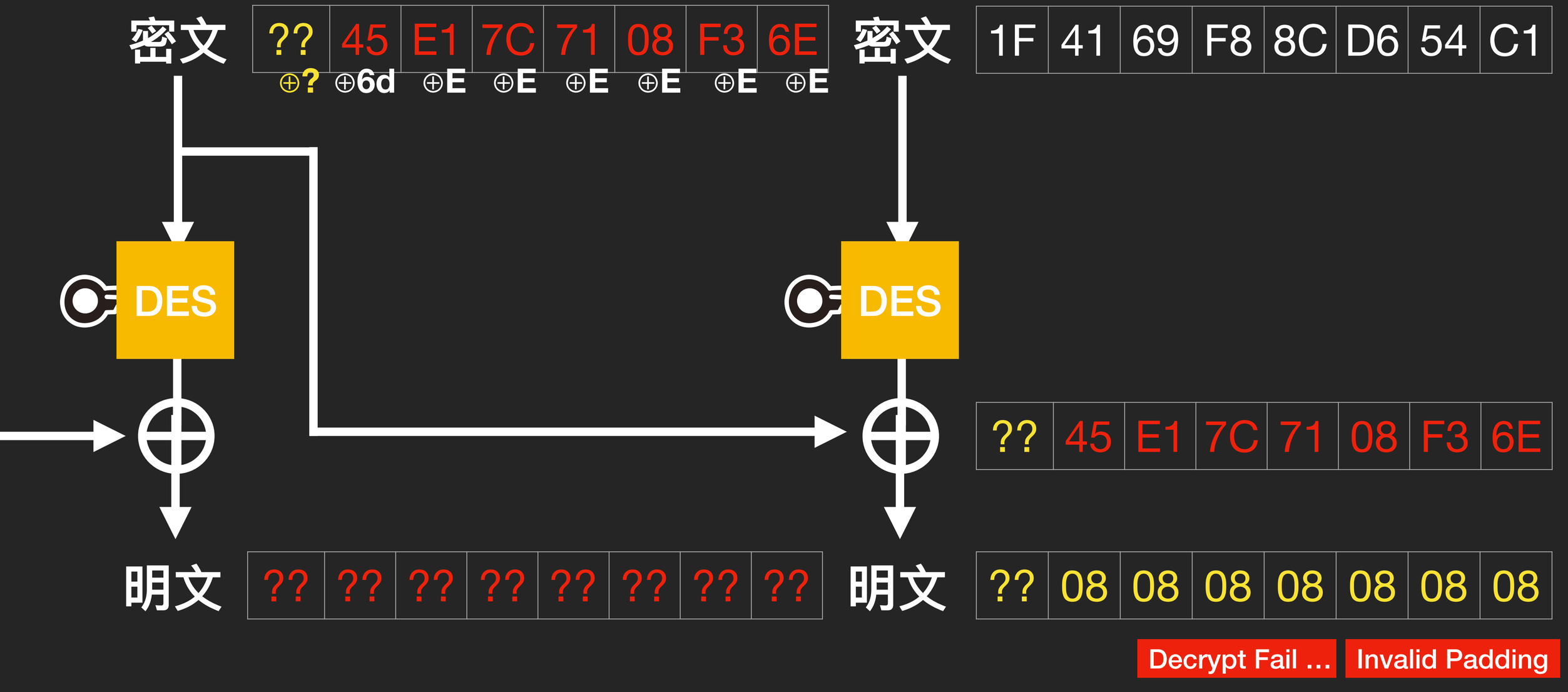
Plaintxt[-1] ⊕ 0x62 => 7

Plaintxt[-1] = 0x62 ⊕ 7

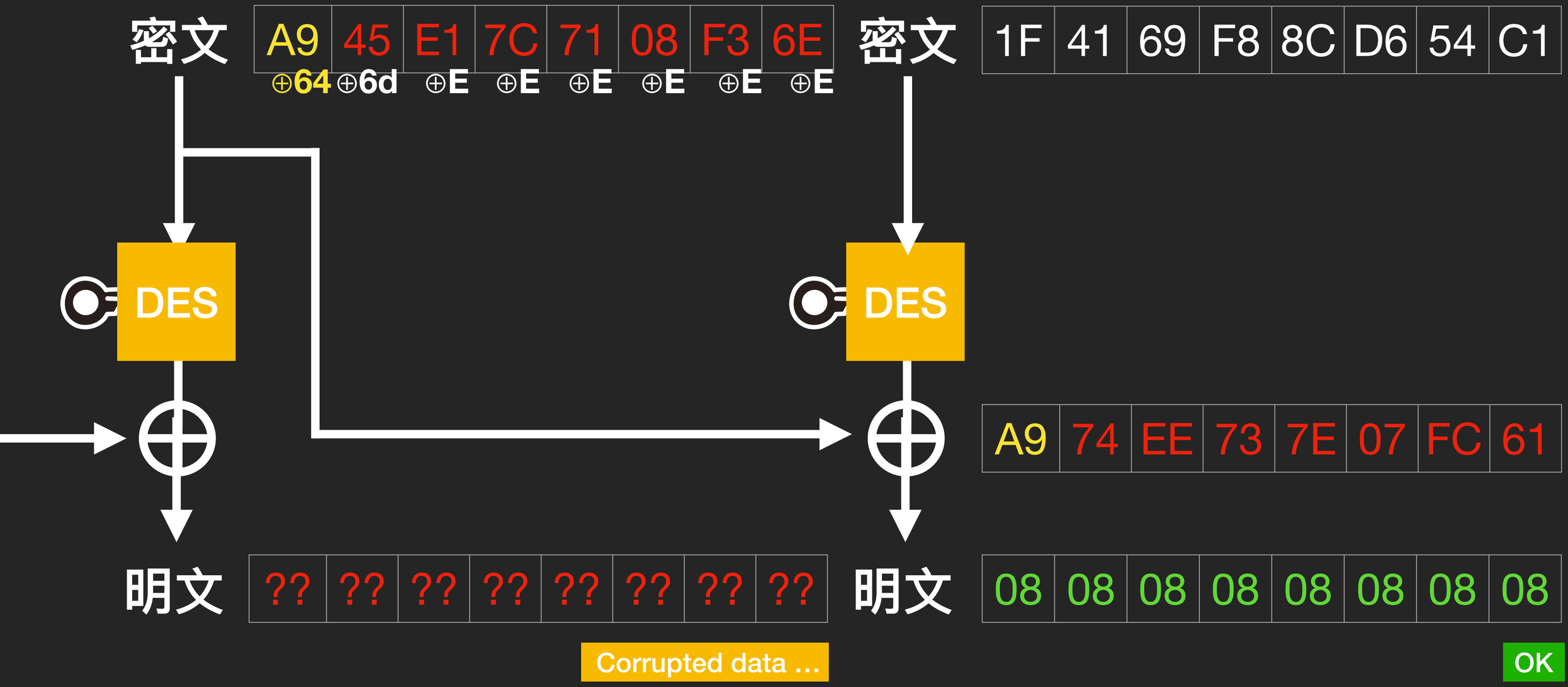
Plaintxt[-1] = ‘e’

Next : 0x65 ⊕ 8 => 0x6d

Padding Oracle Attack



Padding Oracle Attack



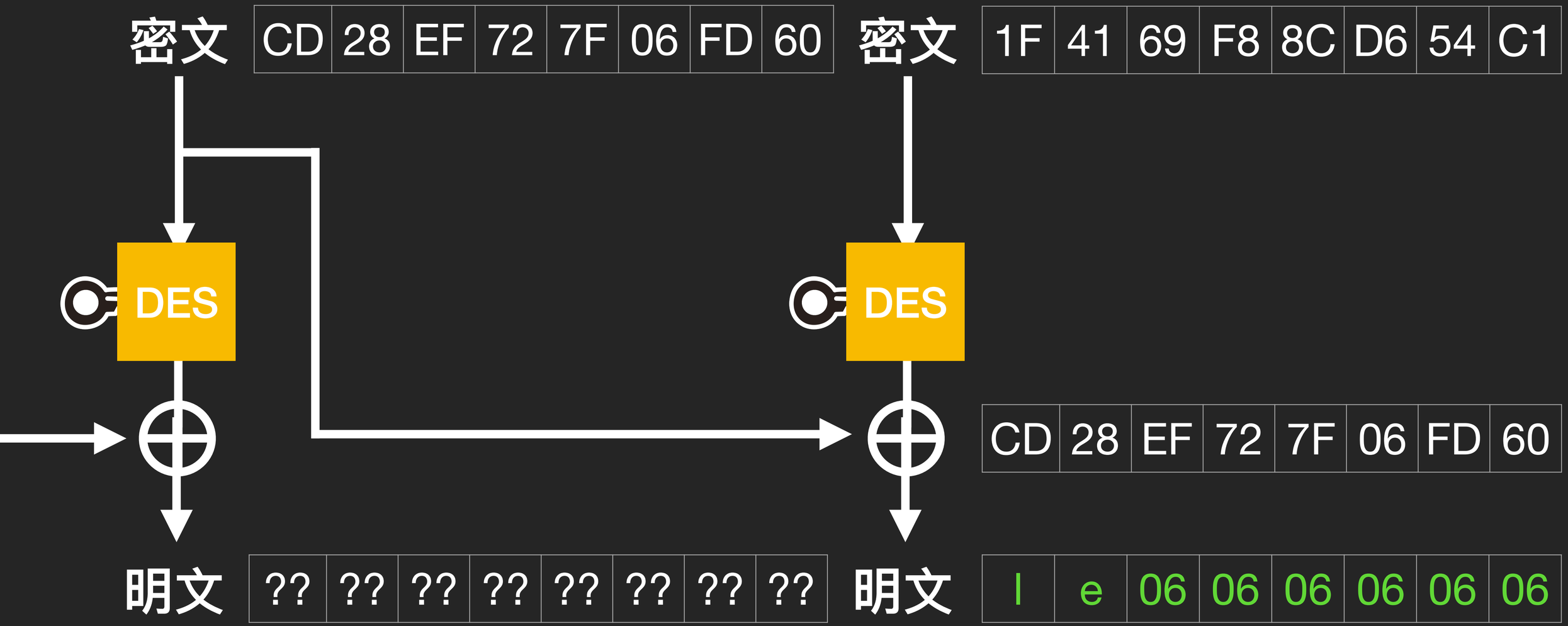
Padding Oracle Attack

密文	1F	41	69	F8	8C	D6	54	C1
	CD	28	EF	72	7F	06	FD	60
明文	08⊕64	08⊕6d	08⊕E	08⊕E	08⊕E	08⊕E	08⊕E	08⊕E

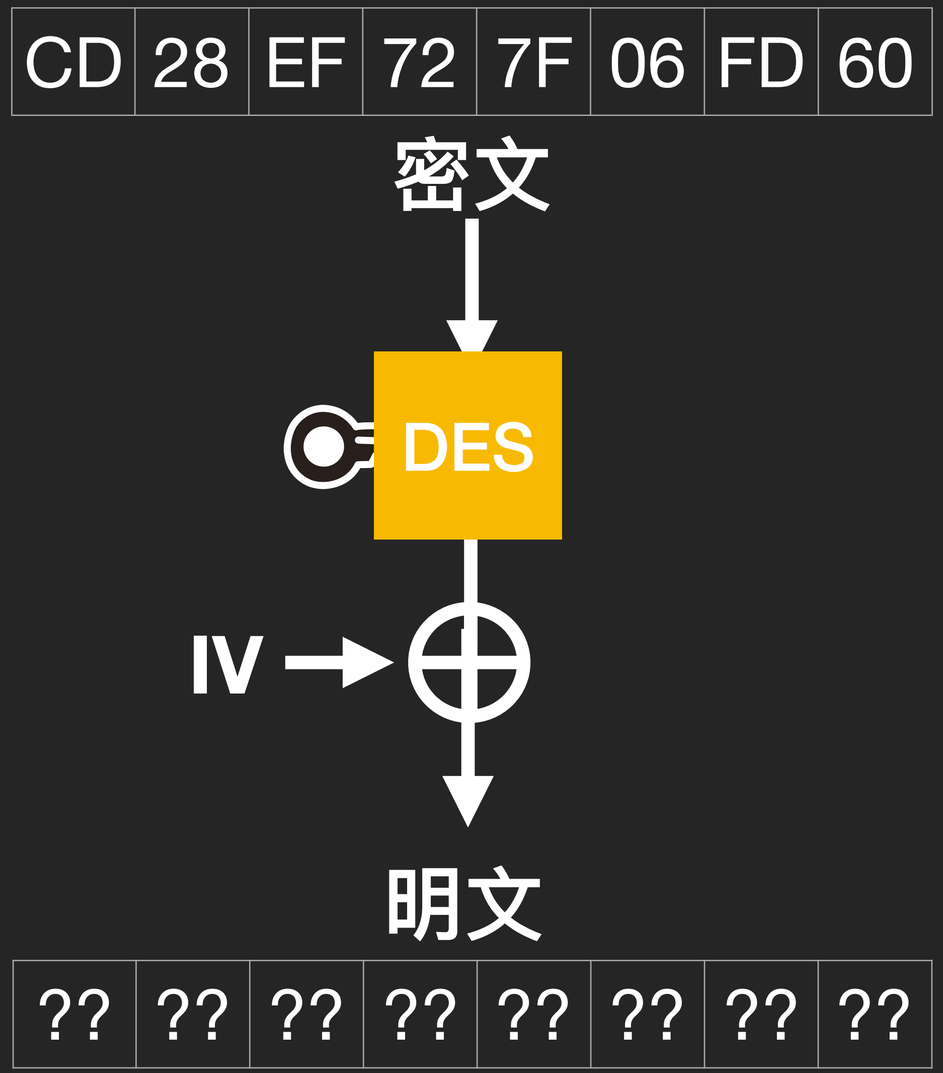
Padding Oracle Attack

密文	1F	41	69	F8	8C	D6	54	C1
	CD	28	EF	72	7F	06	FD	60
明文	I	e	06	06	06	06	06	06

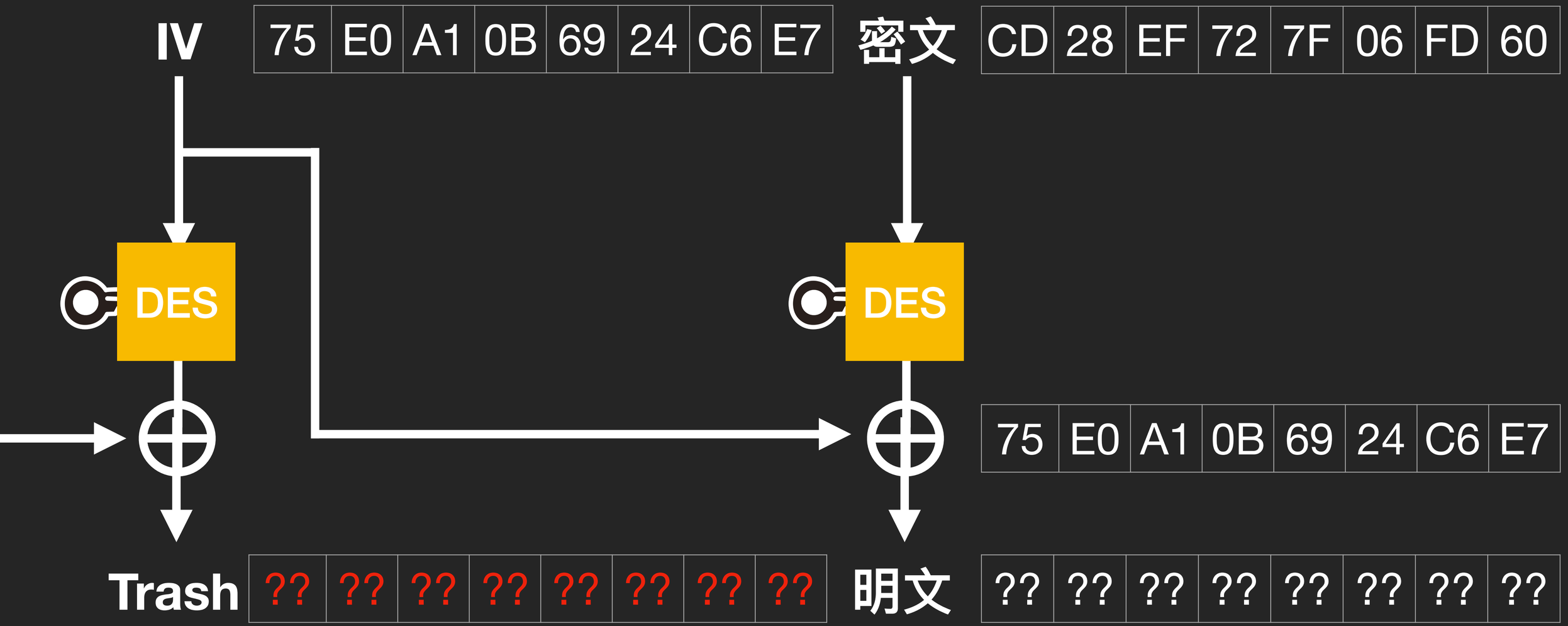
Padding Oracle Attack



Padding Oracle Attack



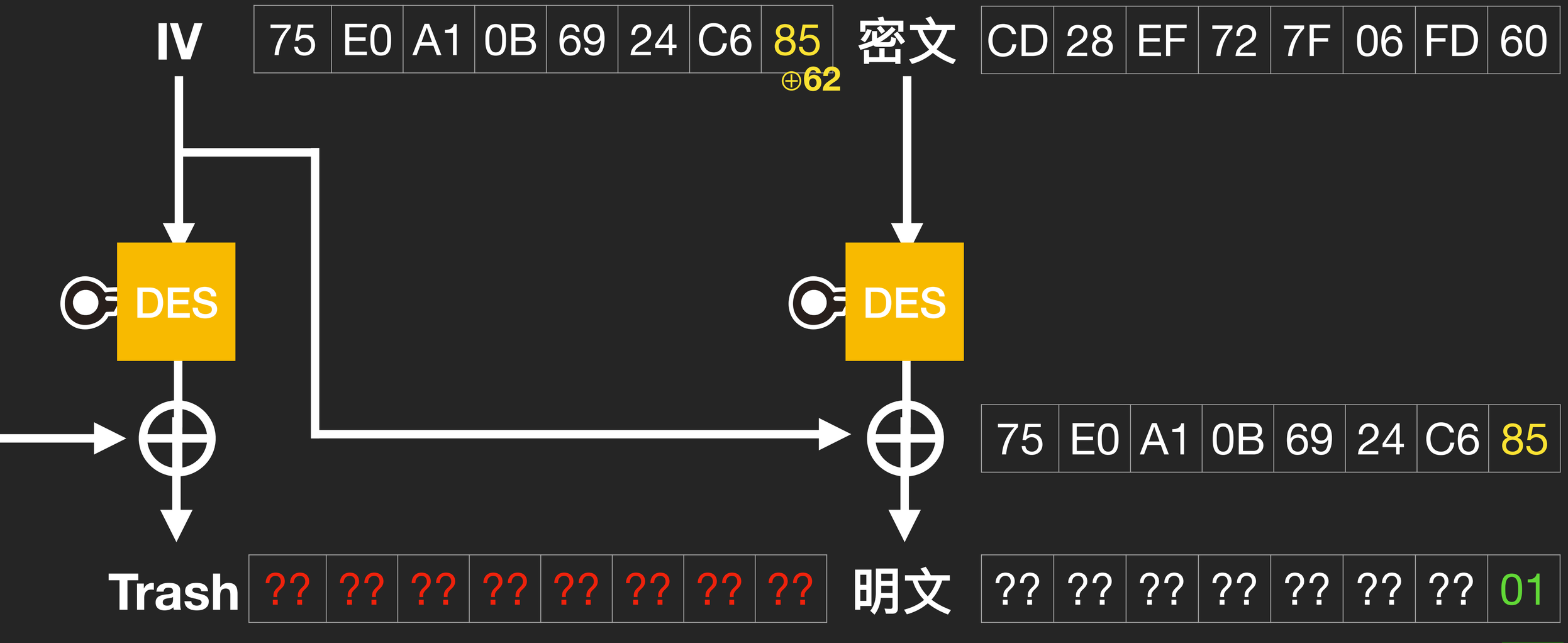
Padding Oracle Attack



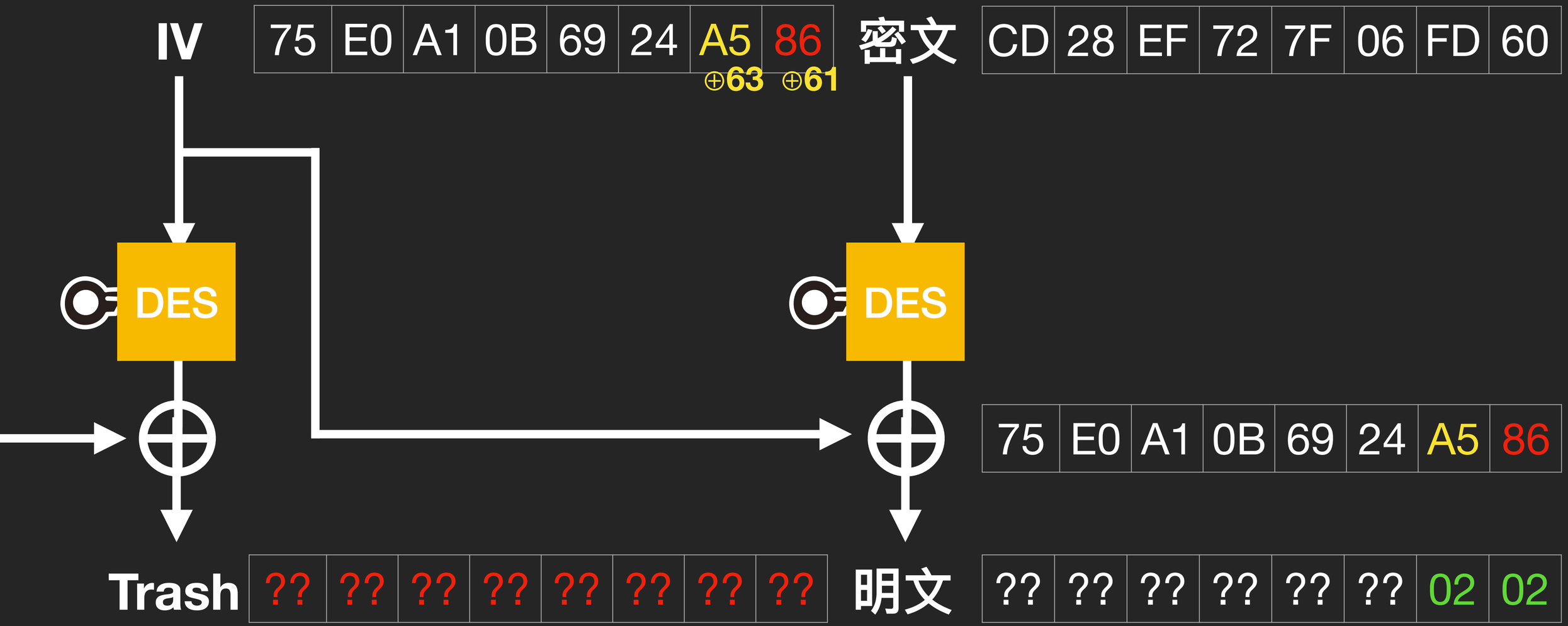
Decrypt Fail …

Invalid Padding

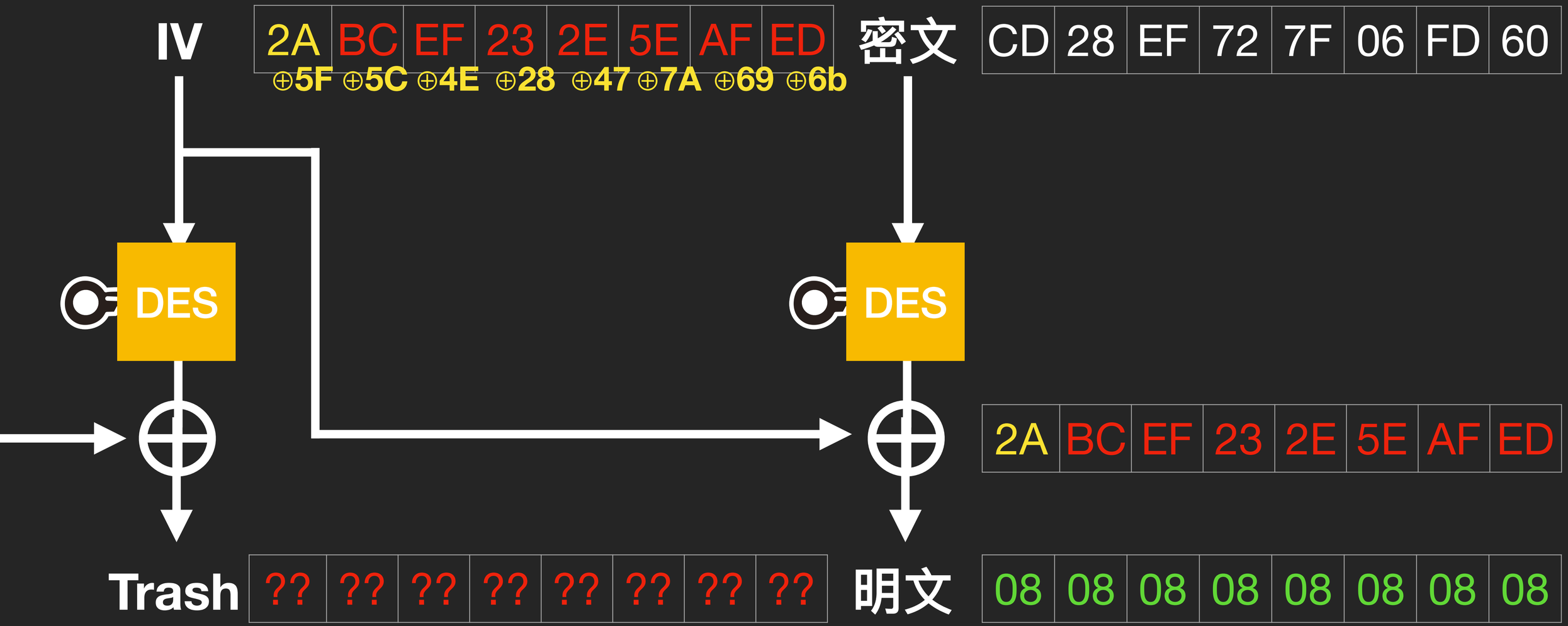
Padding Oracle Attack



Padding Oracle Attack



Padding Oracle Attack



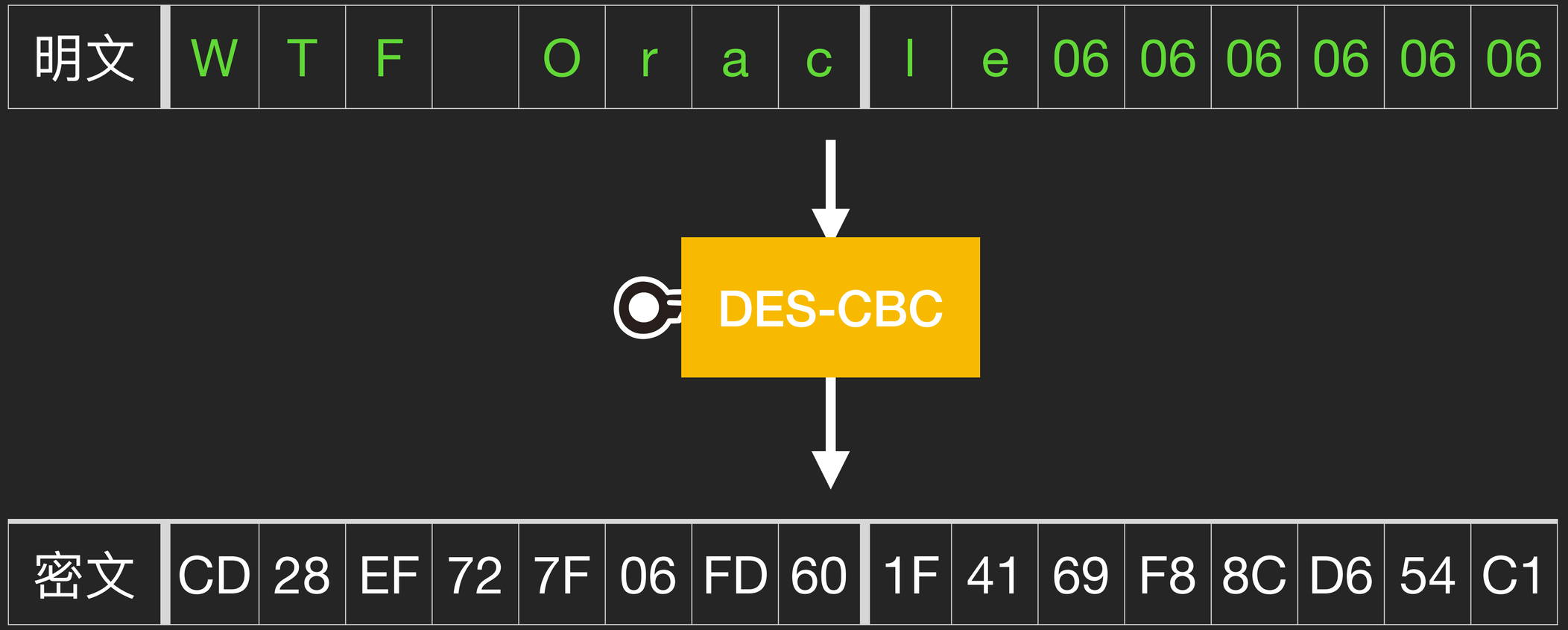
Padding Oracle Attack

密文	CD	28	EF	72	7F	06	FD	60
	75	E0	A1	0B	69	24	C6	E7
明文	08⊕5F	08⊕5C	08⊕4E	08⊕28	08⊕47	08⊕7A	08⊕6B	08⊕69

Padding Oracle Attack

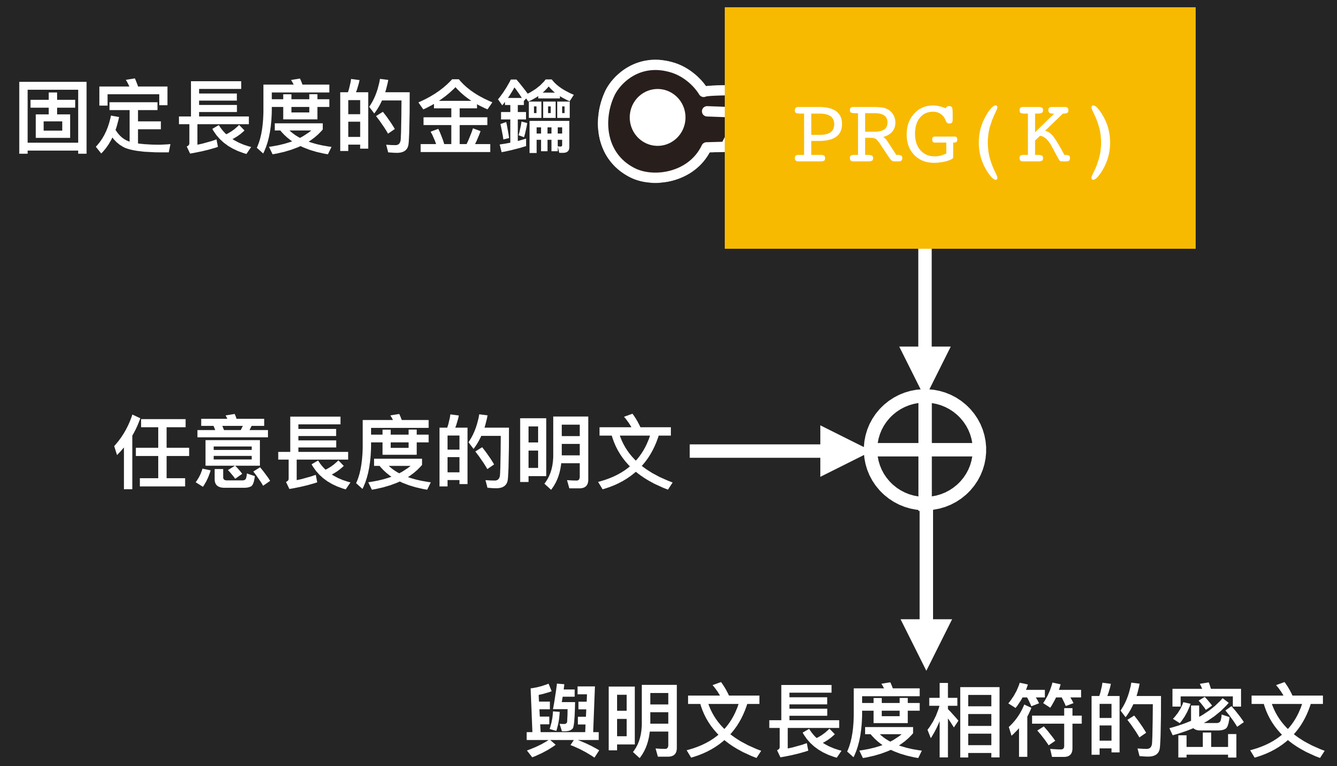
密文	CD	28	EF	72	7F	06	FD	60
	75	E0	A1	0B	69	24	C6	E7
明文	W	T	F		O	r	a	c

Padding Oracle Attack



[0x09] Cathub Party

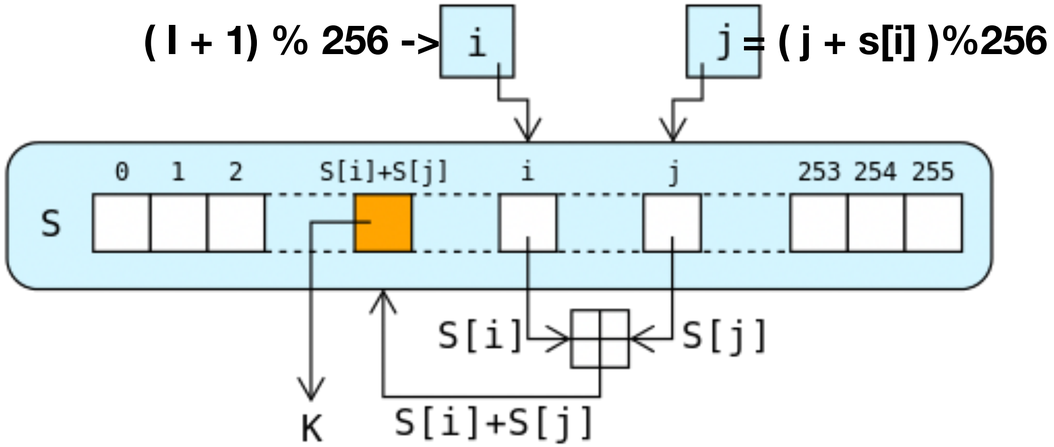
串流密碼 | Stream cipher



RC4 | Rivest Cipher 4

- 又快又簡單
- KSA & PRGA
- 實例：
  WEP, WPA-TKIP, SSL/TLS

PRGA



RC4 | Rivest Cipher 4

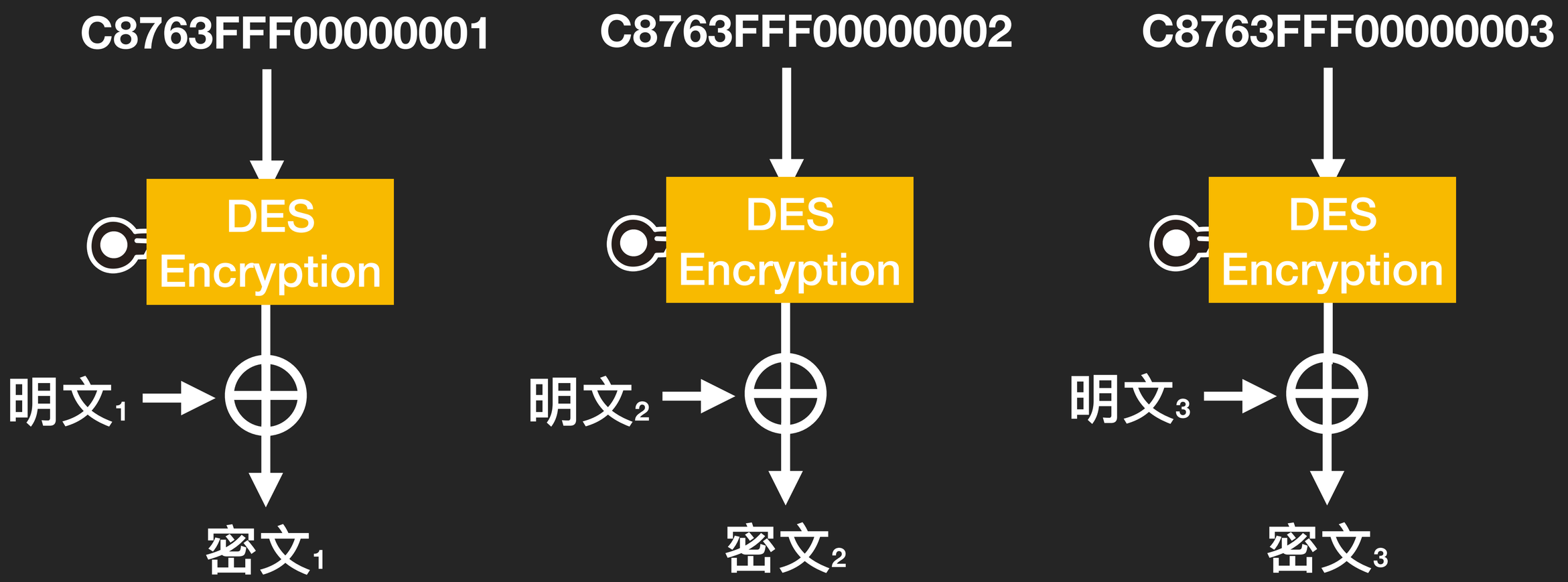
- KSA : Key schedule algorithm

```
def KSA(key):
    S = list(range(256))
    j = 0
    for i in range(256):
        j = (j + S[i] + key[i % len(key)]) % 256
        S[i], S[j] = S[j], S[i]
    return S
```

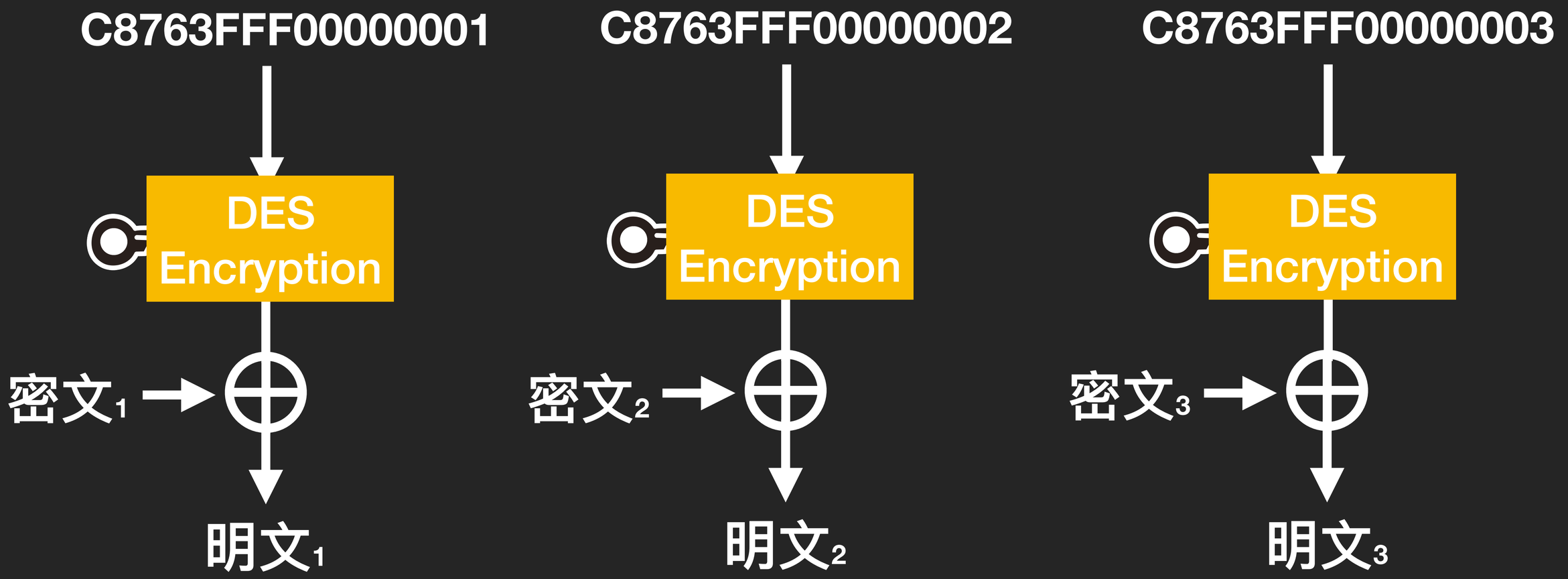
CTR | Counter

- 由 Block cipher 變形而來
- Nonce 相當於半個 Blocksize 的 IV
- Nonce 不可重複使用

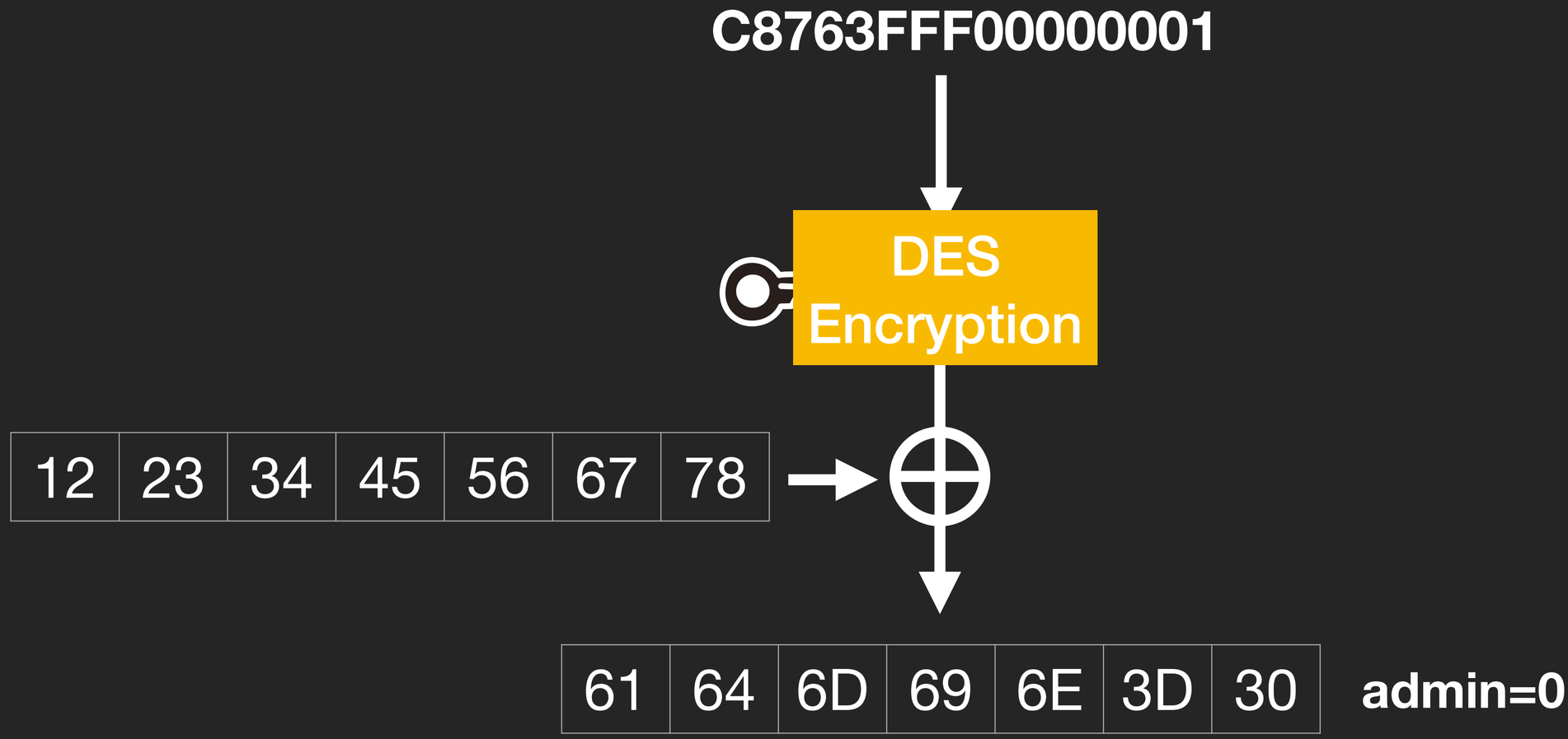
CTR | Counter


C8763FFF00000001
DES Encryption
明文1
密文1
C8763FFF00000002
DES Encryption
明文2
密文2
C8763FFF00000003
DES Encryption
明文3
密文3

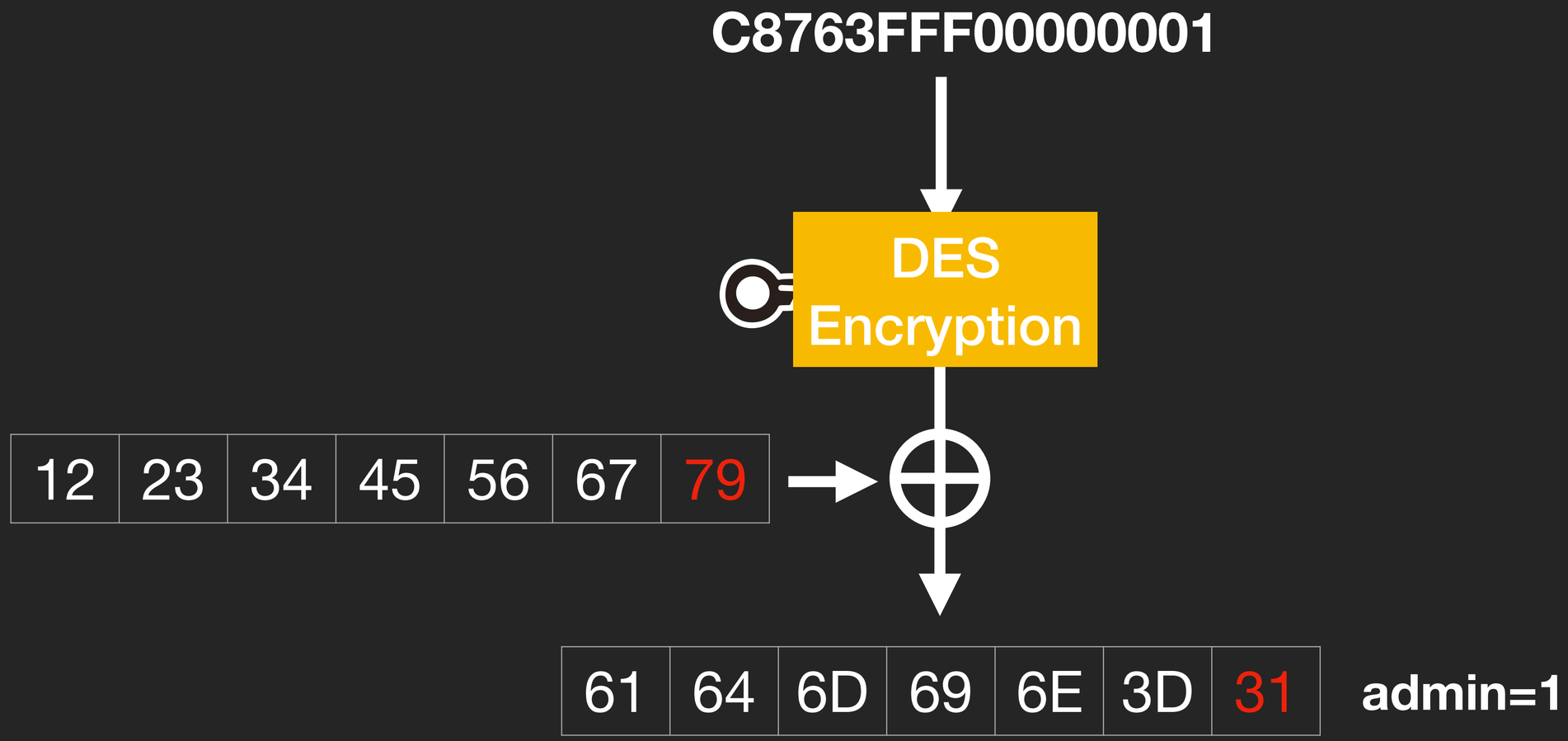
CTR | Counter


C8763FFF00000001
DES Encryption
密文1
明文1
C8763FFF00000002
DES Encryption
密文2
明文2
C8763FFF00000003
DES Encryption
密文3
明文3

位元翻轉 | Bit Flip



位元翻轉 | Bit Flip



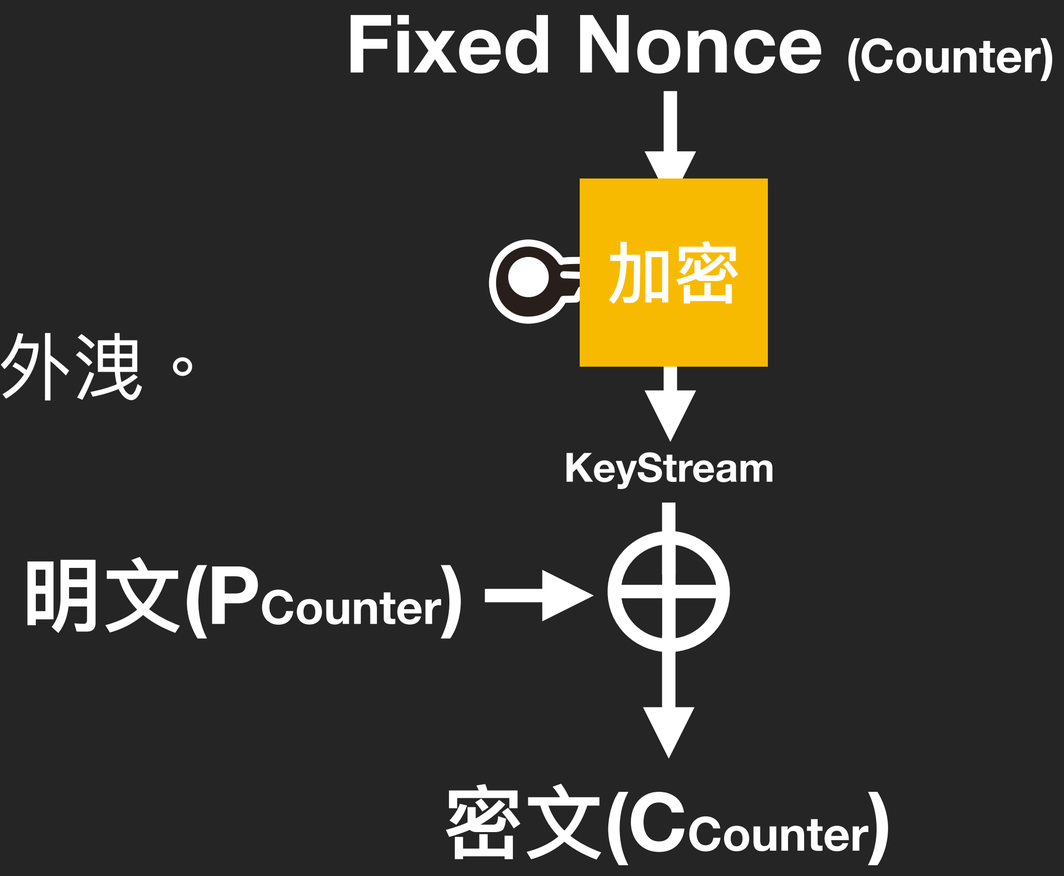
Nonce Reuse & Counter Reset

Nonce 應隨機生成

重複使用 Nonce 會導致 Key Stream 外洩。

$KS_1 = PA_1 \oplus CA_1$

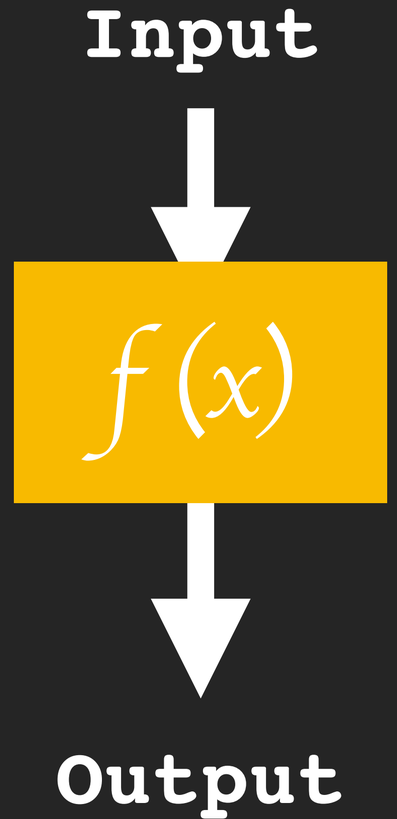
$PB_1 = KS_1 \oplus CB_1$



Hash

單向函數 | One-way compression function

- 可以很容易的算出結果
- 但是很難從結果回推給定的 `Input`



雜湊函數 | Hash function

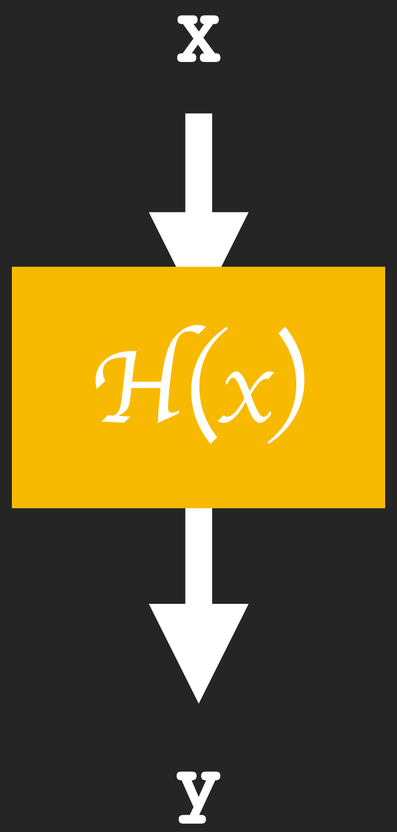
- 功能：將資料壓縮成摘要、保護資料、確保傳遞真實的資訊
- 特徵：固定長度的輸出
- 期望：符合單向函數、$\forall y, \exists! x : h(x) = y$

雜湊函數 | Hash function

- 字串總和 (len)
- Bytes 總和
- Bit 有幾個 1
- LCG

密碼雜湊函數 | Cryptographic Hash function

- Pre-image resistance
  - 已知 y 也無法找到 x
- Second pre-image resistance
  - 已知 y = H(x_1) 無法找到 x_2 使得 y == H(x_2)
- Collision resistance
  - 無法找到一組 x_1、x_2 使得 H(x_1) 和 H(x_2) 有相同的雜湊值
- Avalanche effect
  - x_1、x_2 只有微小的差別，但 y_1、y_2 卻差很多



密碼雜湊函數 | Cryptographic Hash function

Kaibro	→	`8c45dd86e3659040ecdc36b007a99a6e907d2fcd4d2d80142f424912`
KaiBro	→	`1df7b804e1af7caa5d9959a81727a905658d19ba6719cc6b761d5c0c`
Kaibr0	→	`de18feec5e15a615203941bab08edbbedcbdbc91dd8c3885cbdf43dd`
???	→	`8d93f6f29f0cd61bd6f7dbc975a7e7ee096841ab1dfcc90f082f6e53`

碰撞 | Collision

- H(M1) = H(M2)

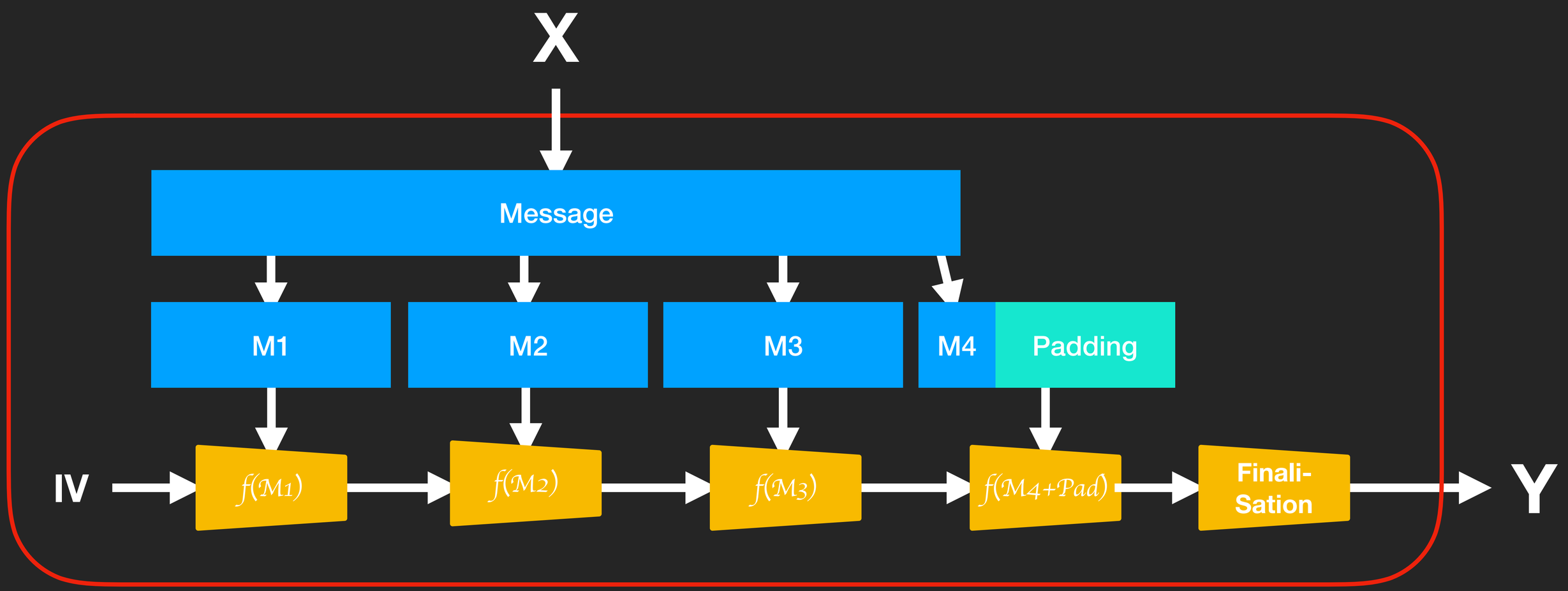
  → H(M1 || M3) = H(M2 || M3)

- Shattered: SHA1 collision blocks in PDF

Merkle–Damgård construction

- Fixed Input → Variable Input
- 該結構可以讓碰撞降低 ... 等

Merkle–Damgård construction



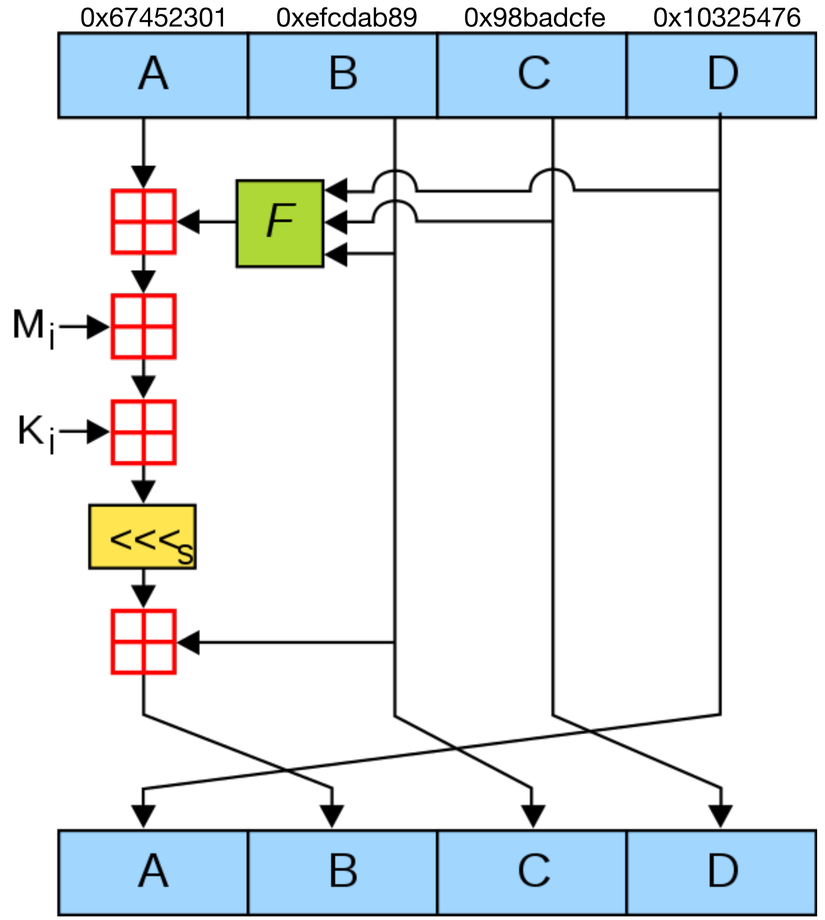
MD5, SHA1, SHA2

MD5

長度：128 bits

特徵：用到 Sin(x)

- ✓ Pre-image resistance
- ✓ Second Pre-image resistance
- ✗ Collision resistance : 2^{18}太小

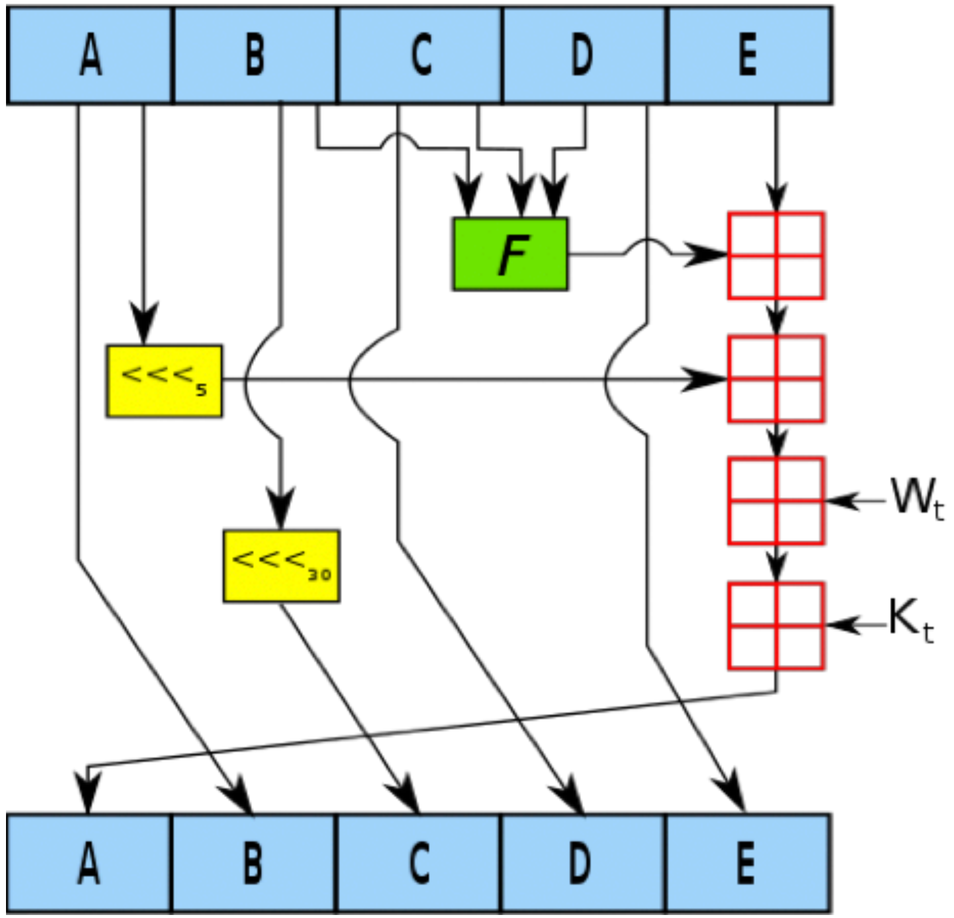


SHA1

長度：160 bits

特徵：0x428a2f98

- ✓ Pre-image resistance
- ✓ Second Pre-image resistance
- ✗ Collision resistance : 2^{60} 不夠大

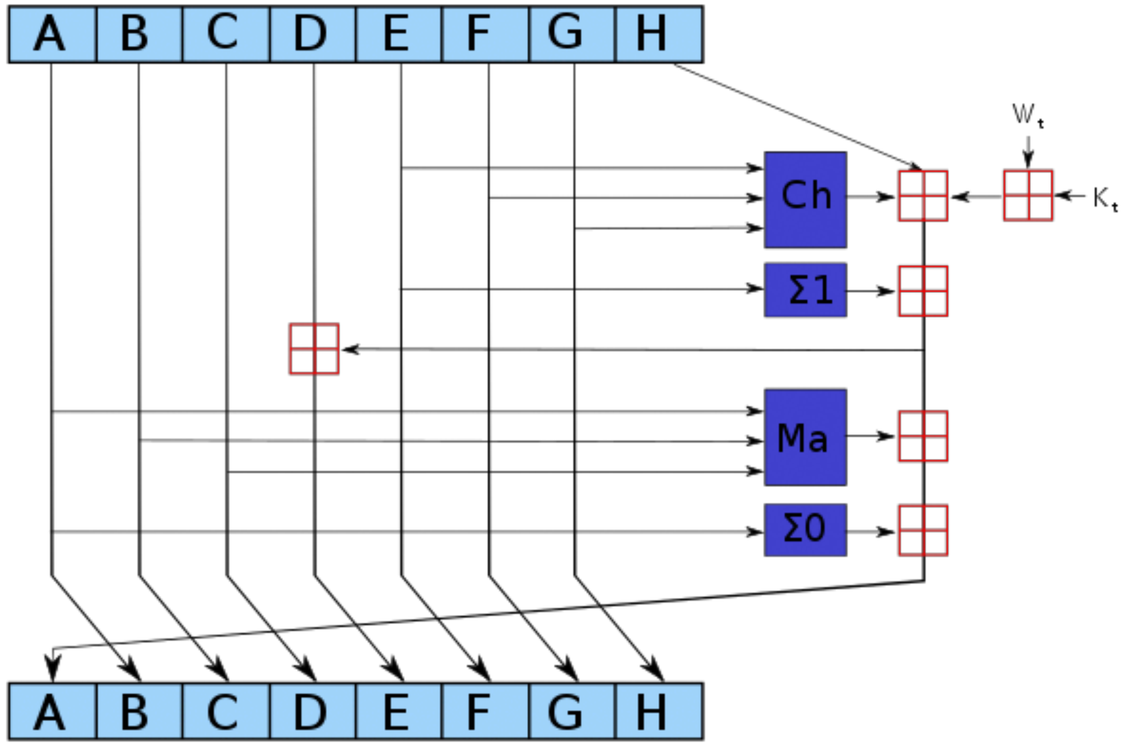


SHA2

長度：224, 256, 384, 512 bit

特徵：Constant 0x428a2f98

- ✓ Pre-image resistance
- ✓ Second Pre-image resistance
- ✓ Collision resistance

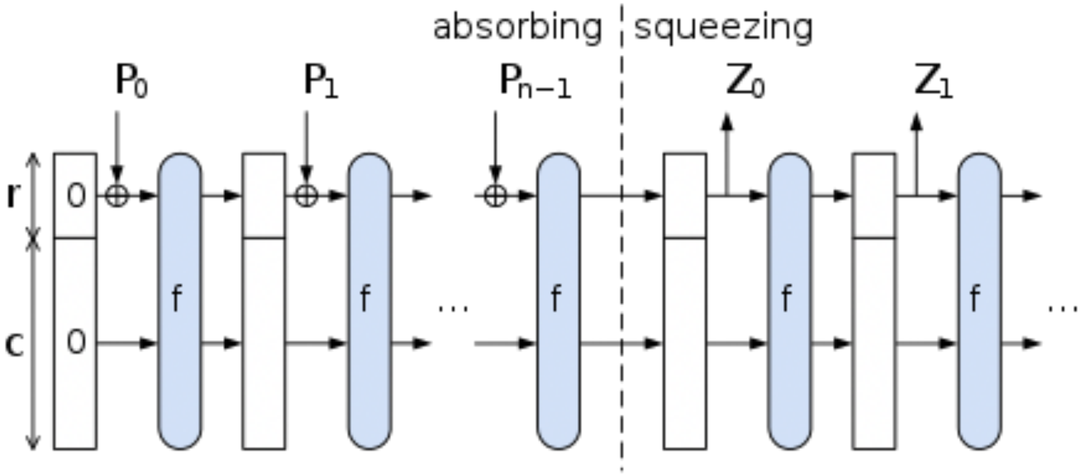


SHA3

長度：224, 256, 384, 512 bit

特徵：0x8000000080008081

- ✓ Pre-image resistance
- ✓ Second Pre-image resistance
- ✓ Collision resistance

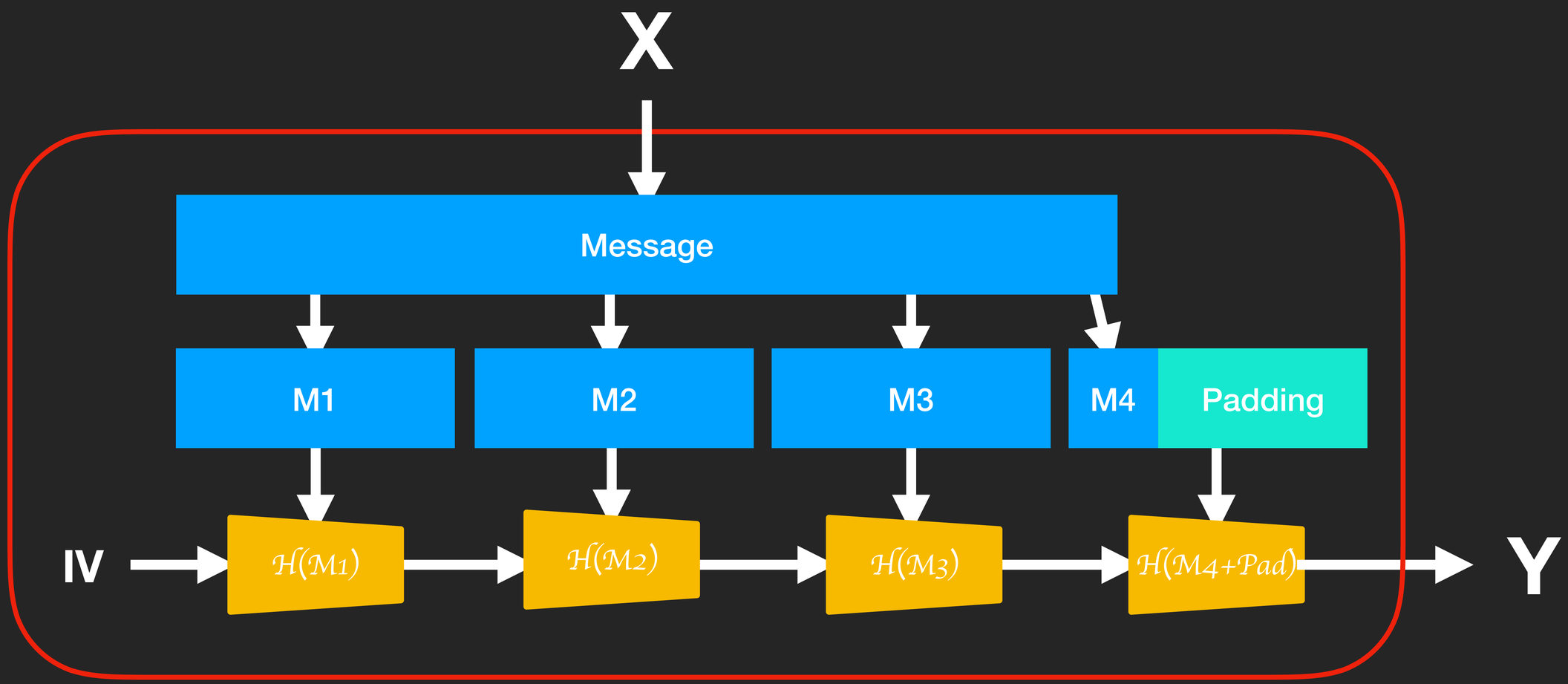


Message authentication code

MAC | Message authentication code

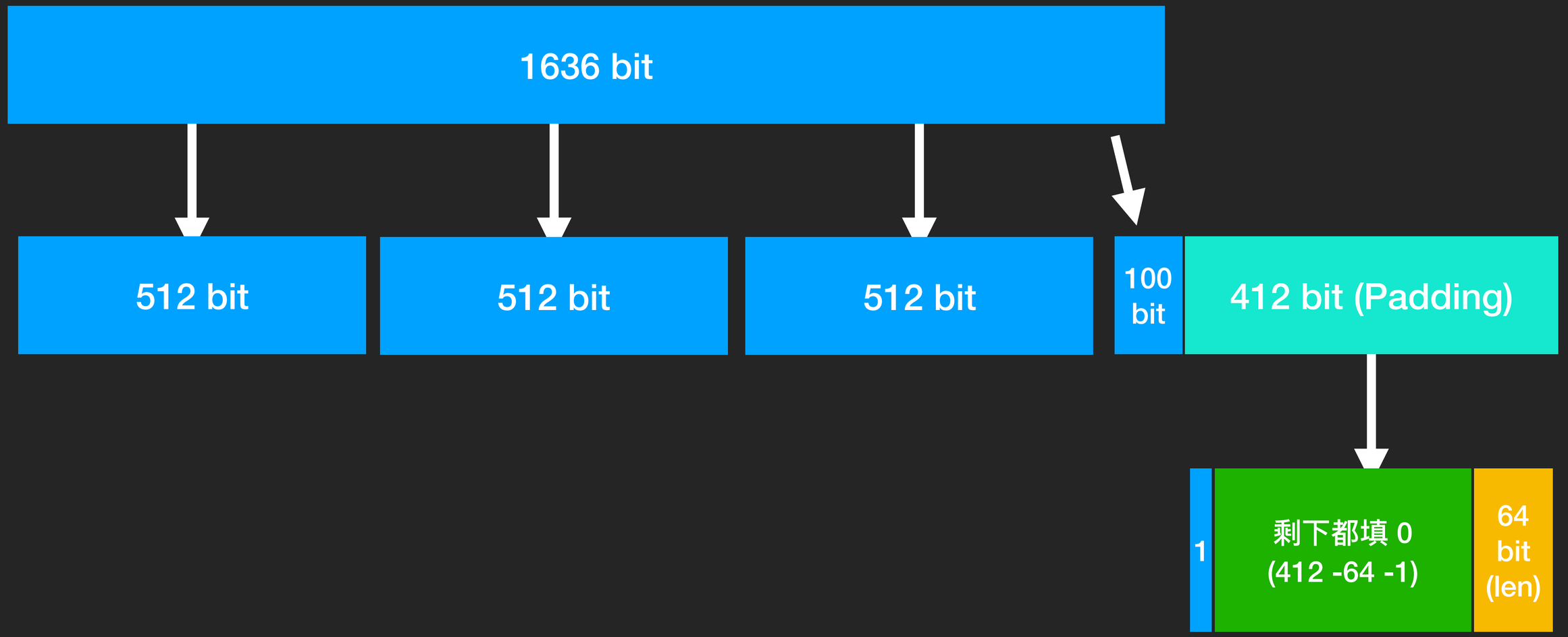
- `MAC(M) = H(M || K) : collision`
- `MAC(M) = H(k || M) : Length extension attack`

長度擴充攻擊 | Length extension attack

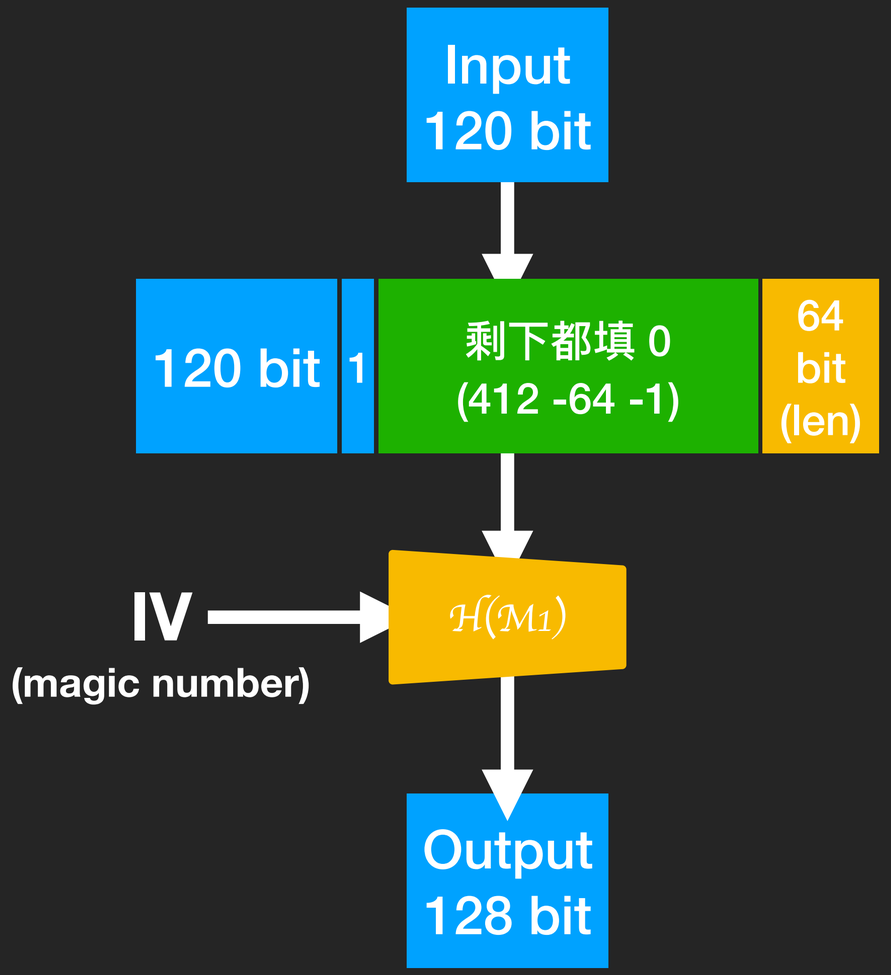


MD5, SHA1, SHA2

填充 | Padding


1636 bit
512 bit
512 bit
512 bit
100 bit
412 bit (Padding)
1
剩下都填 0
(412 -64 -1)
64 bit (len)

長度擴充攻擊 | Length extension attack

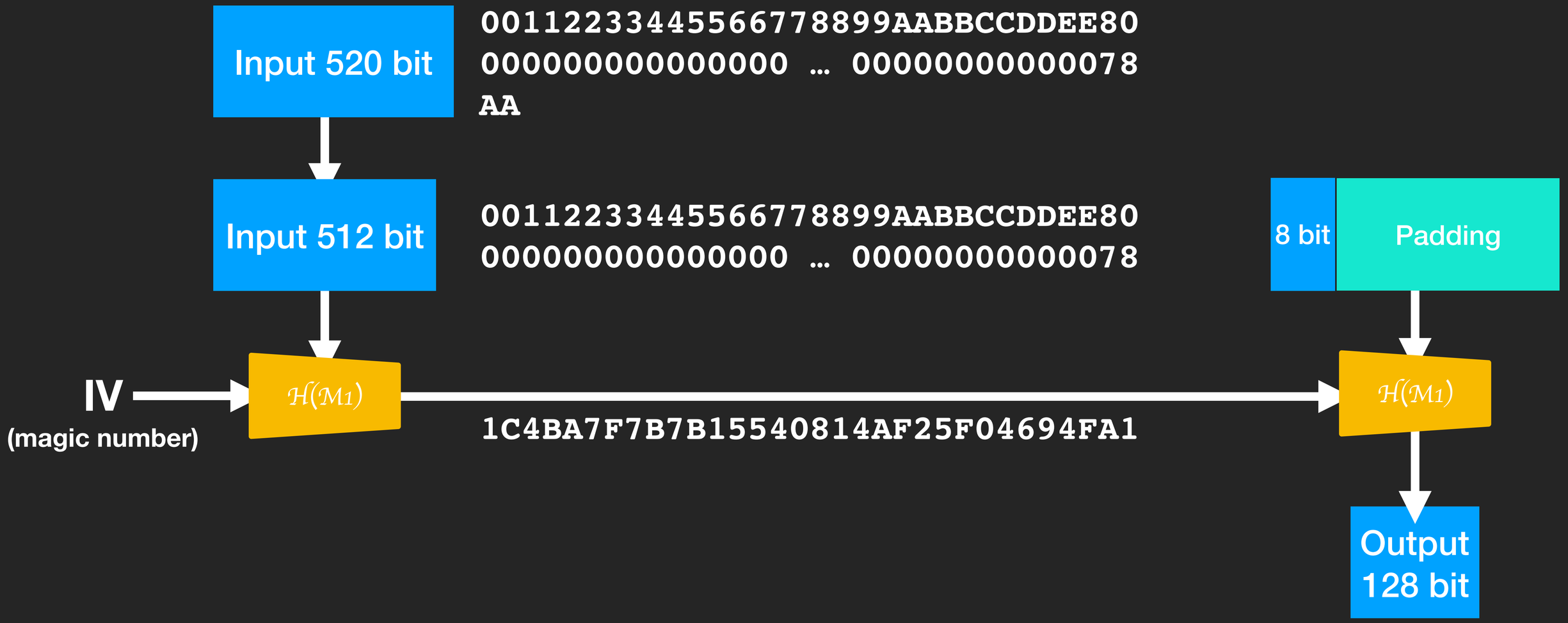



```
00112233445566778899AABBCCDDEE
```

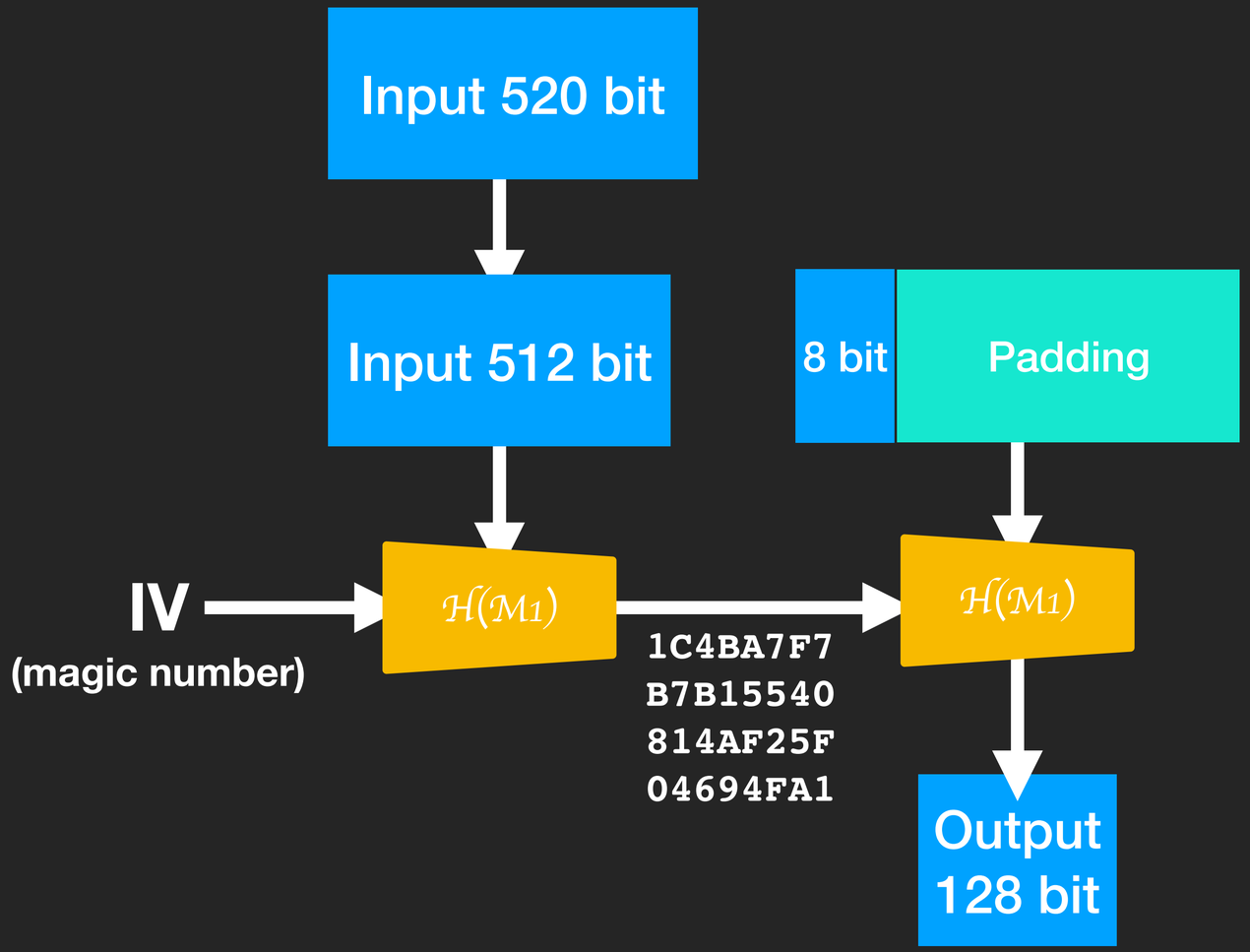
```
00112233445566778899AABBCCDDEE80
000000000000000 … 0000000000078
```

```
1C4BA7F7B7B15540814AF25F04694FA1
```

長度擴充攻擊 | Length extension attack



長度擴充攻擊 | Length extension attack

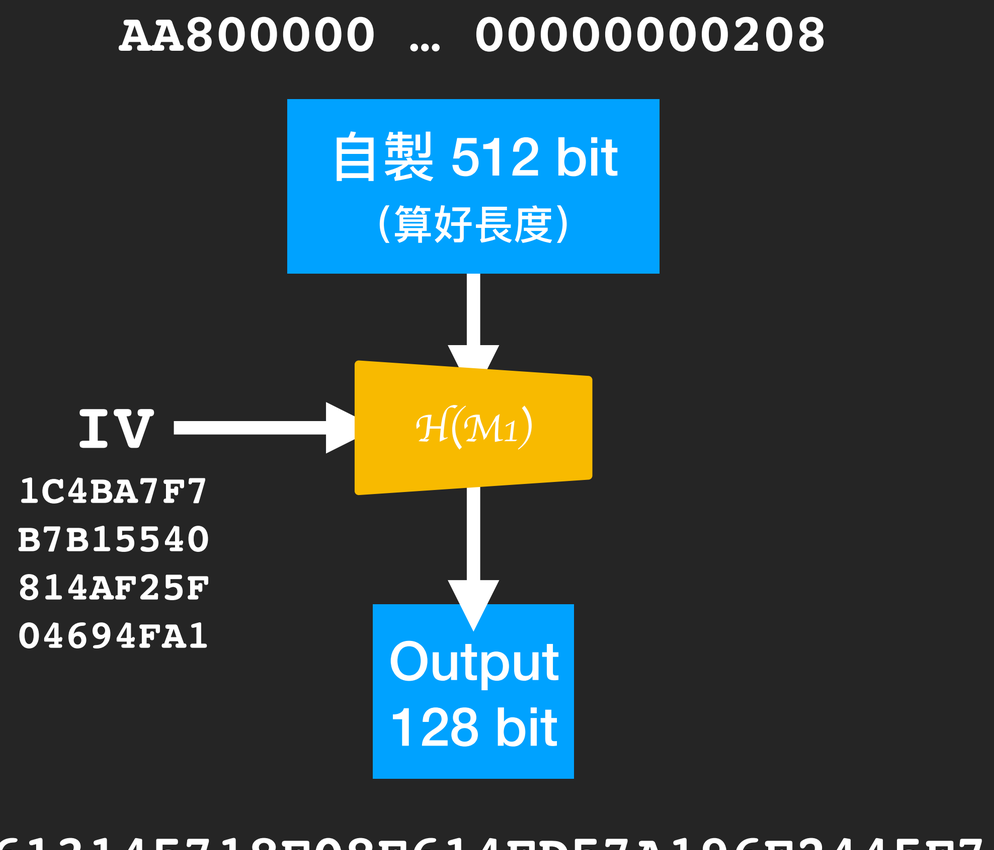


AA800000 … 00000000208

E6131457 18E08F61 4ED57A19 6E2445E7

長度擴充攻擊 | Length extension attack

- 若已知 Hash 值，且可輸入 bytes
- 即可任意加字在後方
- 自行算出其 hash value



HMAC | keyed-hash Message authentication code

- `HMAC(k, M) = H((k' ⊕ opad)||H((k' ⊕ ipad)||m))`