

Block Cipher Mode

by oalieno

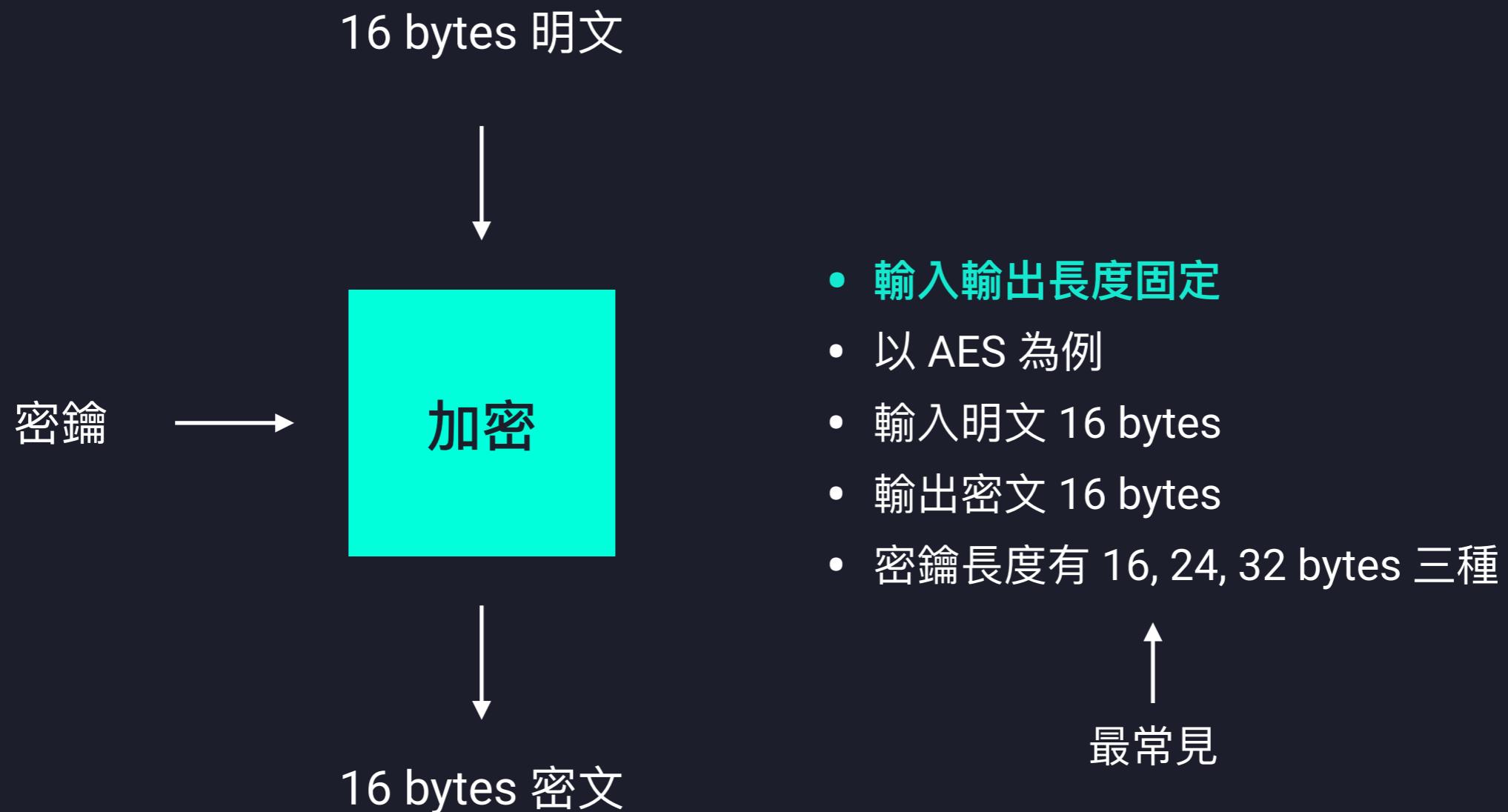


目錄

- Block Cipher
- Block Cipher Mode
 - ECB Mode
 - CBC Mode
 - CTR Mode
 - GCM Mode
- Attack Scenarios
 - ECB Mode / Cut & Paste
 - ECB Mode / Prepend Oracle Attack
 - CBC Mode / Bit-Flipping Attack
 - CBC Mode / Padding Oracle Attack
 - GCM Mode / Forbidden Attack

Block Cipher

Block Cipher

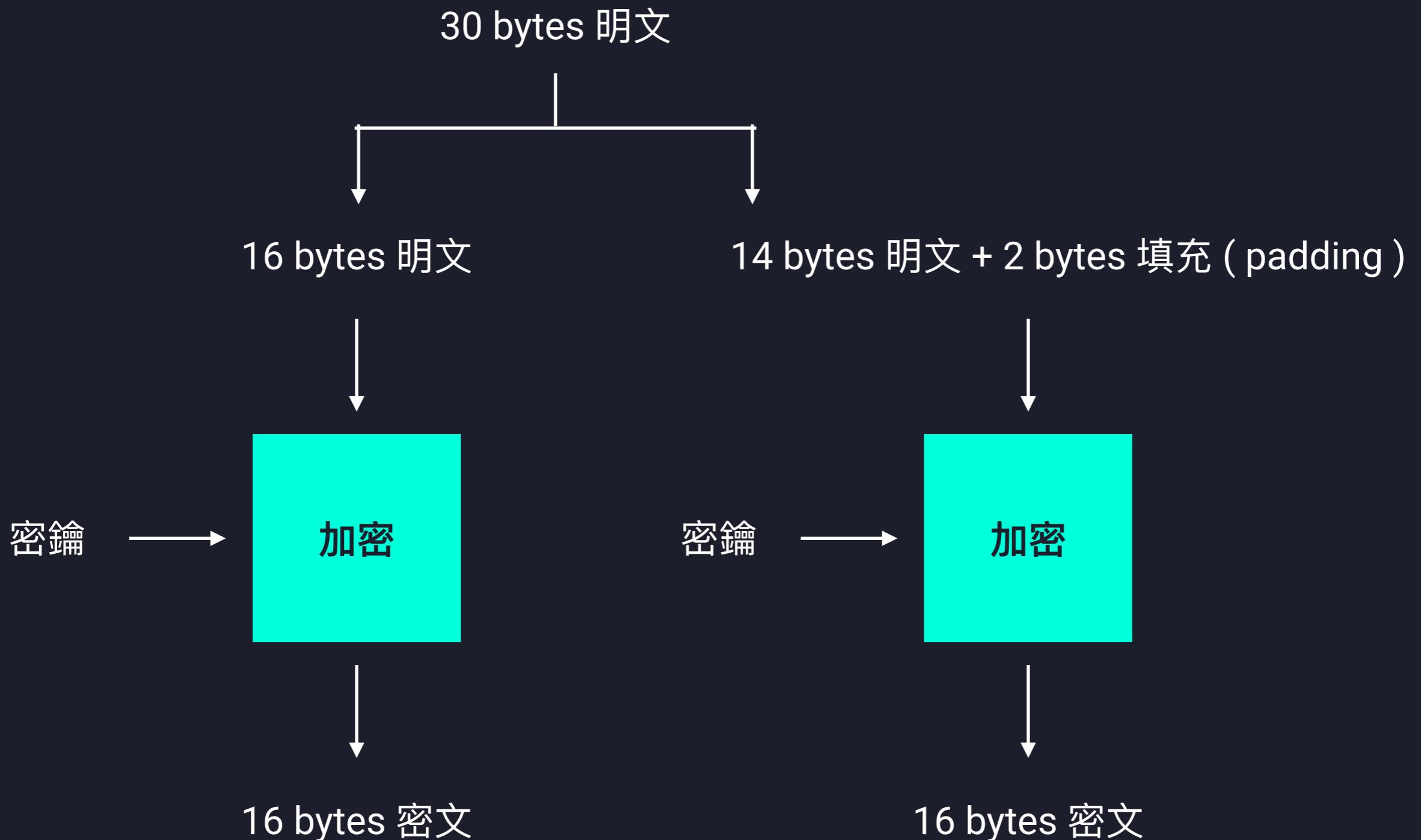


Block Cipher

我要加密的明文不是 16 bytes 怎麼辦？

切成很多個 16 bytes

Block Cipher



Block Cipher Mode

Block Cipher Mode

- AES 是 Block Cipher，輸入輸出長度固定
- 所以要加密任意長度的明文，需要一些額外加工
- 一些常見的加工模式：ECB, CBC, CFB, OFB, CTR...
- 有 AEAD 的模式：CCM, GCM, OCB...

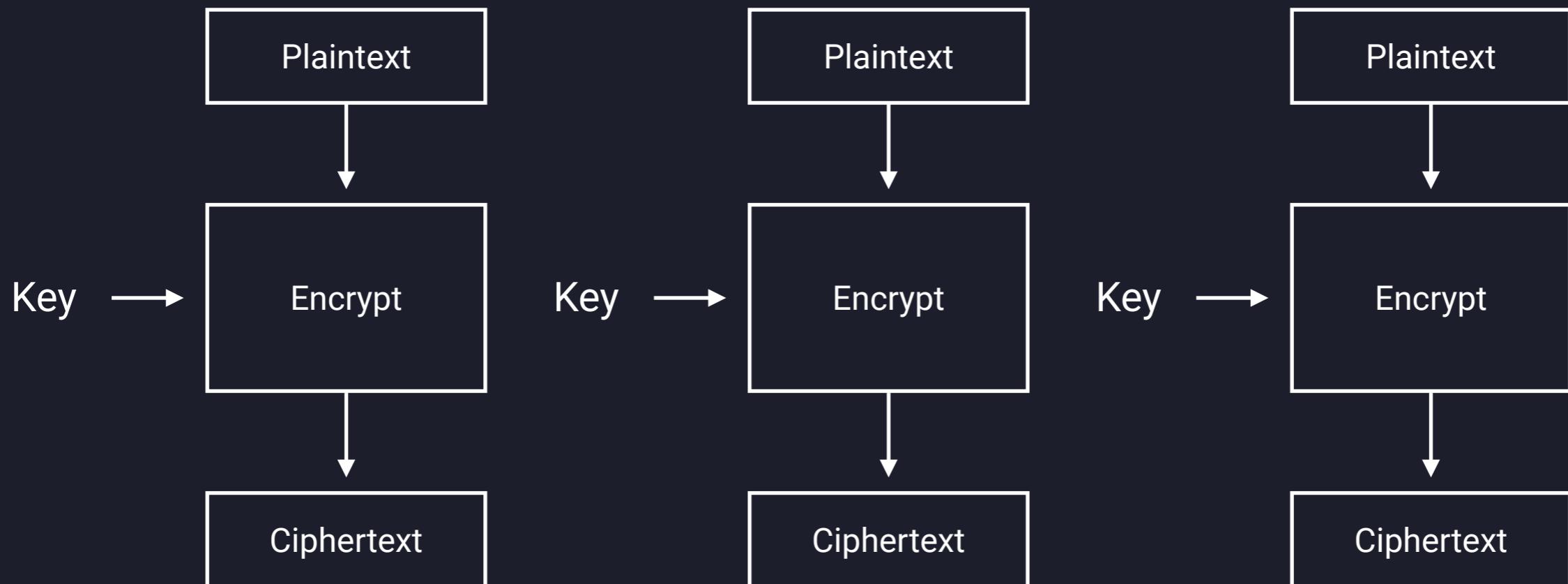
Authenticated **E**ntryption with **A**sociated **D**ata

Authenticated Encryption with Associated Data

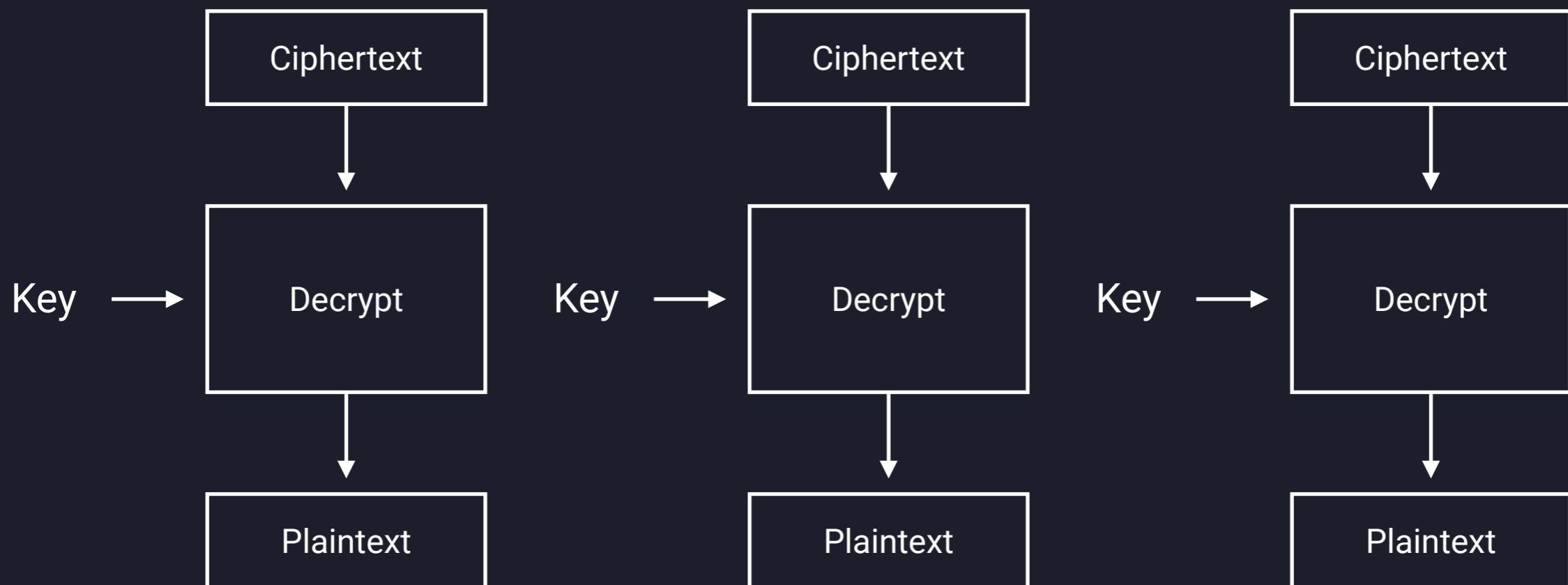
- Authenticated Encryption (AE)
 - 會多計算一個 MAC (Message Authentication Code)
- Associated Data (AD)
 - 可以連同一些沒加密的資料一起做 Authentication
 - 比如一些不需要加密的 Header Information

ECB Mode

ECB Mode Encryption



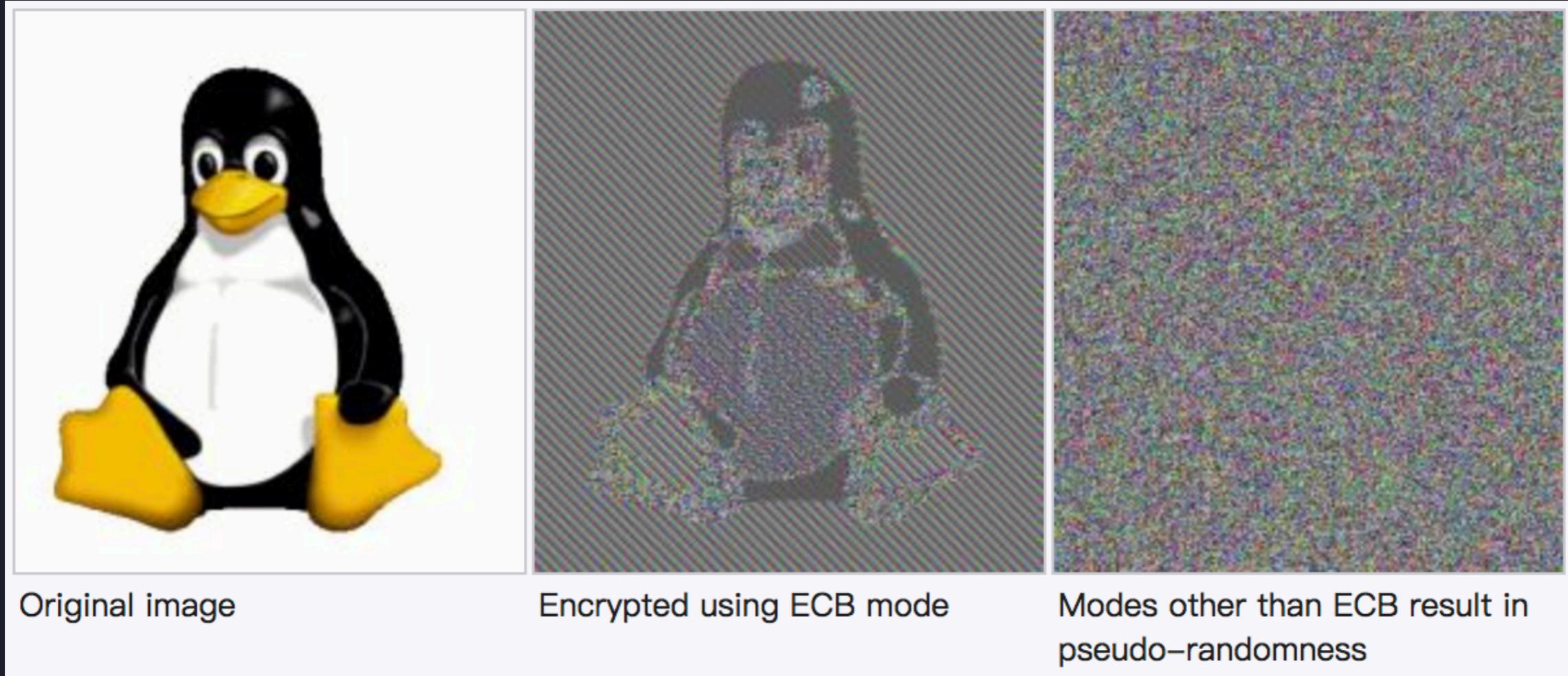
ECB Mode Decryption



ECB Mode

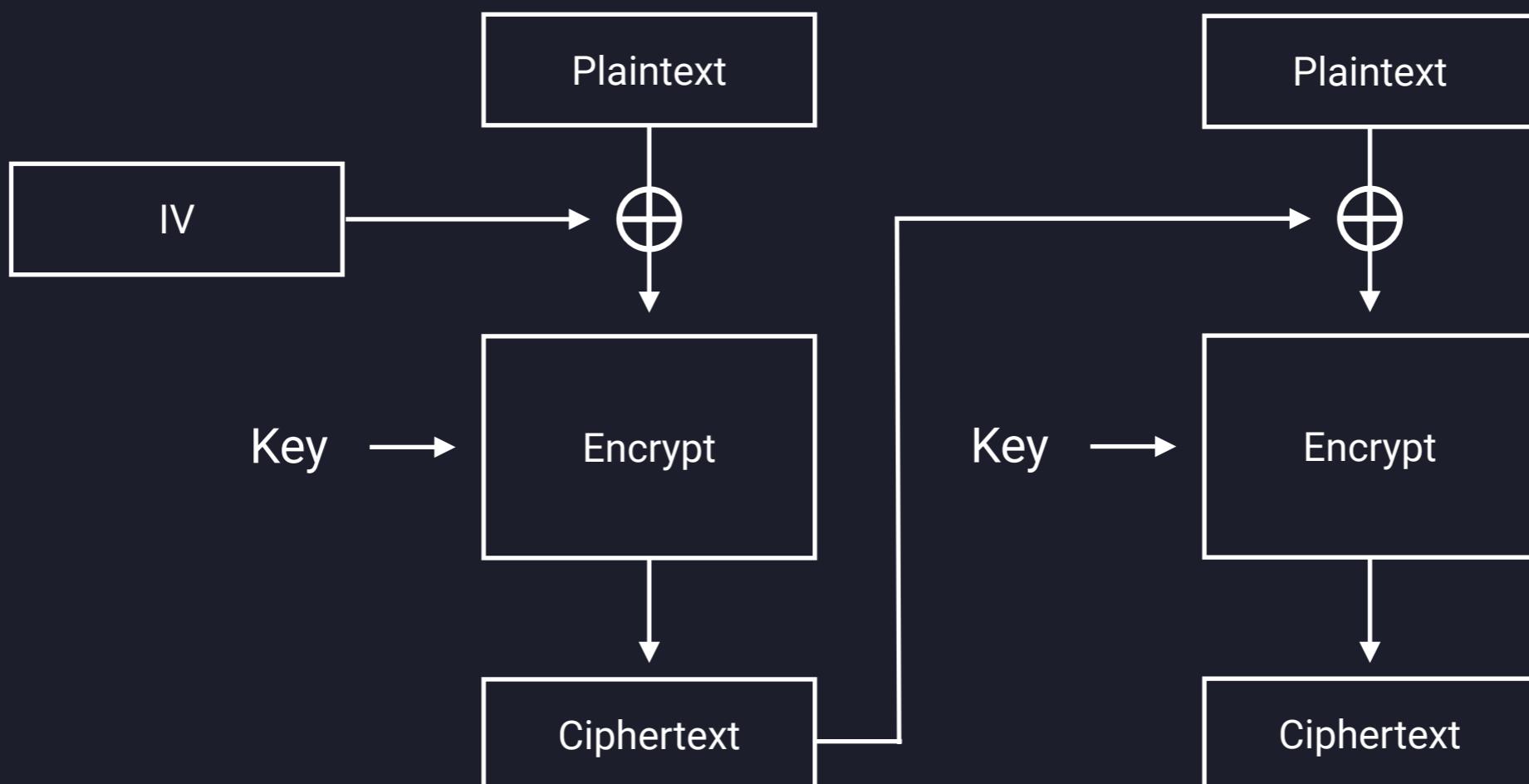
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

ECB 模式的缺點：相同的明文區塊會加密出相同的密文區塊

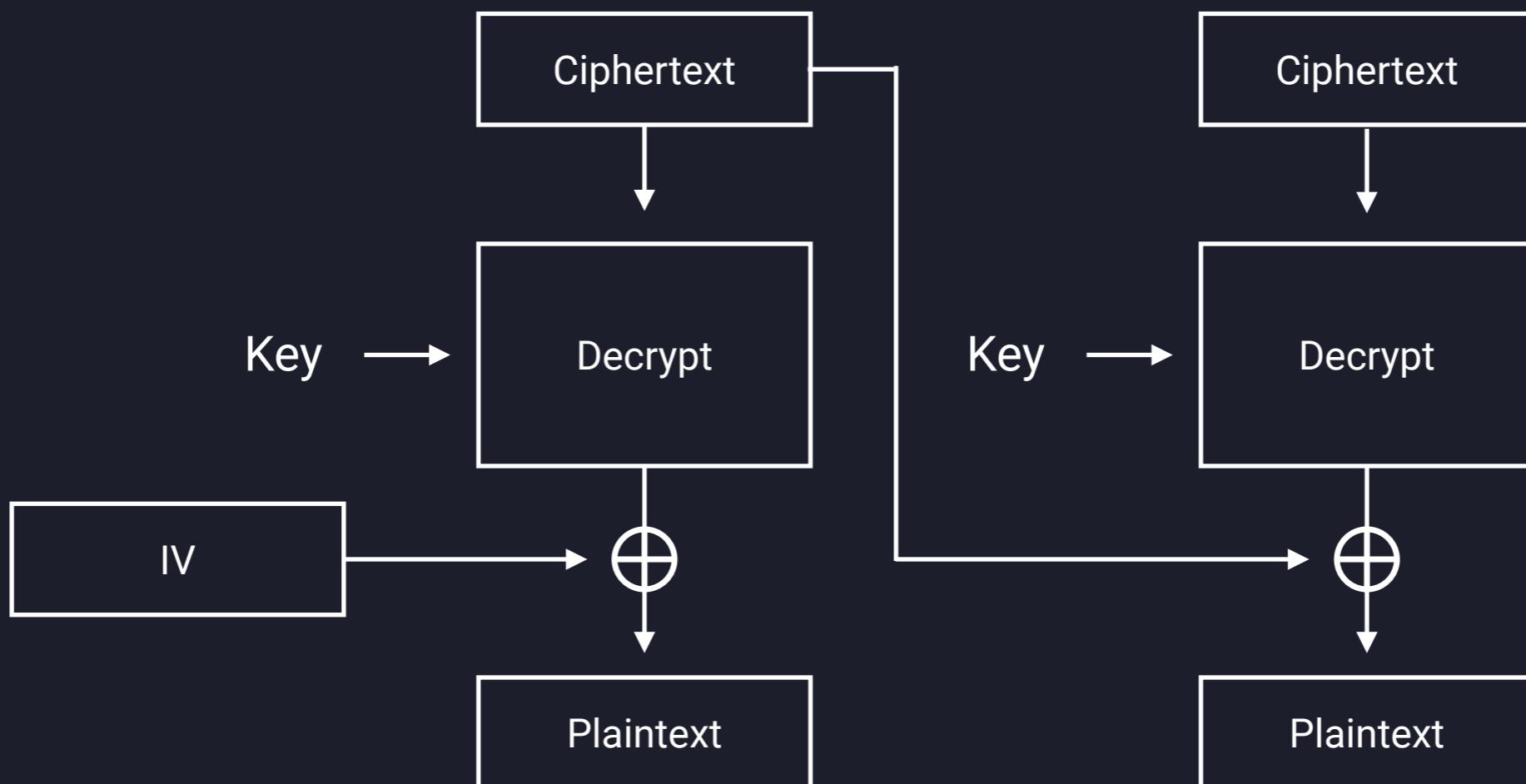


CBC Mode

CBC Mode Encryption



CBC Mode Decryption

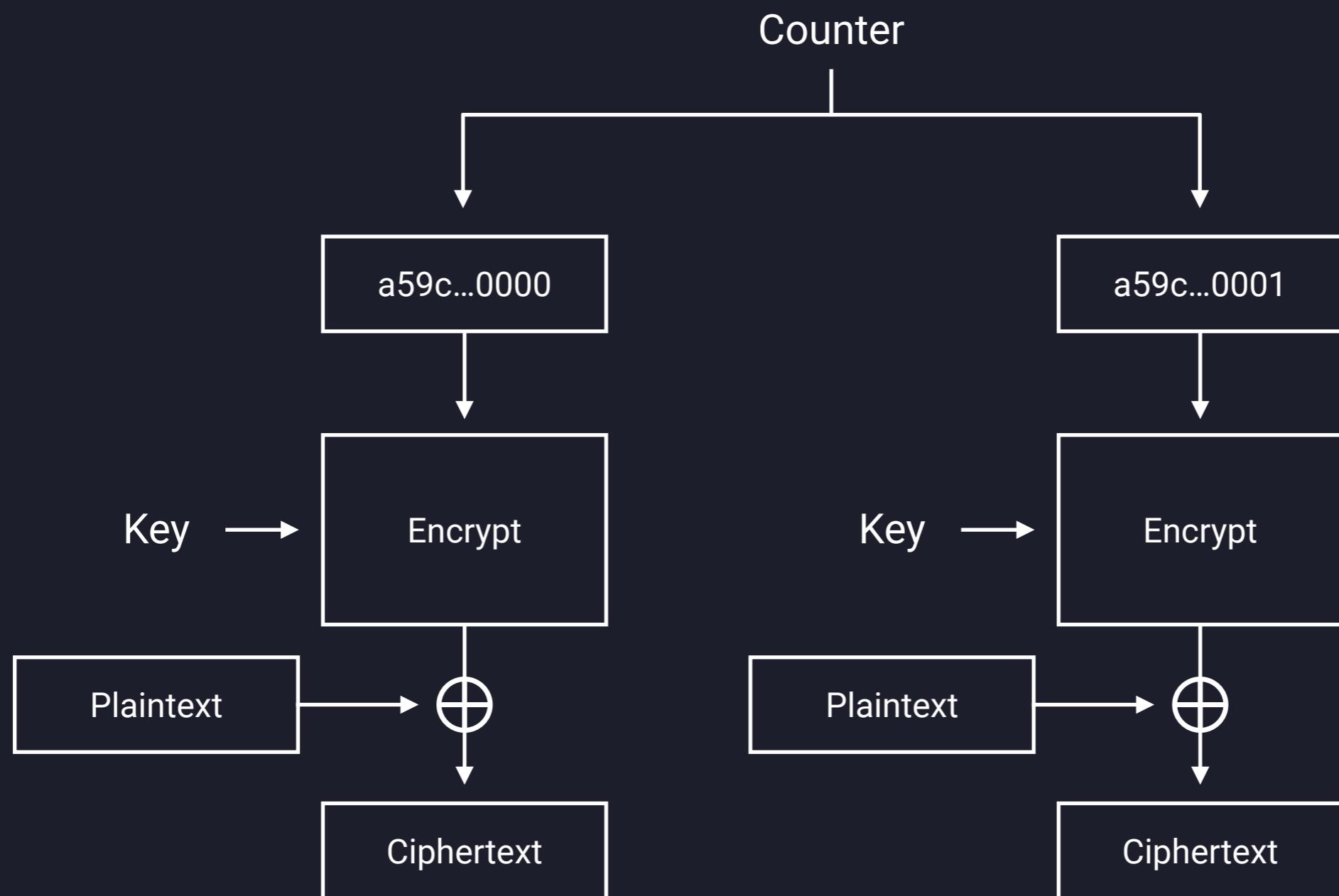


CBC Mode

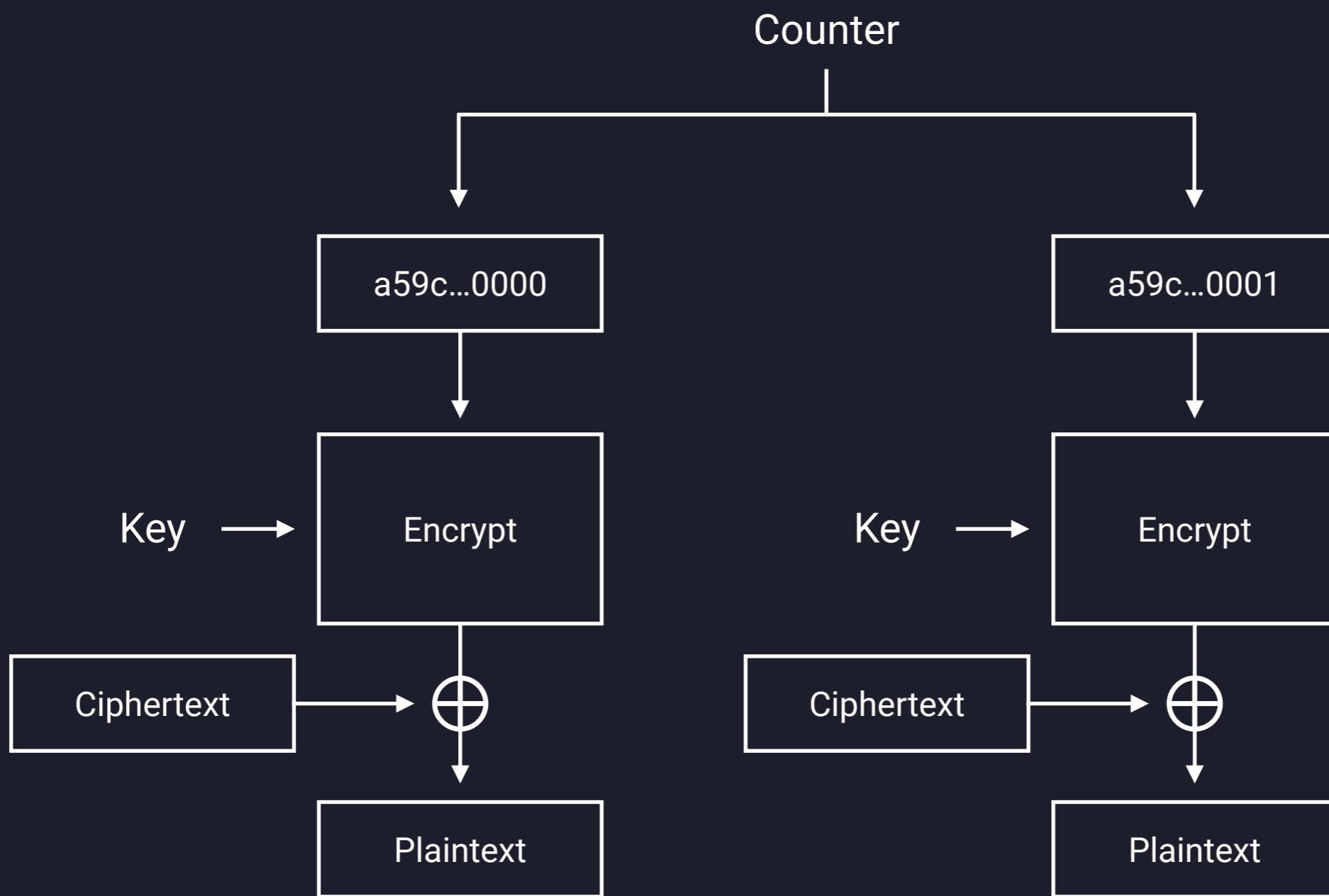
- 每塊密文都依賴前面所有的明文
- 相同的區塊明文會加密出不同的區塊密文
- 有一個初始化向量 (Initial Vector)，簡稱 IV

CTR Mode

CTR Mode Encryption



CTR Mode Decryption



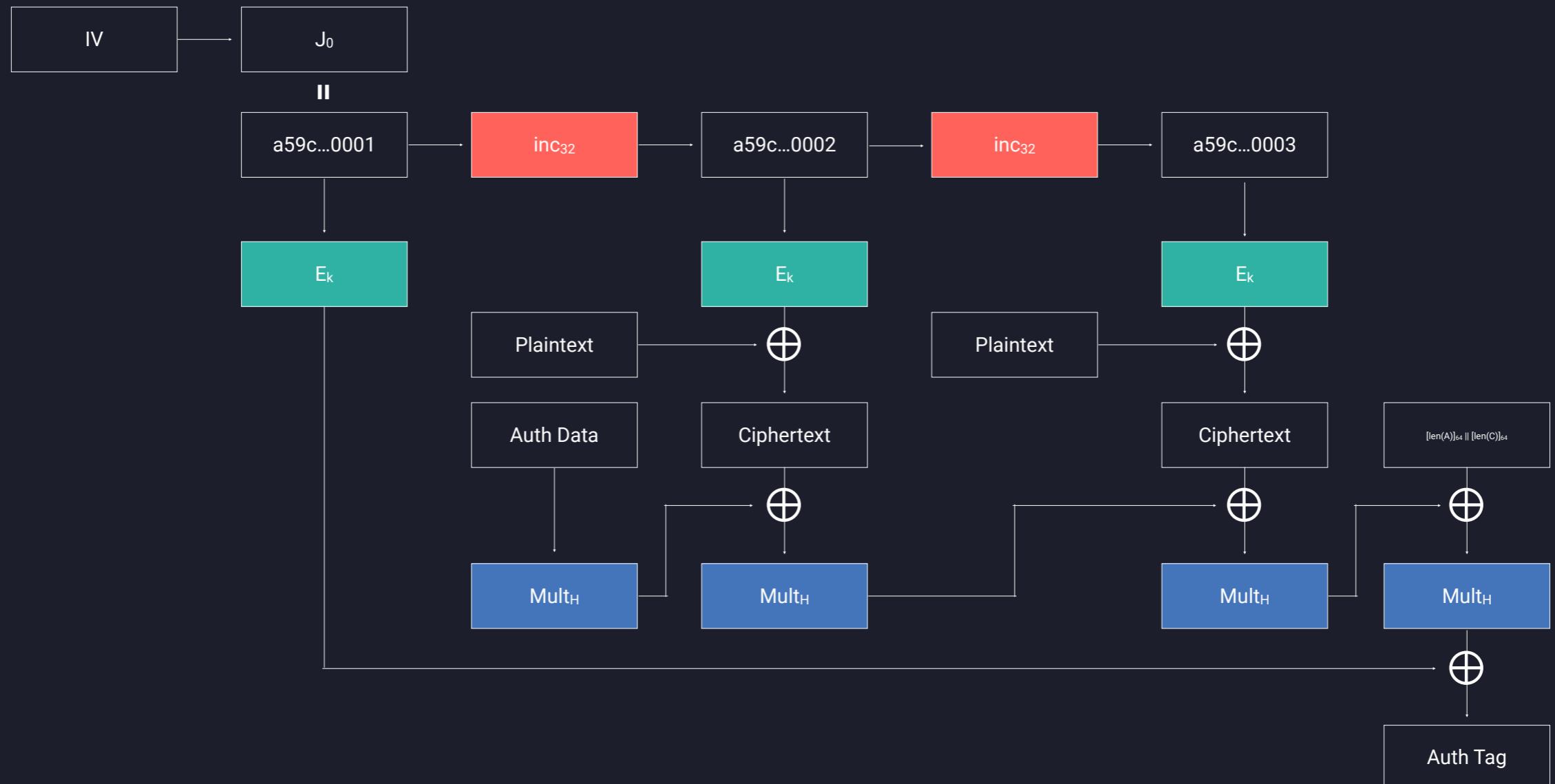
CTR Mode

- 利用 AES 去產生 xor key 然後做 xor cipher
- 加密和解密都是用 AES Encryption，因為要產生相同的 xor key
- Counter 會初始一個隨機數字，每次加一
- Block 之間沒有互相依賴，可平行運算

GCM Mode

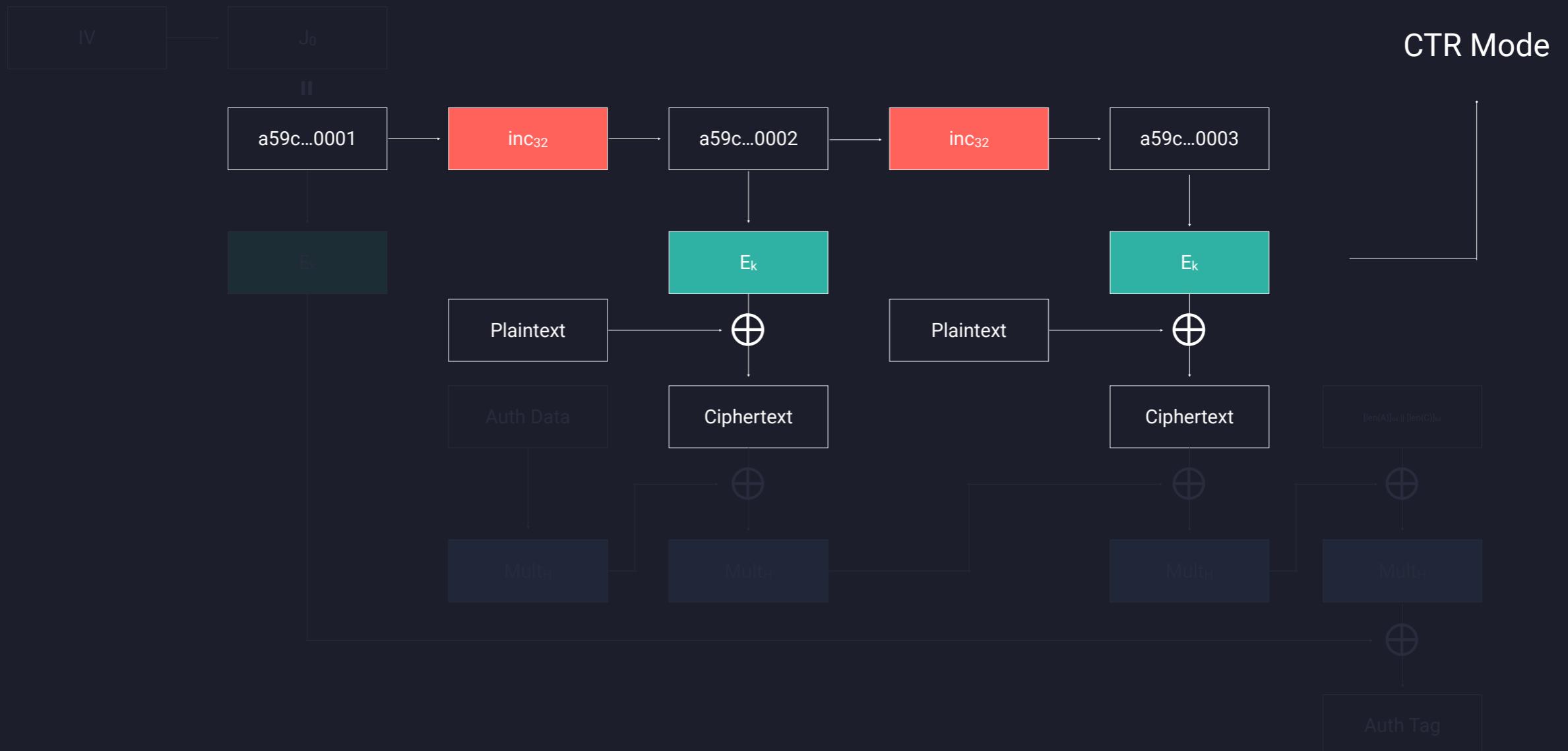
GCM Mode Encryption

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>



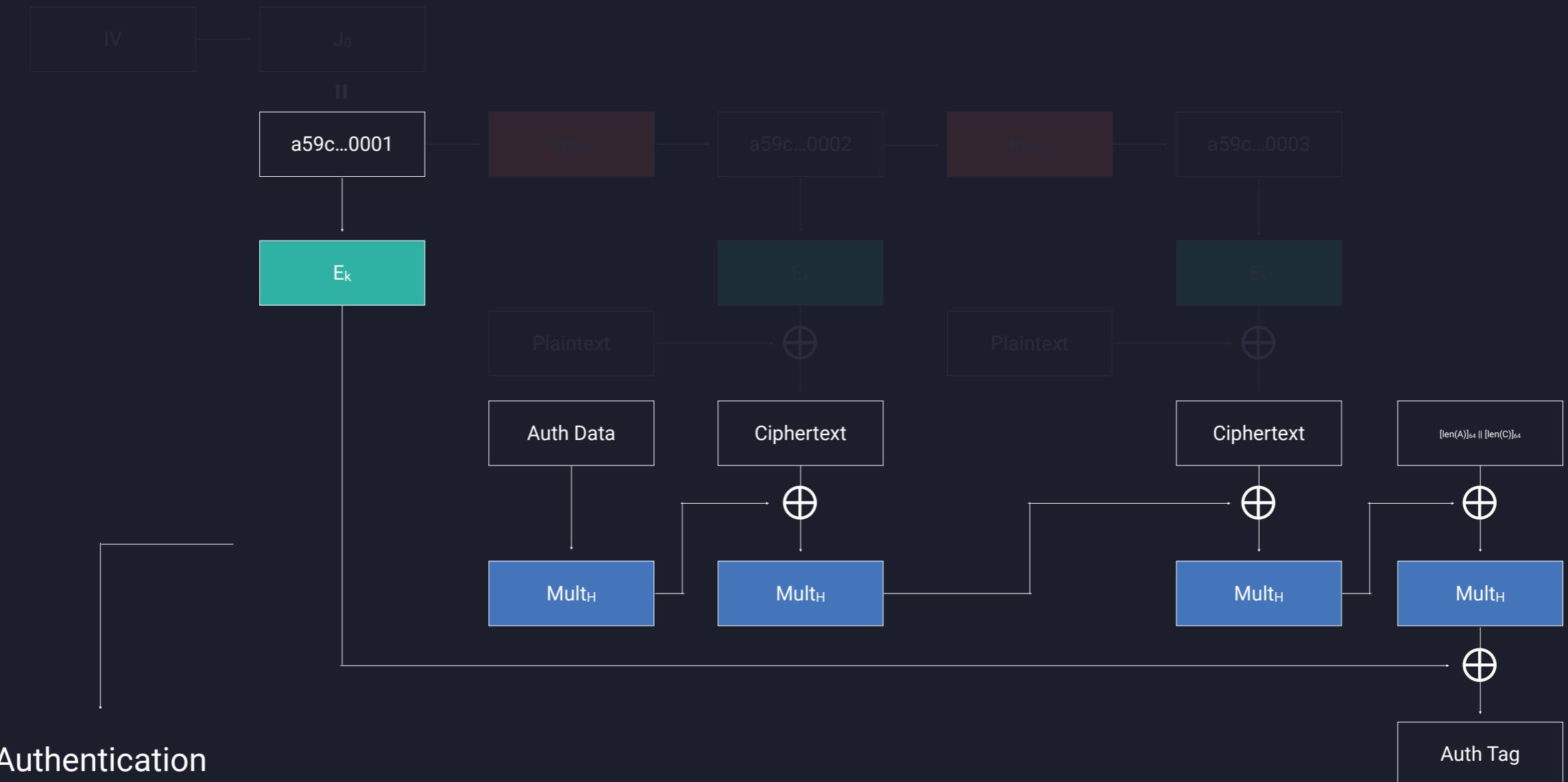
GCM Mode Encryption

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>



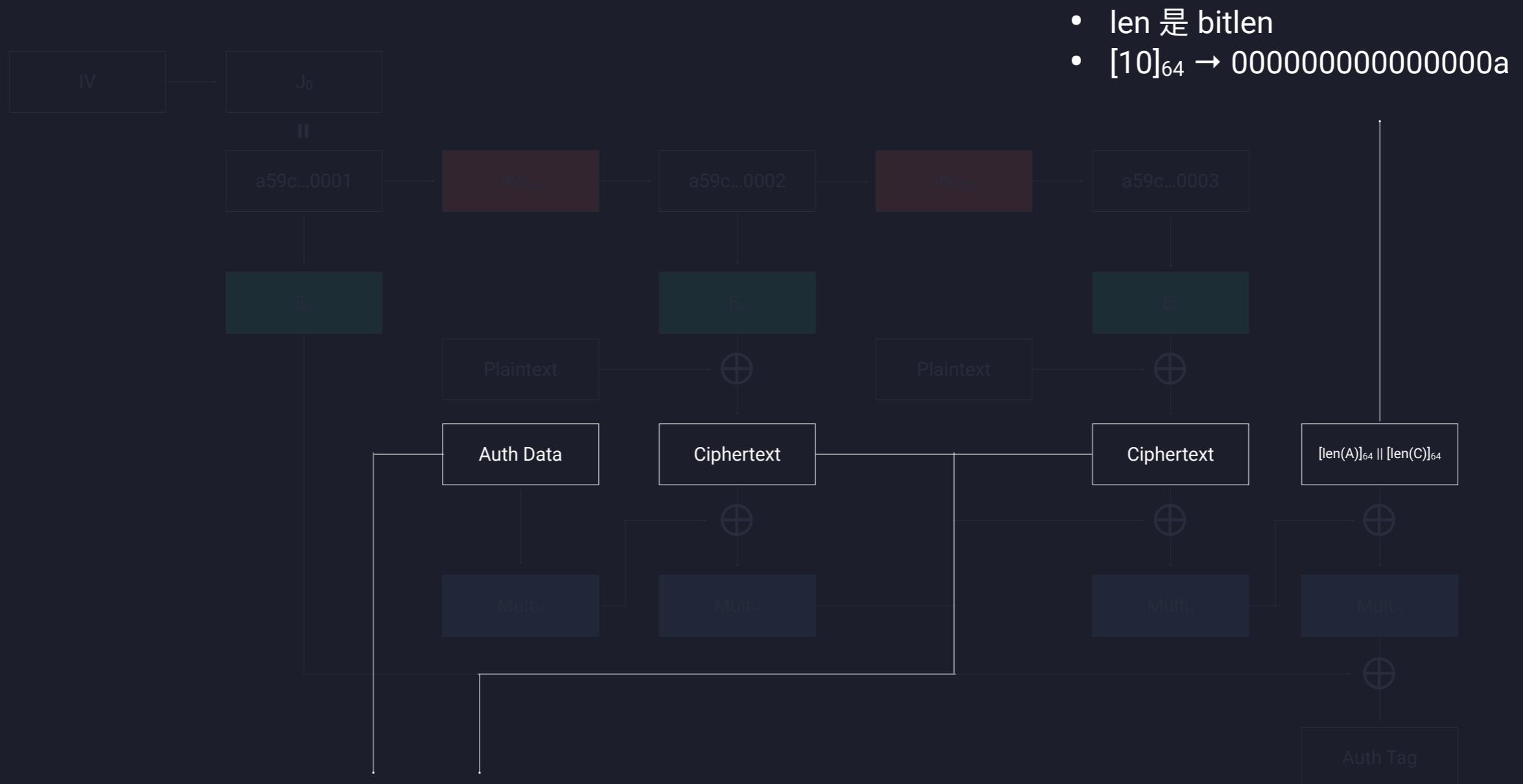
GCM Mode Encryption

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>



GCM Mode Encryption

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>



不是 16 bytes 的倍數會在後面補 \x00

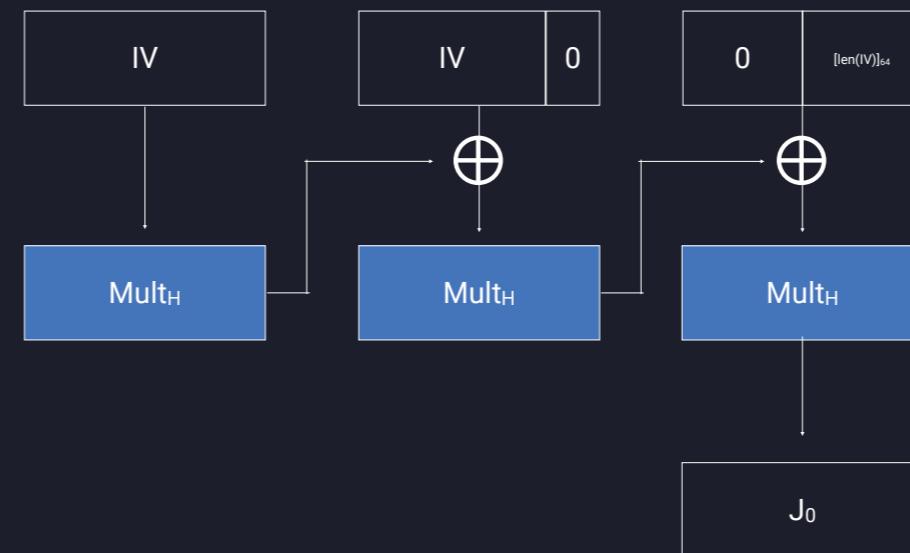
GCM Mode Encryption

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>

如果 $\text{len}(\text{IV}) = 96$

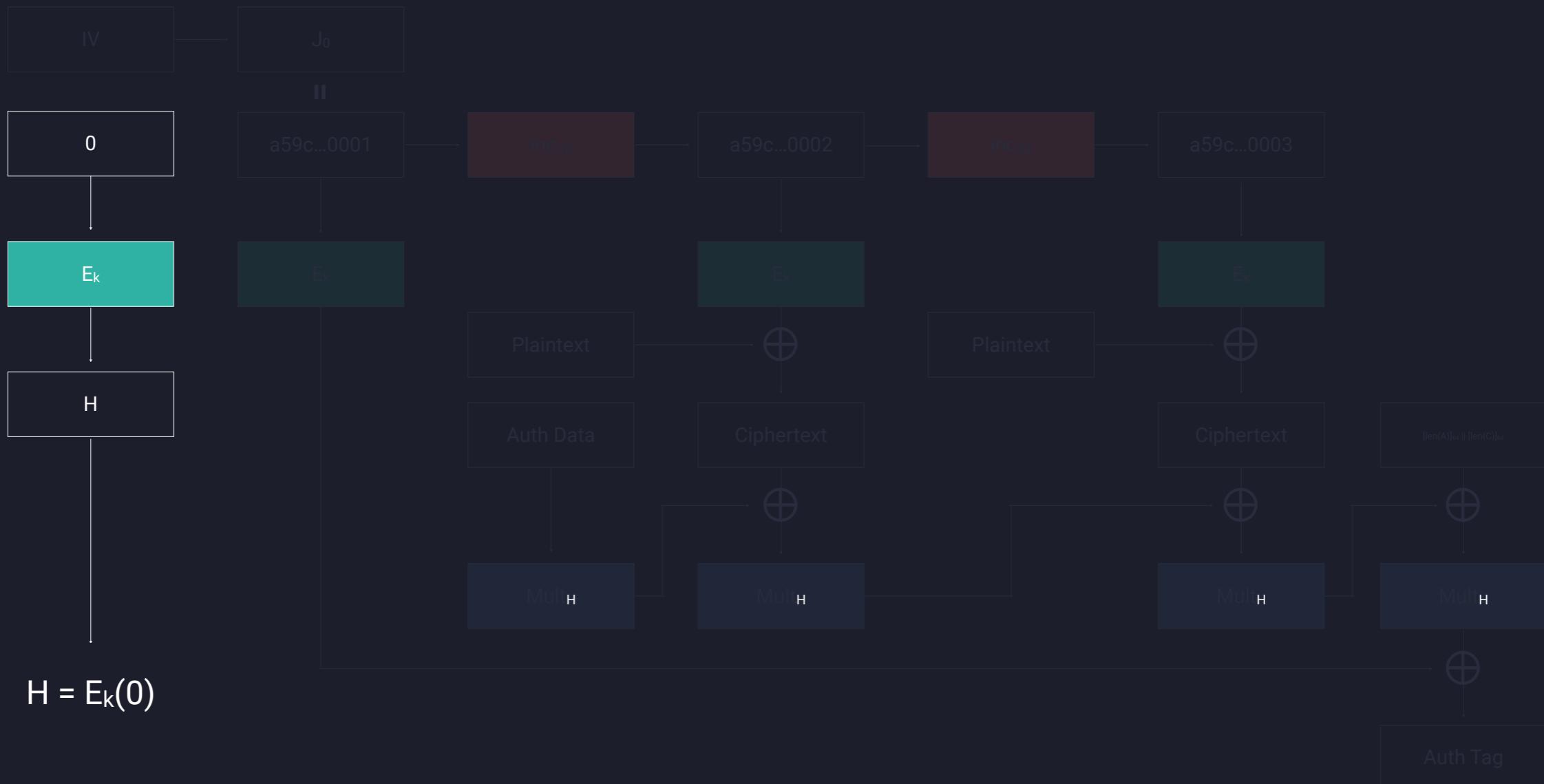


如果 $\text{len}(\text{IV}) \neq 96$



GCM Mode Encryption

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>



GCM Mode Encryption

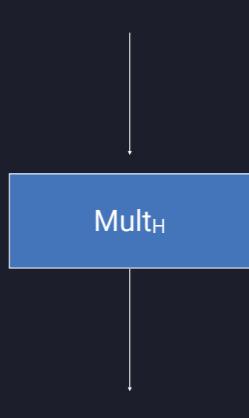
<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>



- 低位 32 bits 加一 modulo 2^{32}
- 剩下的高位不動

GCM Mode Encryption

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>

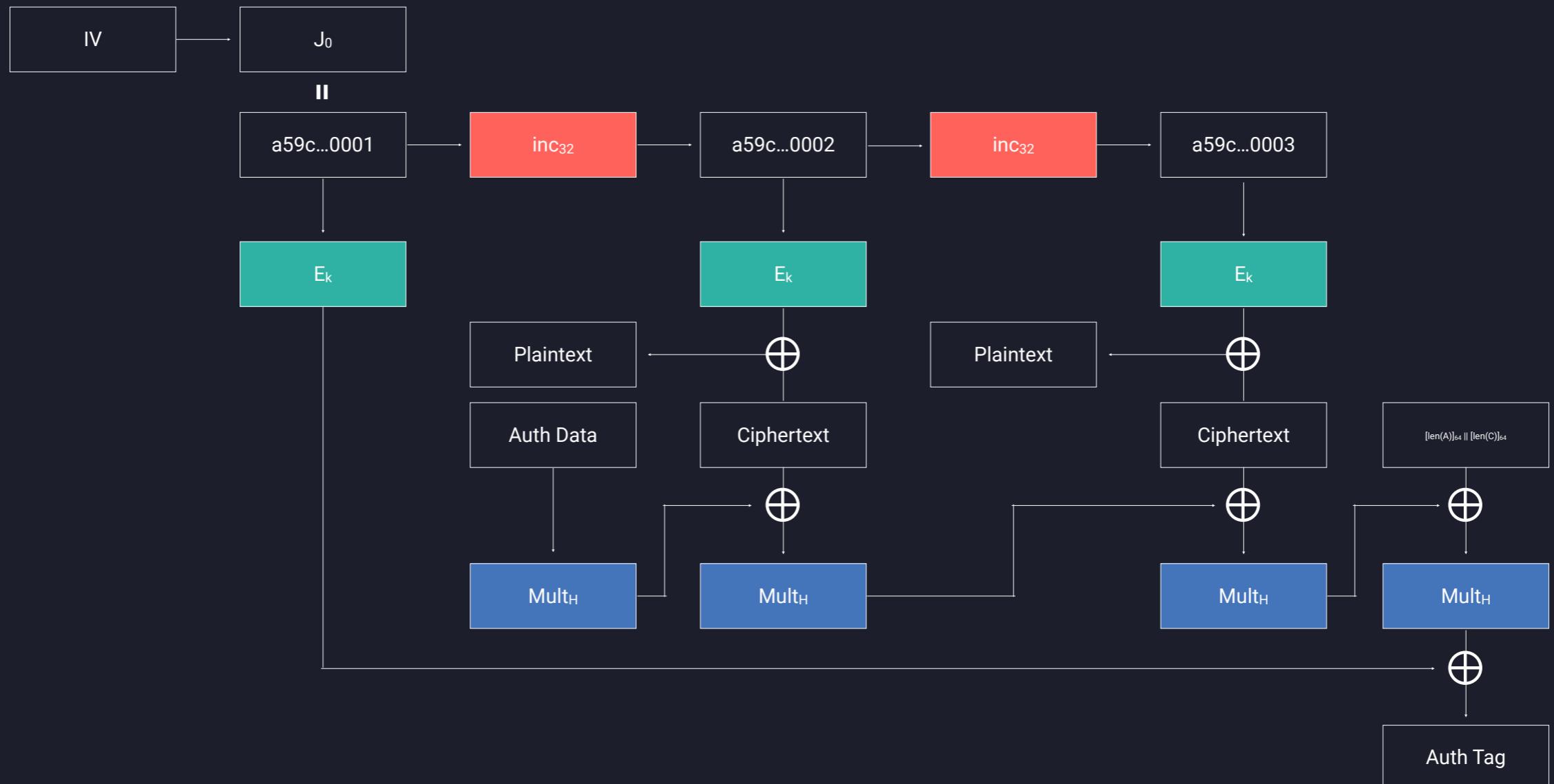
- 
- Mult_H 是在 $\text{GF}(2^{128})$ 下乘上 H
 - $\text{GF}(2^{128})$ 的 reduction modulus 是 $x^{128} + x^7 + x^2 + x + 1$
 - 從 bytes 轉成 $\text{GF}(2^{128})$ 下的元素
 - u is the variable of the polynomial
 - $x_0x_1\dots x_{127} \rightarrow x_0 + x_1u + \dots + x_{127}u^{127}$
 - '\x00...\x00\x01\x02' $\rightarrow x^{119}+x^{126}$

Galois Field

- Galois Field (GF) 或叫 Finite Field
- GF(2^{128}) 裡面的元素可以表示成一個 degree = 127 的多項式
- 加法和乘法就是做多項式的加法和除法 modulo 一個 degree = 128 的 irreducible polynomial
 - GCM Mode 用的 GF(2^{128}) 是 modulo $x^{128} + x^7 + x^2 + x + 1$

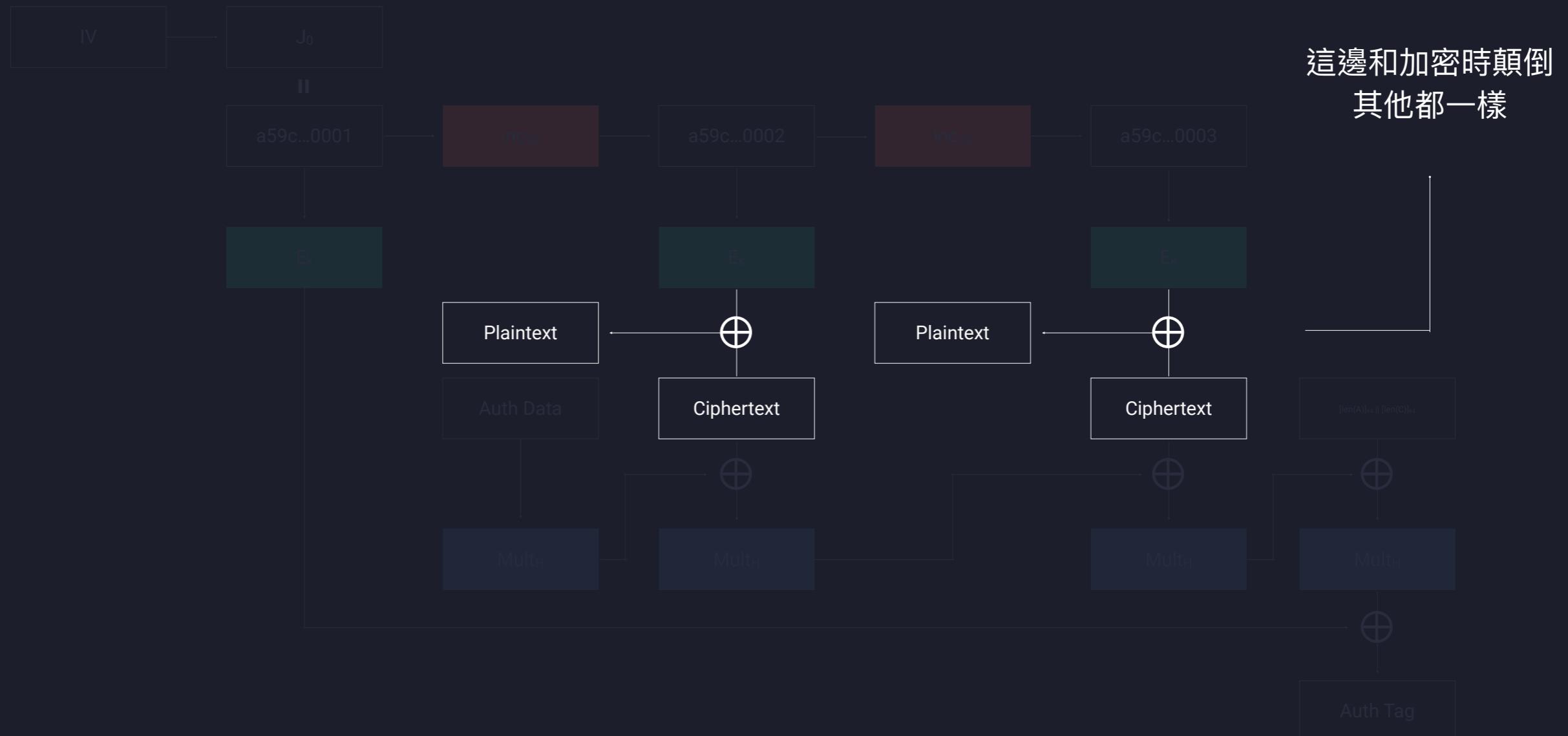
GCM Mode Decryption

<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>



GCM Mode Decryption

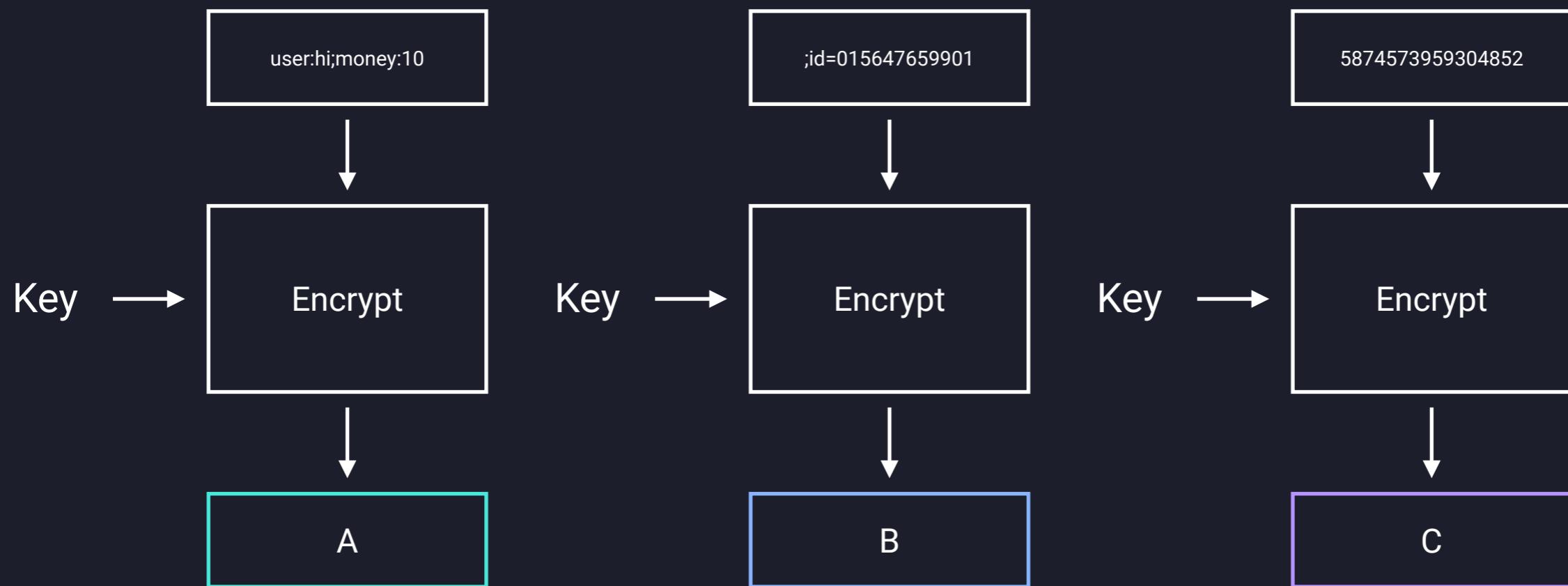
<https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38d.pdf>



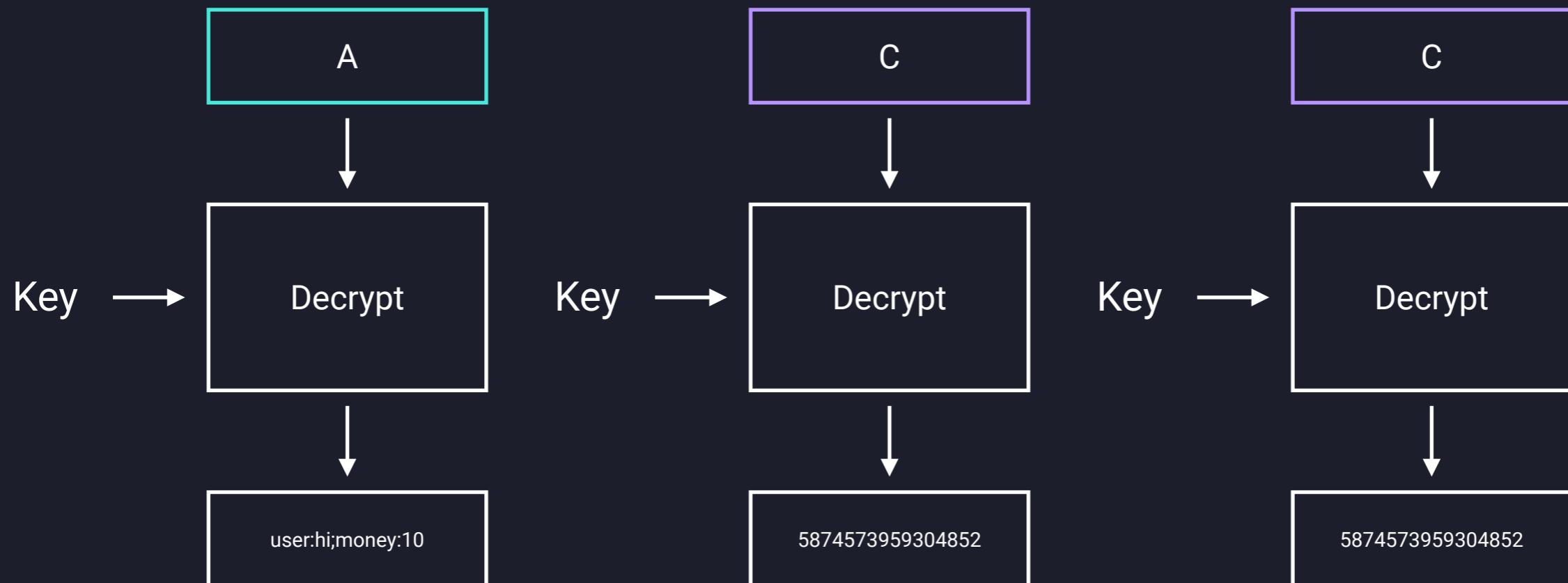
Attack Scenarios

ECB Mode / Cut & Paste

ECB Mode / Cut & Paste



ECB Mode / Cut & Paste



錢錢變多了~

ECB Mode / Prepend Oracle Attack

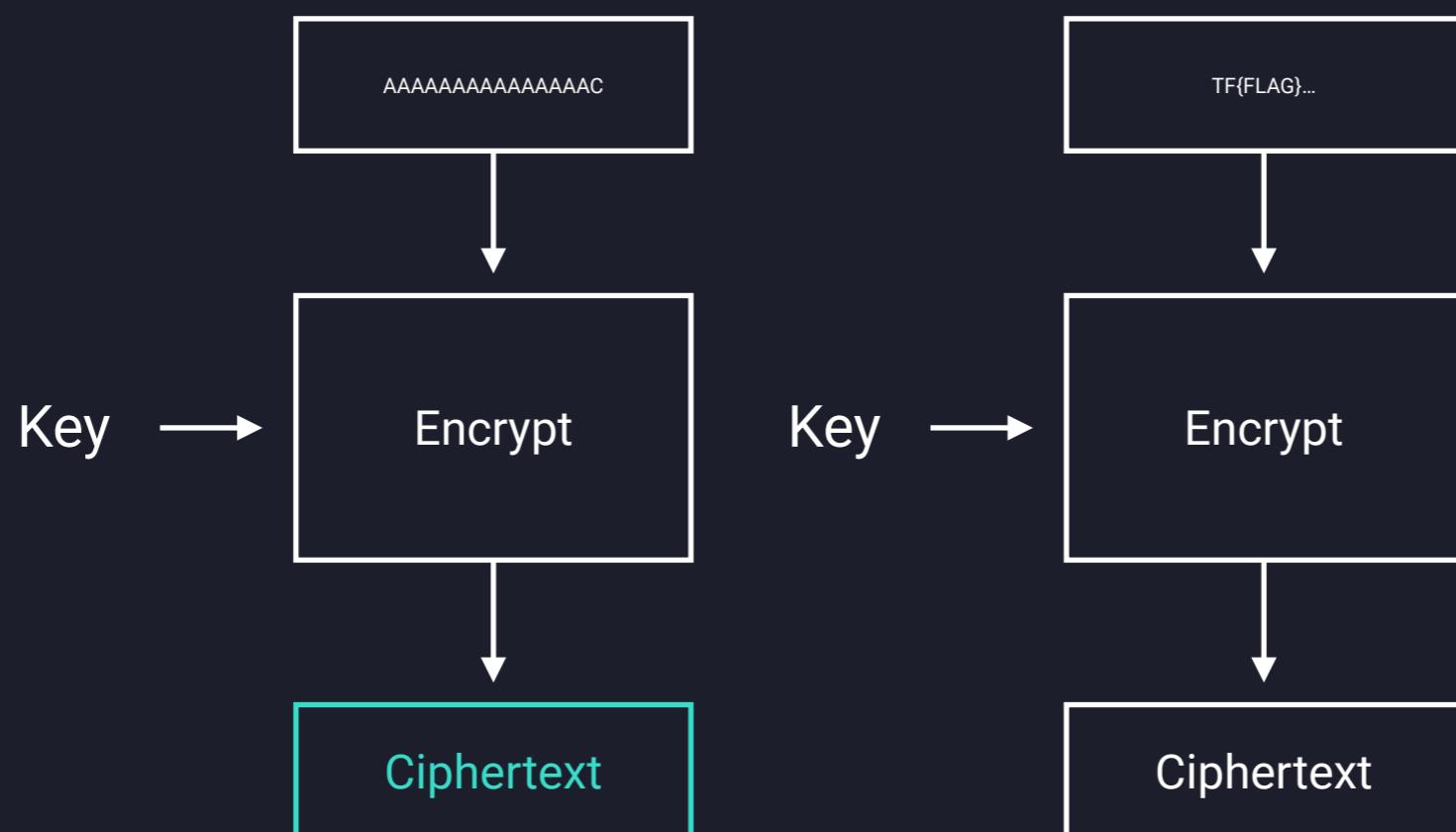
The Oracle

- 使用 AES ECB Mode
- 伺服器將我們的明文 prepend 在 flag 前面作加密

```
def oracle(plain):  
    aes = AES.new(KEY, AES.MODE_ECB)  
    return aes.encrypt(plain + flag)
```

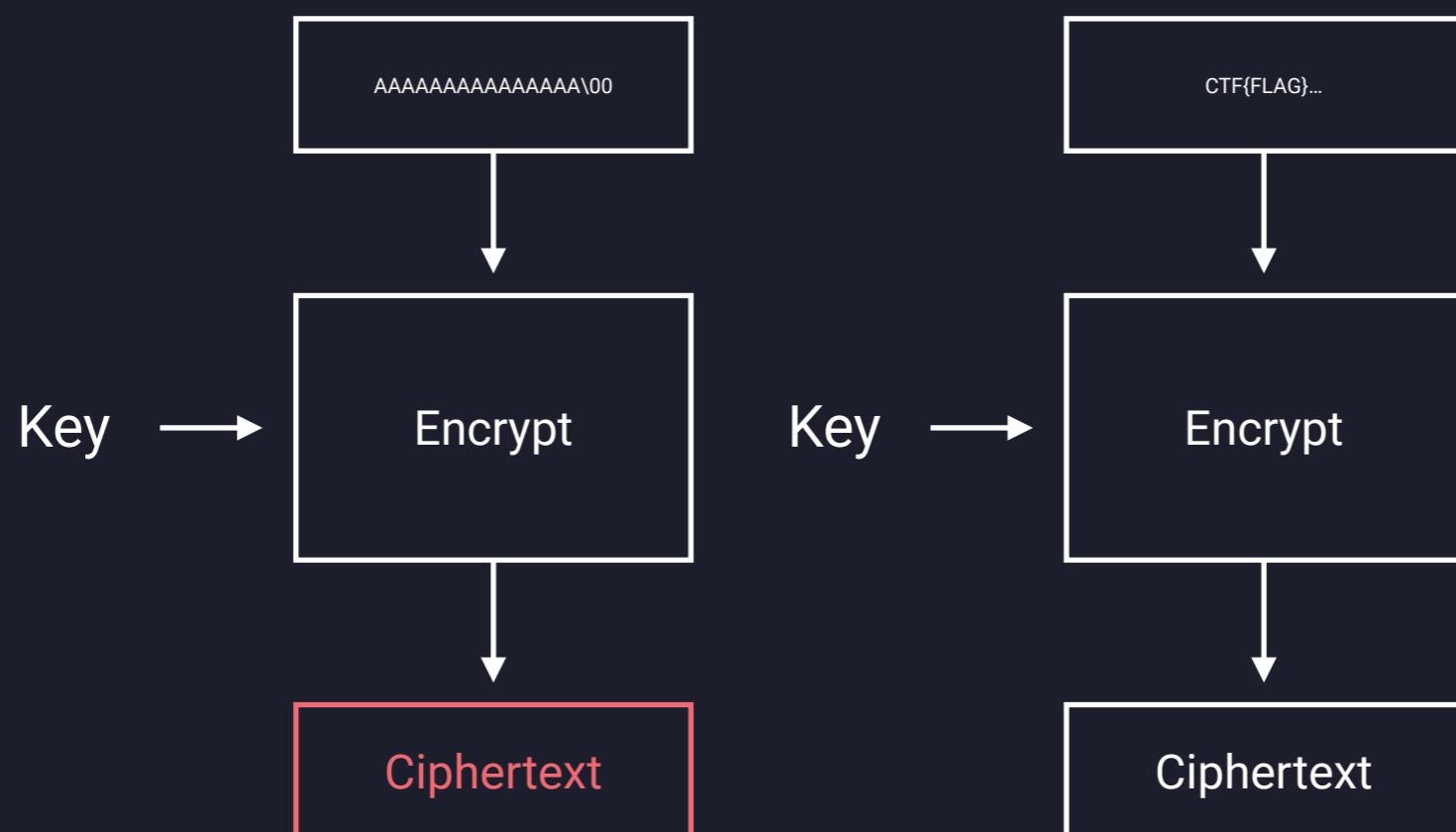
找 flag[0]

先塞 15 個垃圾
讓 flag 的第一個字元掉進來



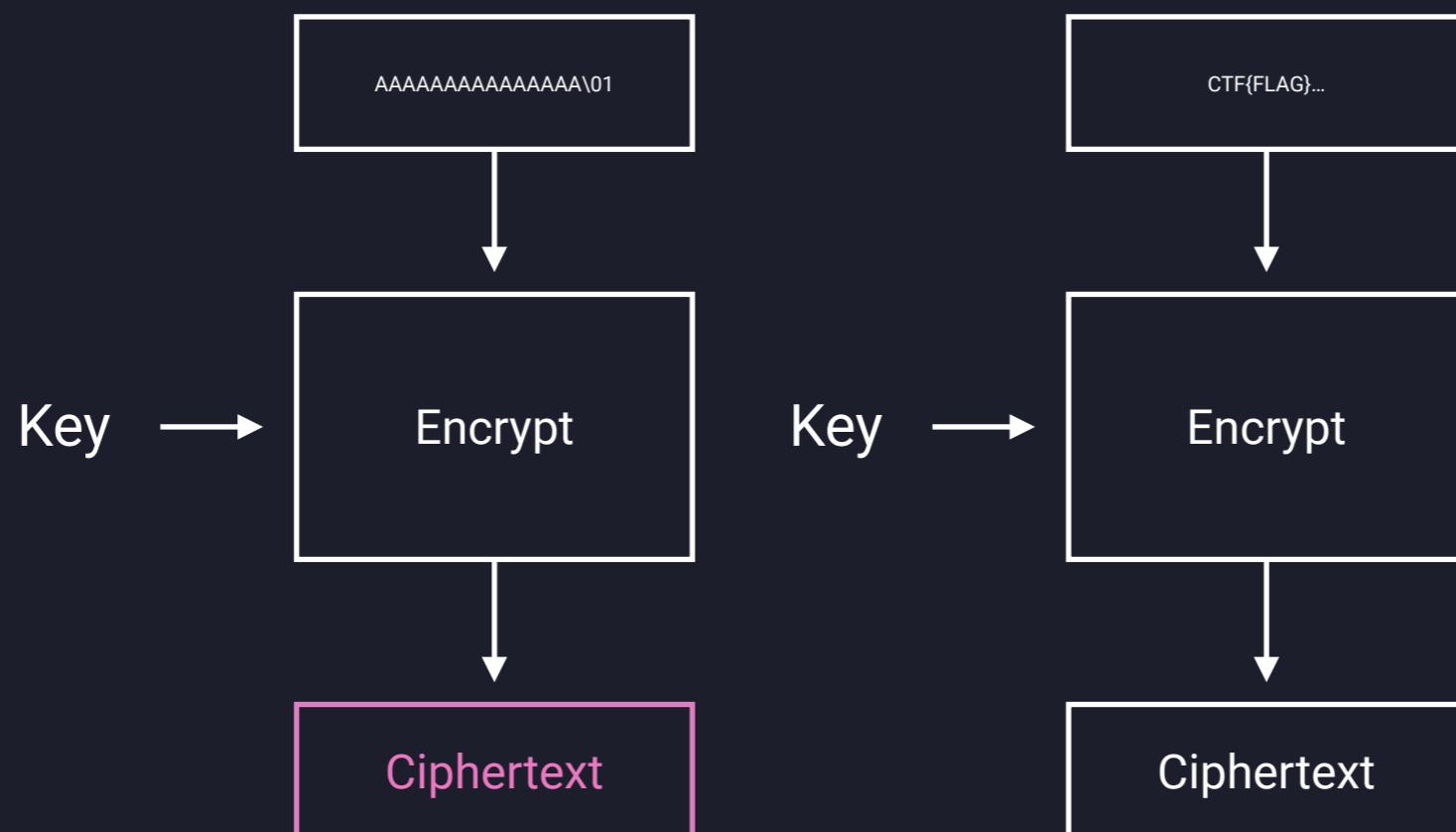
找 flag[0]

塞 15 個一樣的垃圾 +
爆搜最後一個 byte



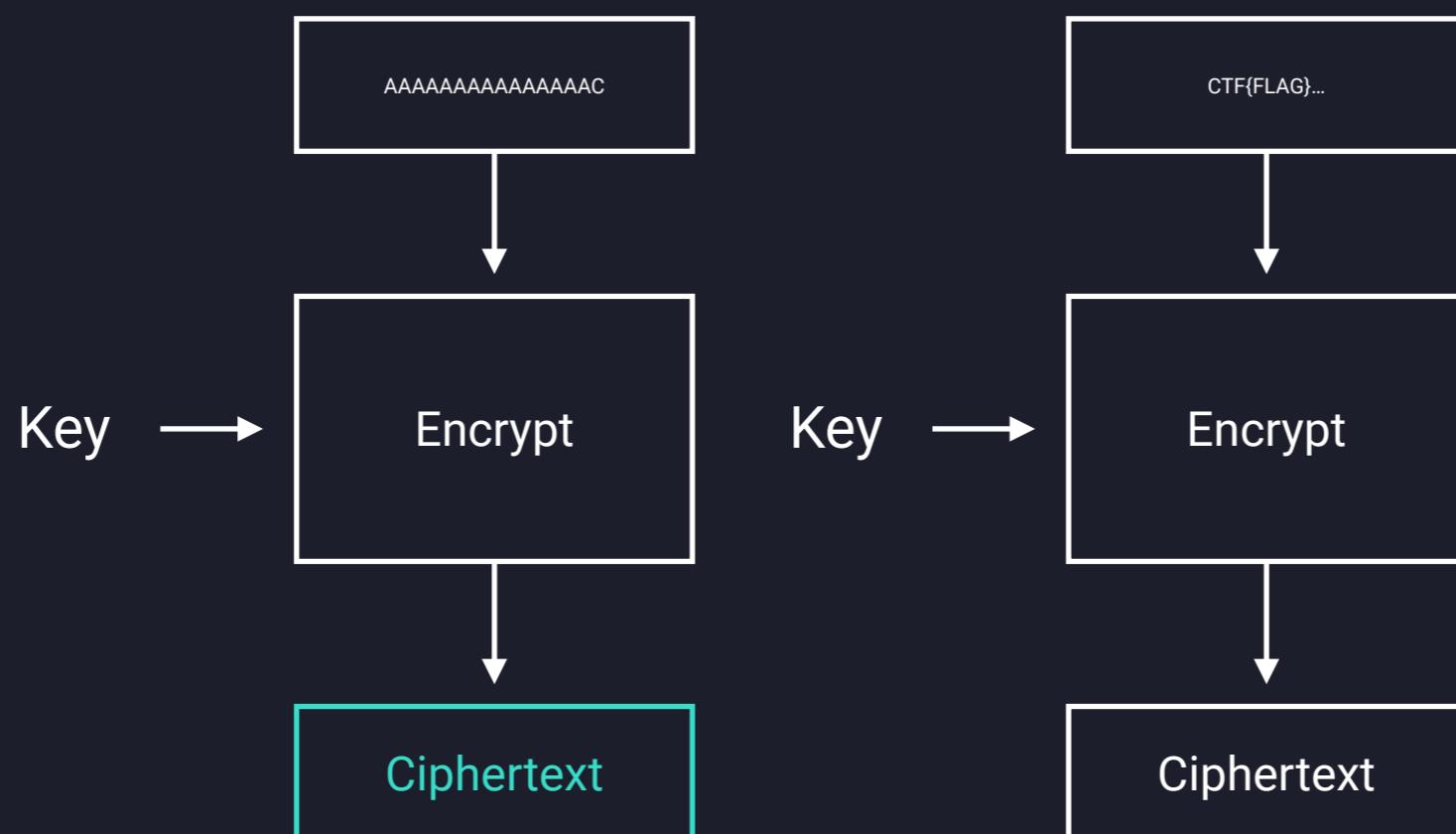
找 flag[0]

塞 15 個一樣的垃圾 +
爆搜最後一個 byte



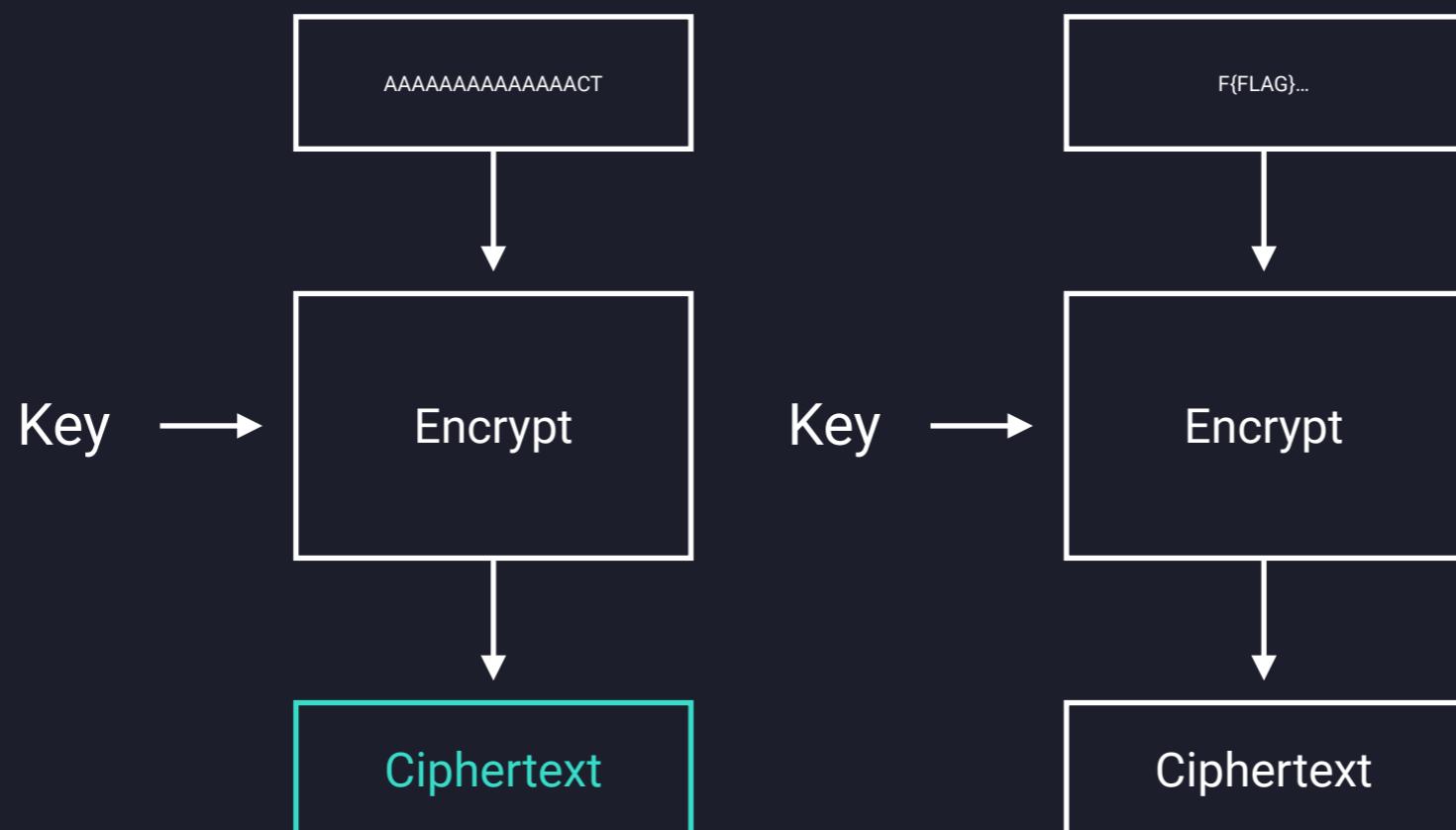
找 flag[0]

找到了 flag 的第一個 byte



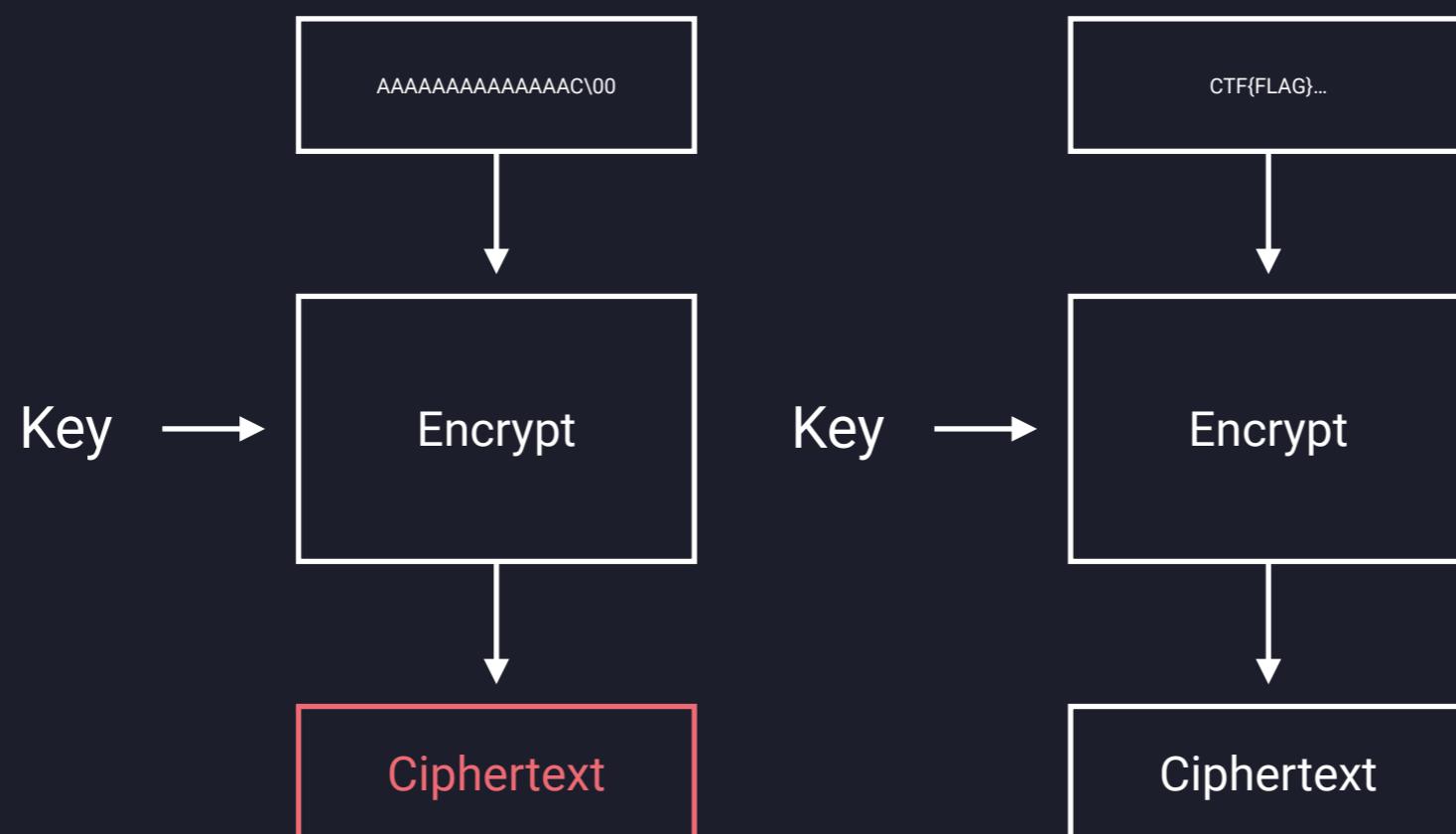
找 flag[1]

先塞 14 個垃圾
讓 flag 的前兩個字元掉進來



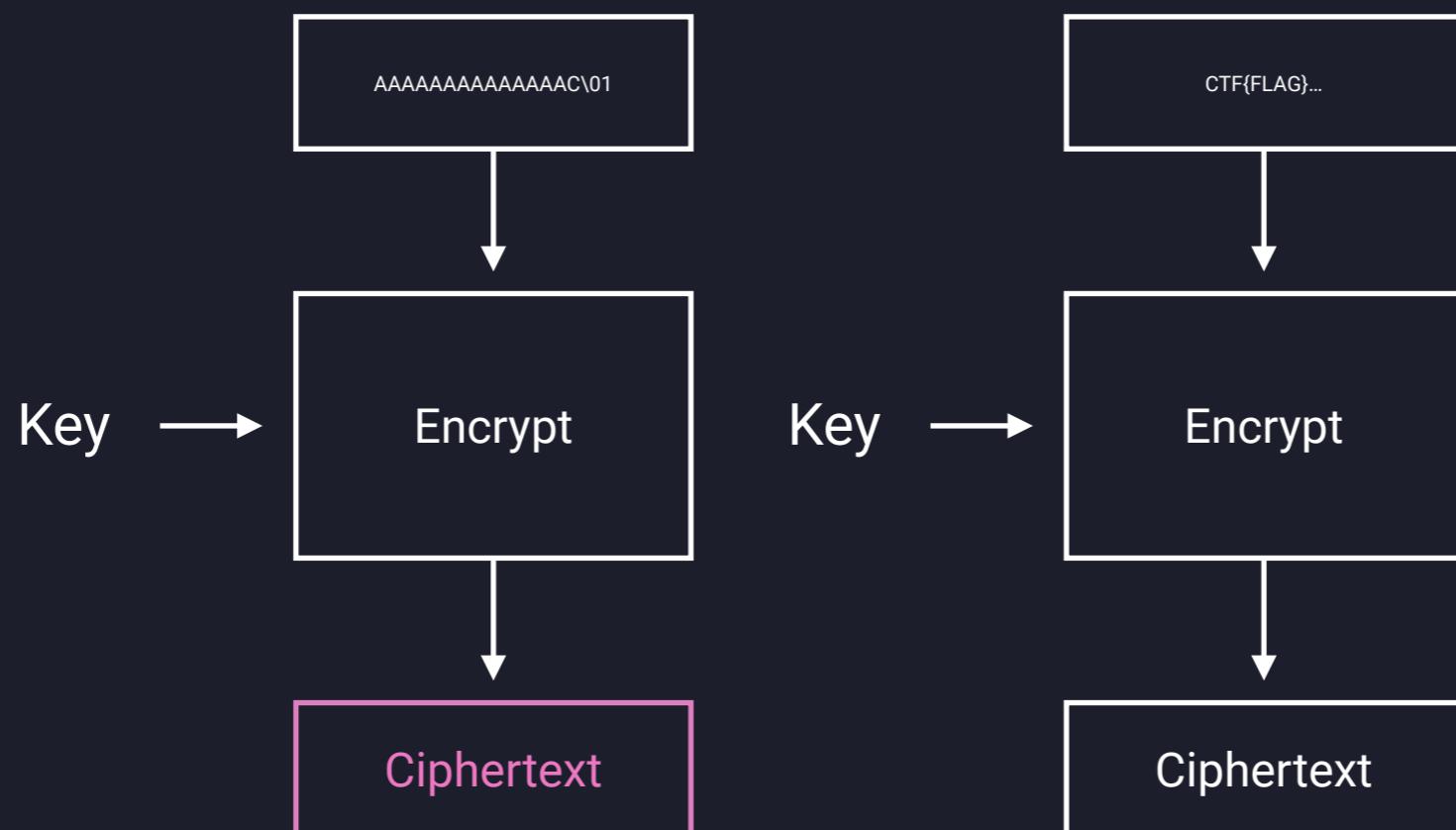
找 flag[1]

塞 14 個一樣的垃圾 +
已知的 flag 第一個 byte +
爆搜最後一個 byte



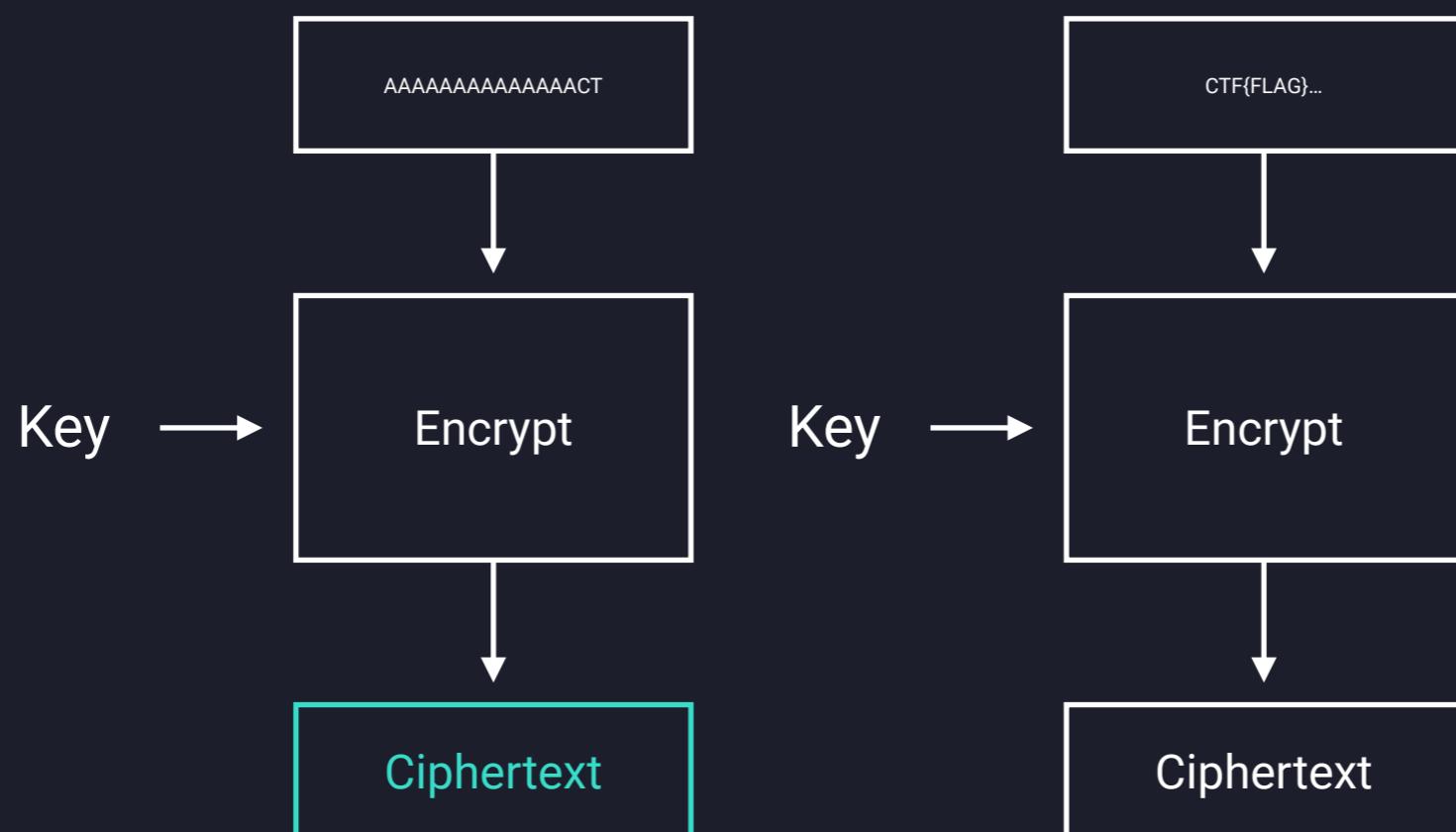
找 flag[1]

塞 14 個一樣的垃圾 +
已知的 flag 第一個 byte +
爆搜最後一個 byte



找 flag[1]

找到了 flag 的第二個 byte



Finally

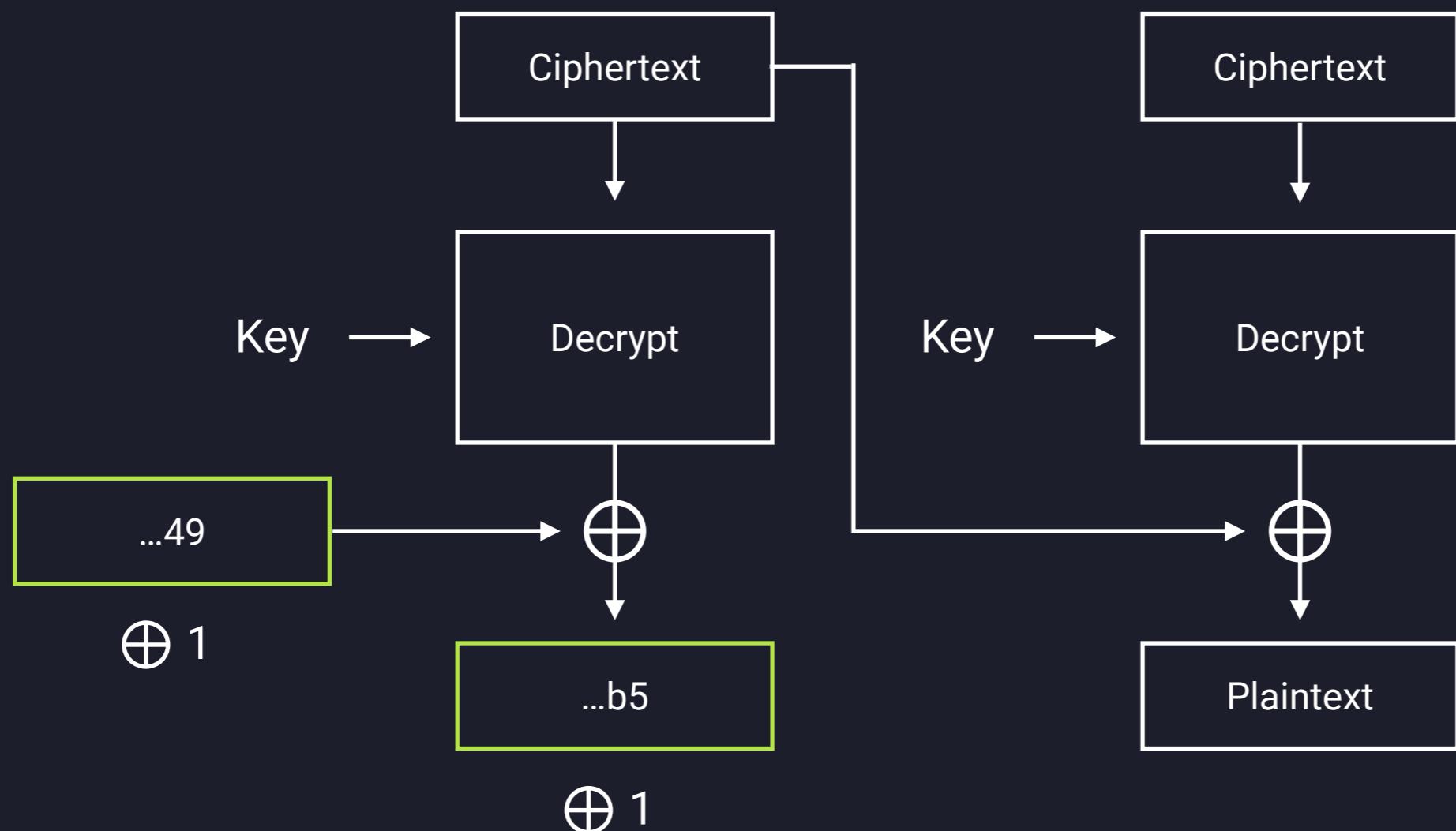
- 這樣一直做下去就能解密出 flag 的所有字元了
- 最多需要做 $\text{len(flag)} * 257$ 次

CTF Challenges

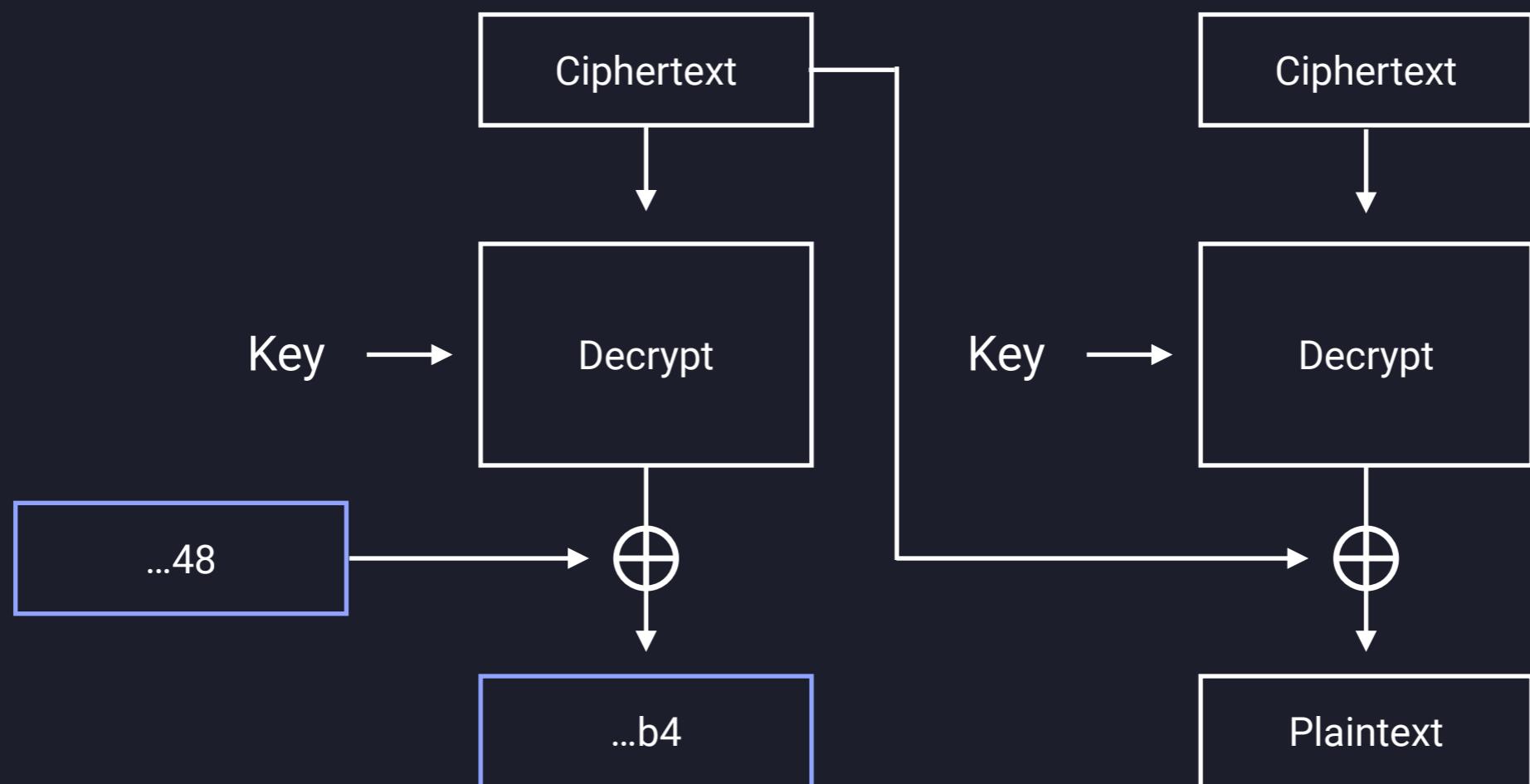
- [UTCTF 2020 - Random ECB](#)
- [TUCTF 2018 - AESential Lesson](#)
- [cryptohack.org - ECB Oracle](#)

CBC Mode / Bit-Flipping Attack

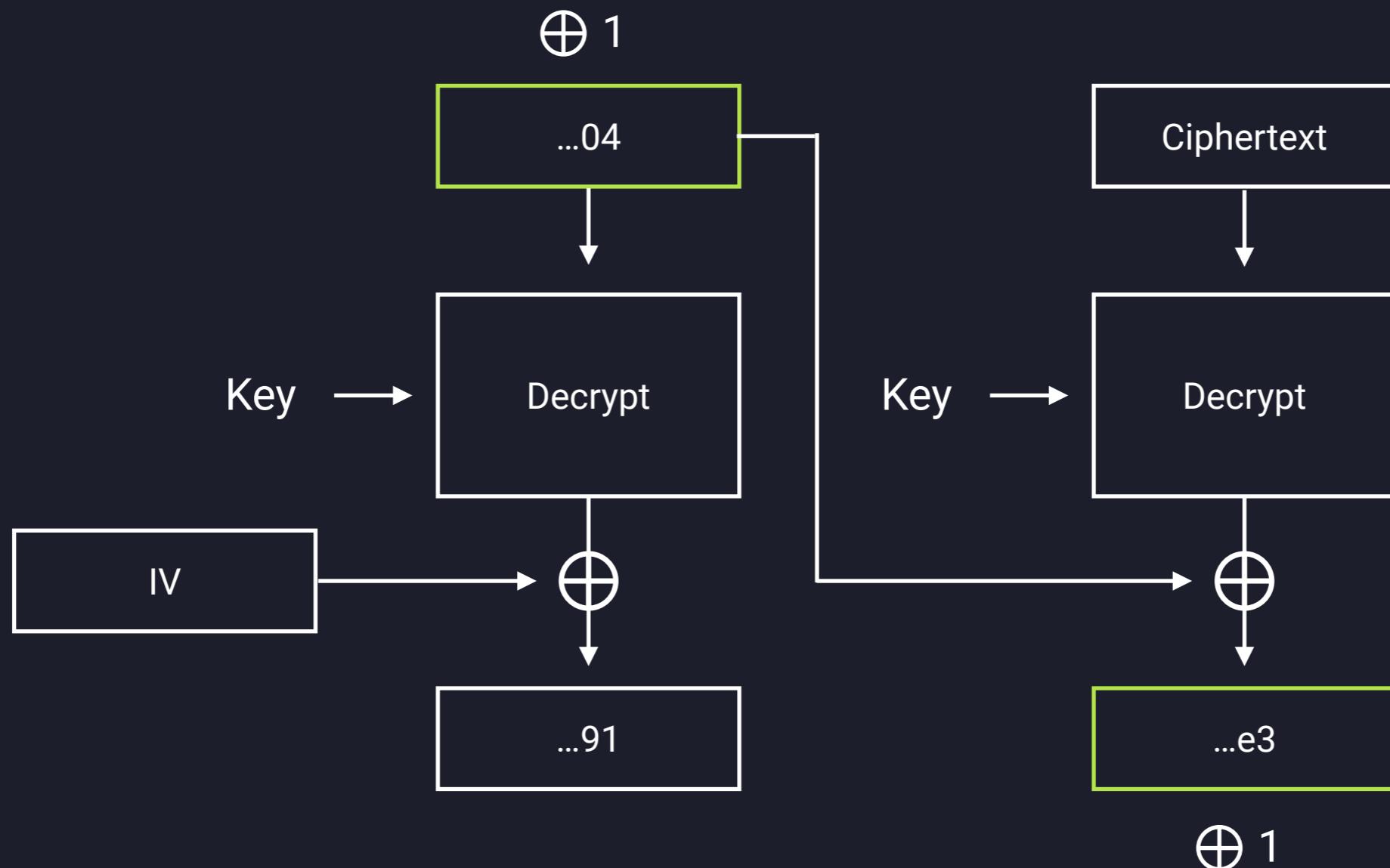
CBC Mode / Bit-Flipping Attack



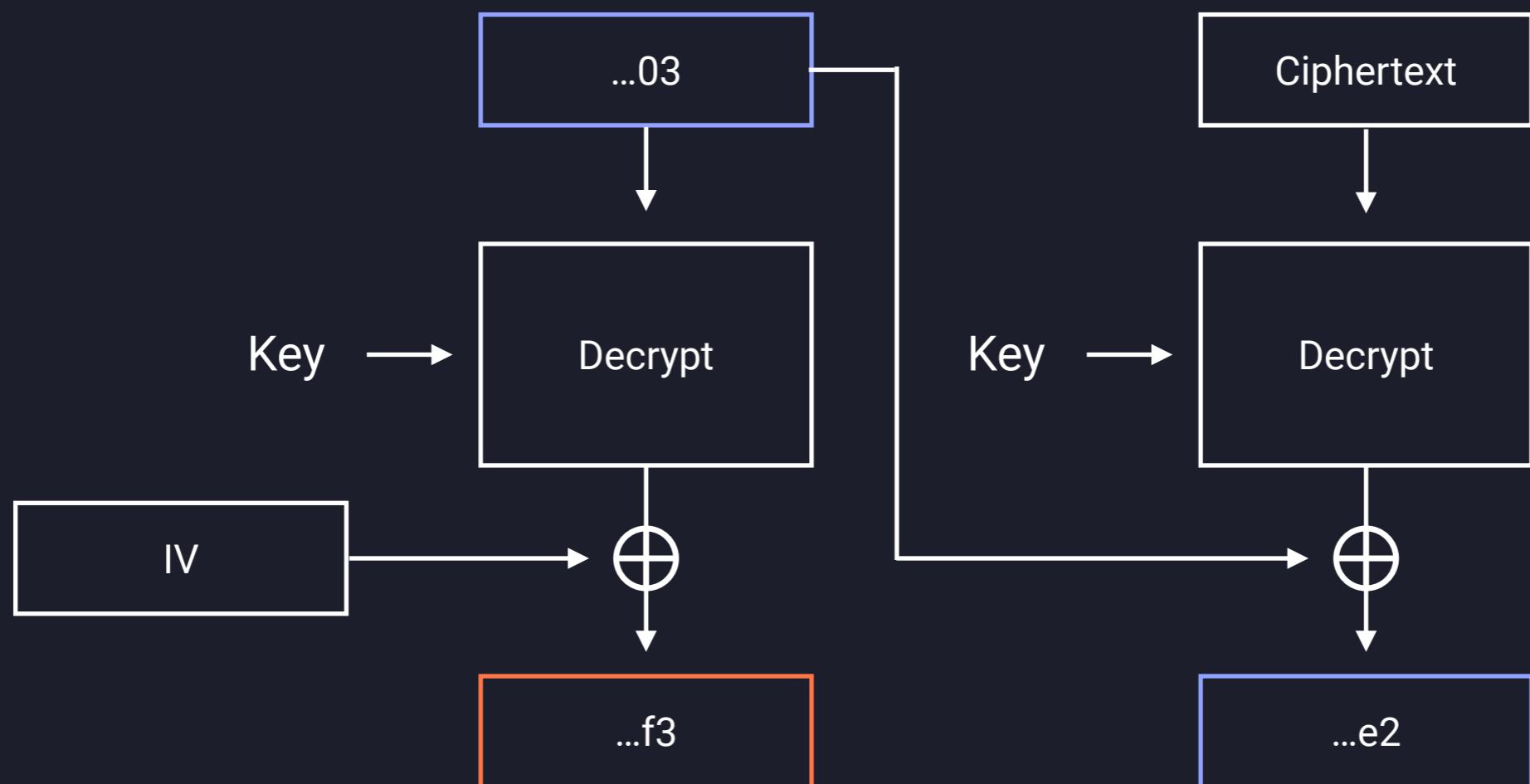
CBC Mode / Bit-Flipping Attack



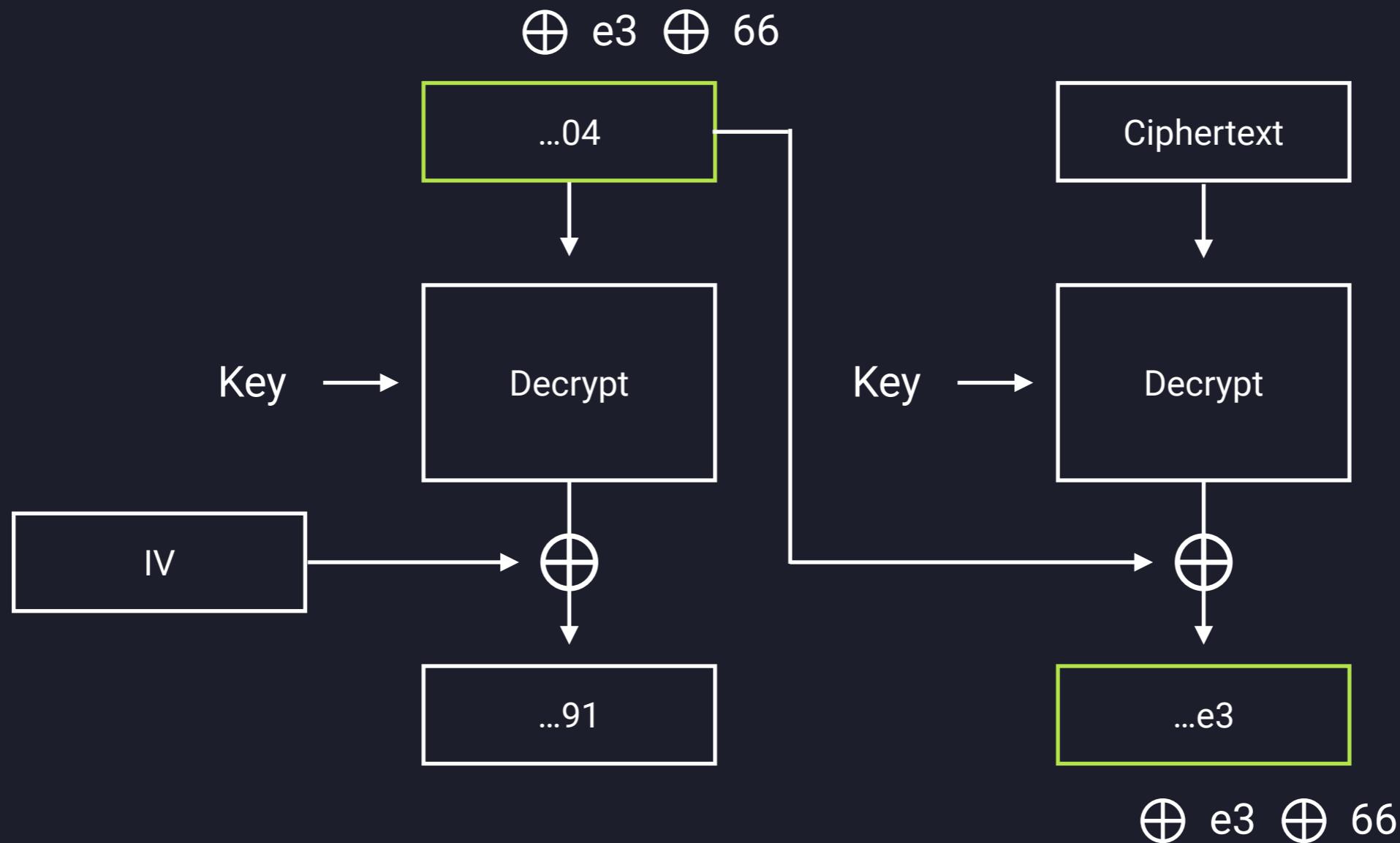
CBC Mode / Bit-Flipping Attack



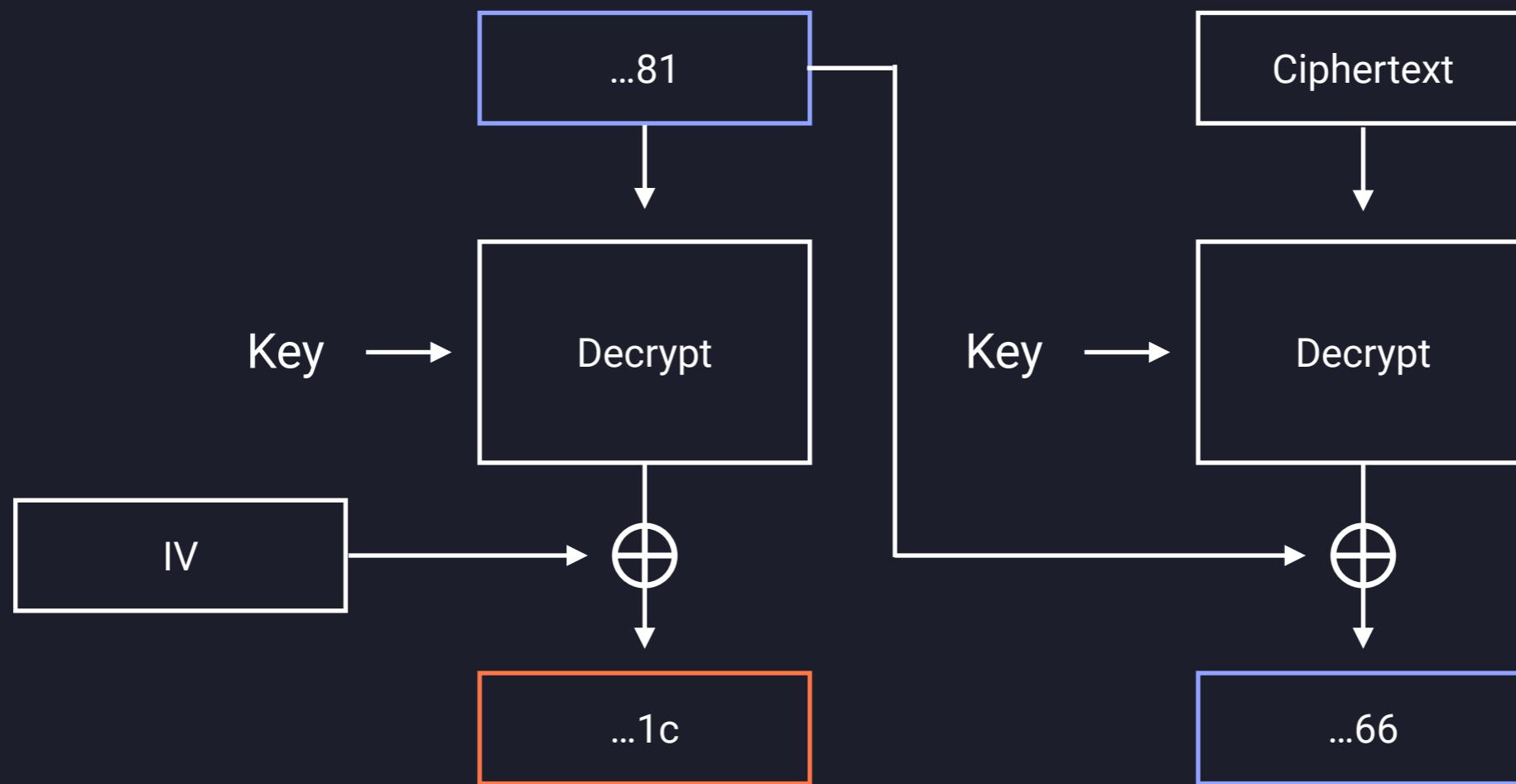
CBC Mode / Bit-Flipping Attack



CBC Mode / Bit-Flipping Attack



CBC Mode / Bit-Flipping Attack



將解密的明文改成我們指定的 0x66

CBC Mode / Bit-Flipping Attack

- 透過修改 IV 和 Ciphertext 來控制 Plaintext
- 其他 CFB, OFB, CTR 也存在相同的攻擊方式

CTF Challenges

- [AIS3 preexam 2018 - EFAIL](#)
- [AIS3 preexam 2018 - BLIND](#)
- [AceBear CTF - CNVService](#)
- [Teaser Dragon CTF 2018 - AES-128-TSB](#)

CBC Mode / Padding Oracle Attack

The Oracle

- 使用 AES CBC Mode 配上 PKCS#7 Padding Scheme
- 伺服器能幫我們解密訊息
- 如果解密出來的訊息 Padding 錯誤會噴錯

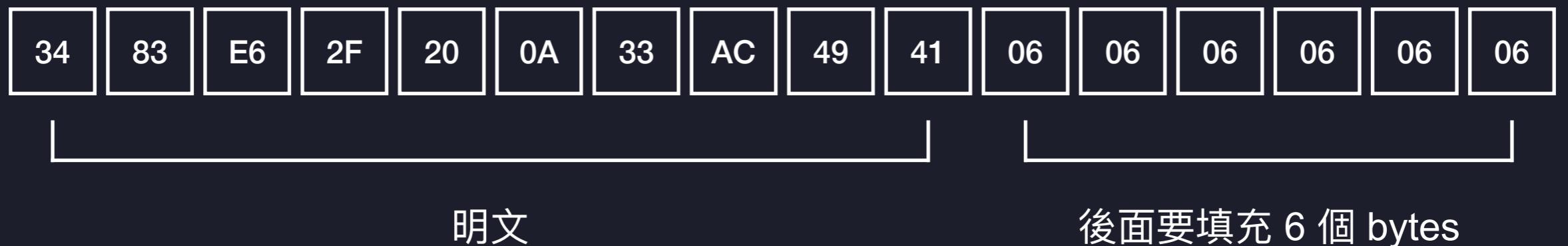
```
def unpad(data):  
    if not all([x == data[-1] for x in data[-data[-1]:]]):  
        raise ValueError  
    return data[:-data[-1]]  
  
def oracle(cipher):  
    aes = AES.new(KEY, AES.MODE_CBC)  
    try:  
        plain = unpad(aes.decrypt(cipher))  
    except ValueError:  
        return False  
    return True
```

PKCS#7

PKCS#7 : Cryptographic Message Syntax

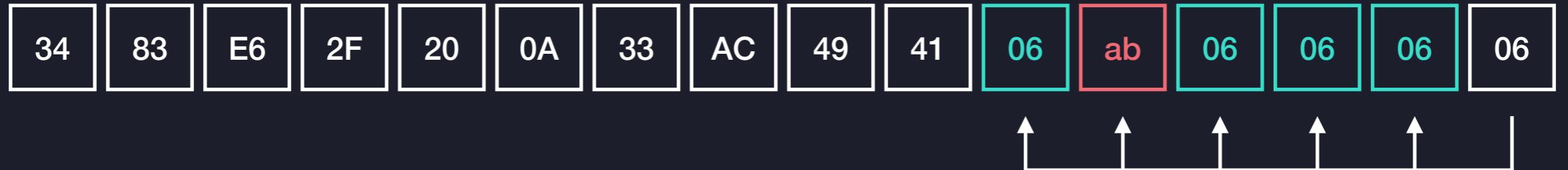
<https://tools.ietf.org/html/rfc2315>

- 這個標準裡面定義了一種 Padding 的格式
 - 要填充 5 個 bytes 就填充 5 個 0x05
 - 要填充 2 個 bytes 就填充 2 個 0x02



Padding 錯誤

- 怎麼樣會 Padding 錯誤？
- 抓最後一個 byte 就可以知道 Padding 長度



都要等於最後一個 byte

Python Implementation

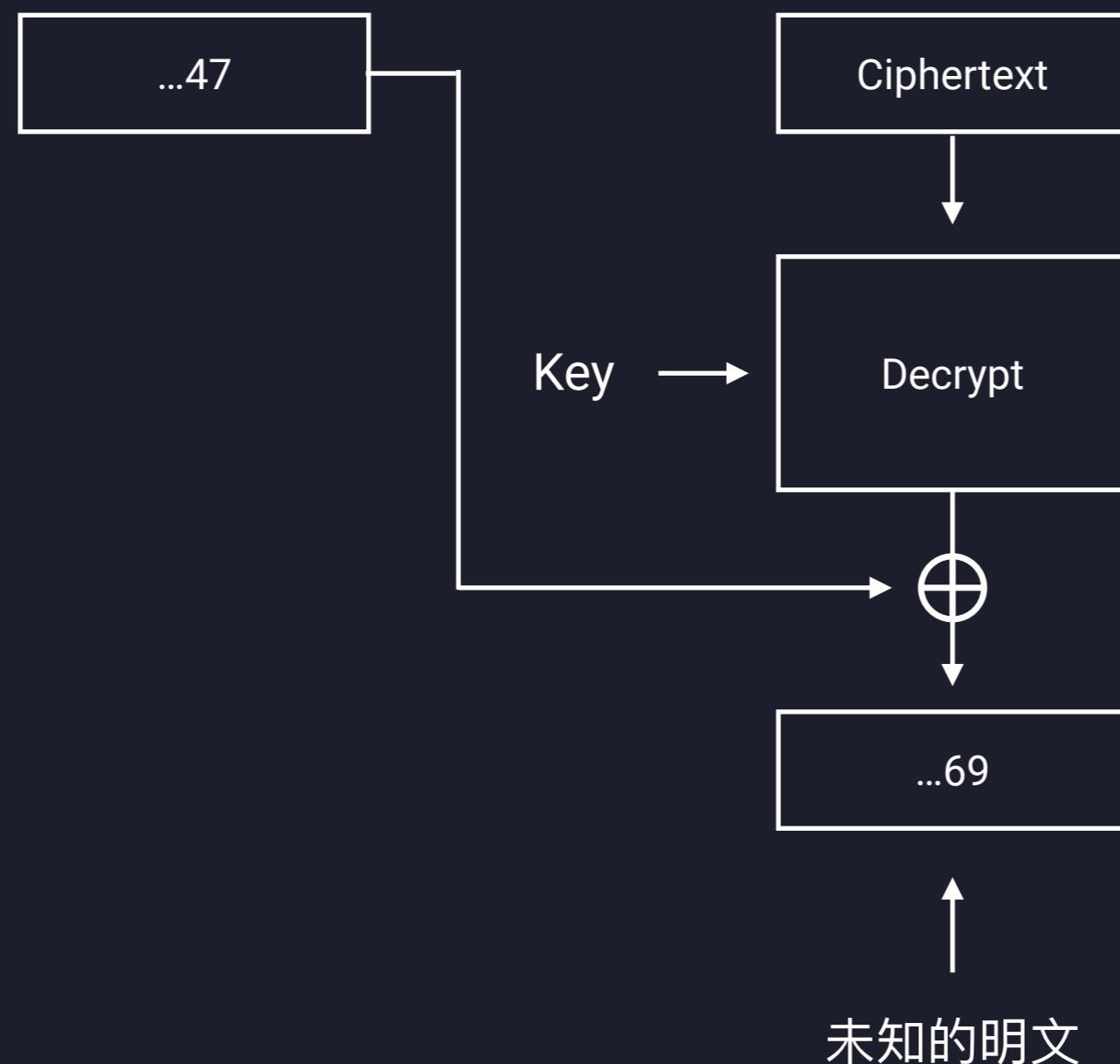
```
def pad(data):
    p = 16 - len(data) % 16
    return data + bytes([p]) * p

def unpad(data):
    if not all([x == data[-1] for x in data[-data[-1]:]]):
        raise ValueError
    return data[:-data[-1]]
```

解 plaintext[-1]

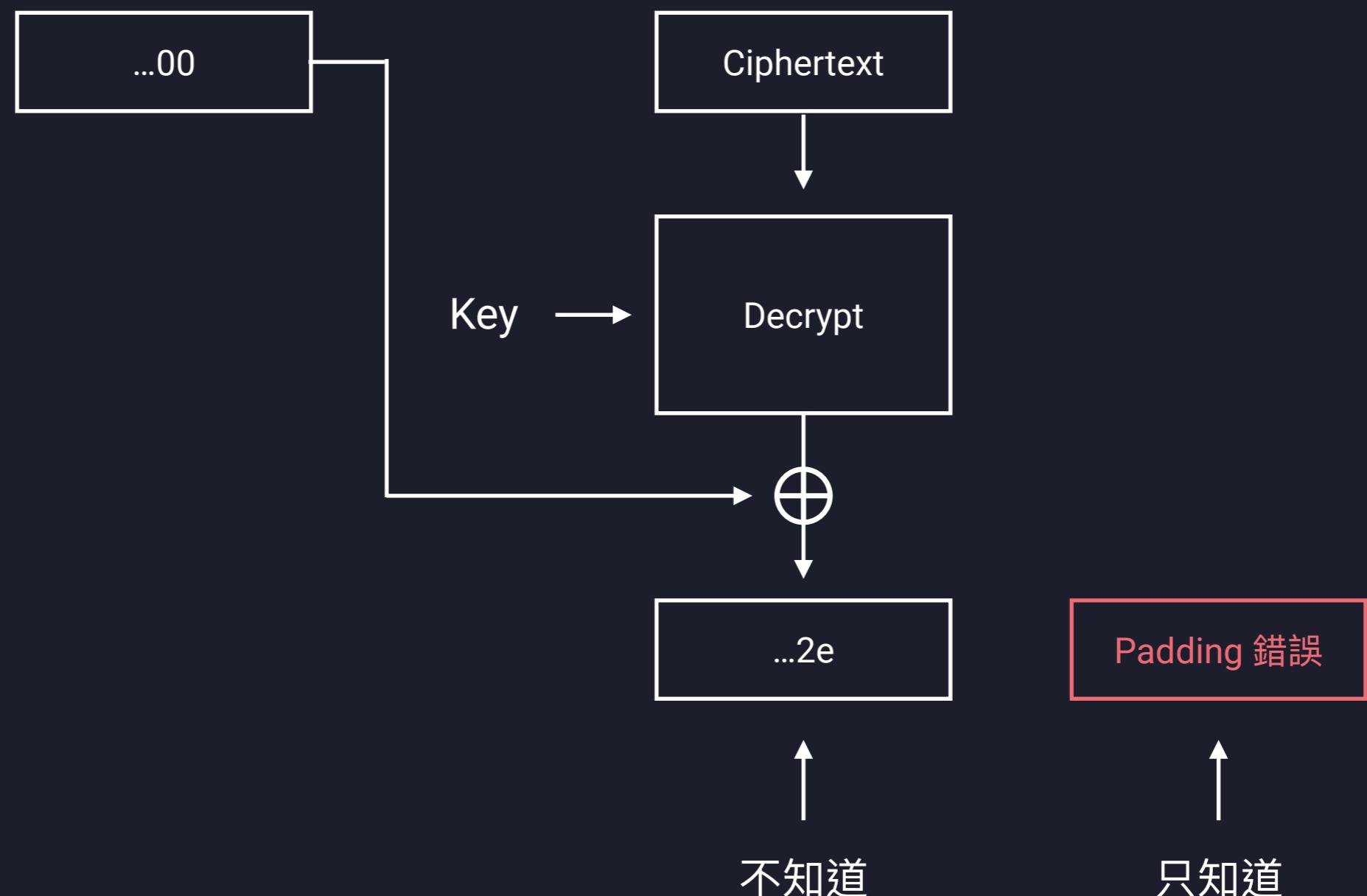
Padding Oracle Attack

已知的密文



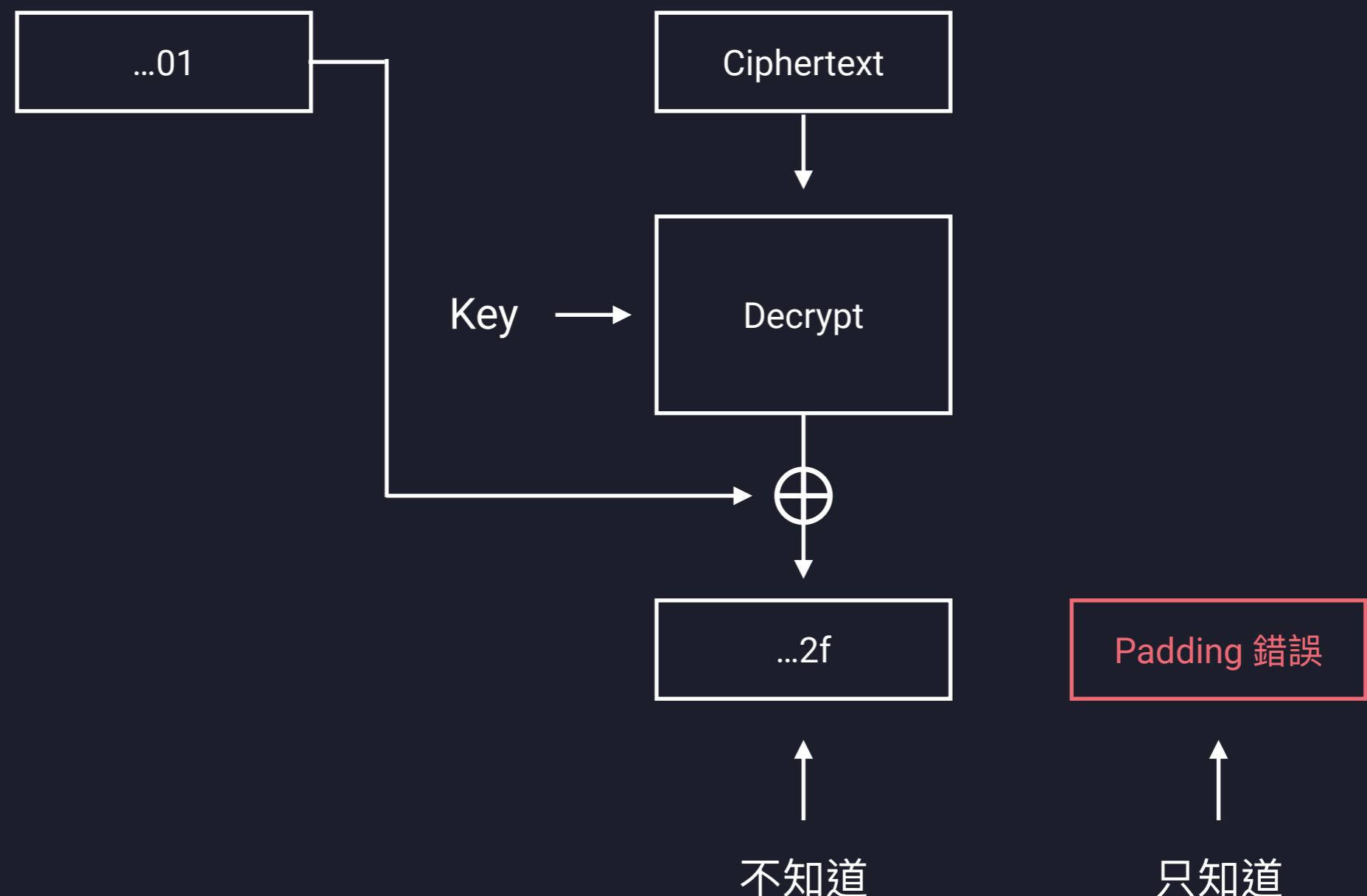
Padding Oracle Attack

暴力嘗試最後一個 byte



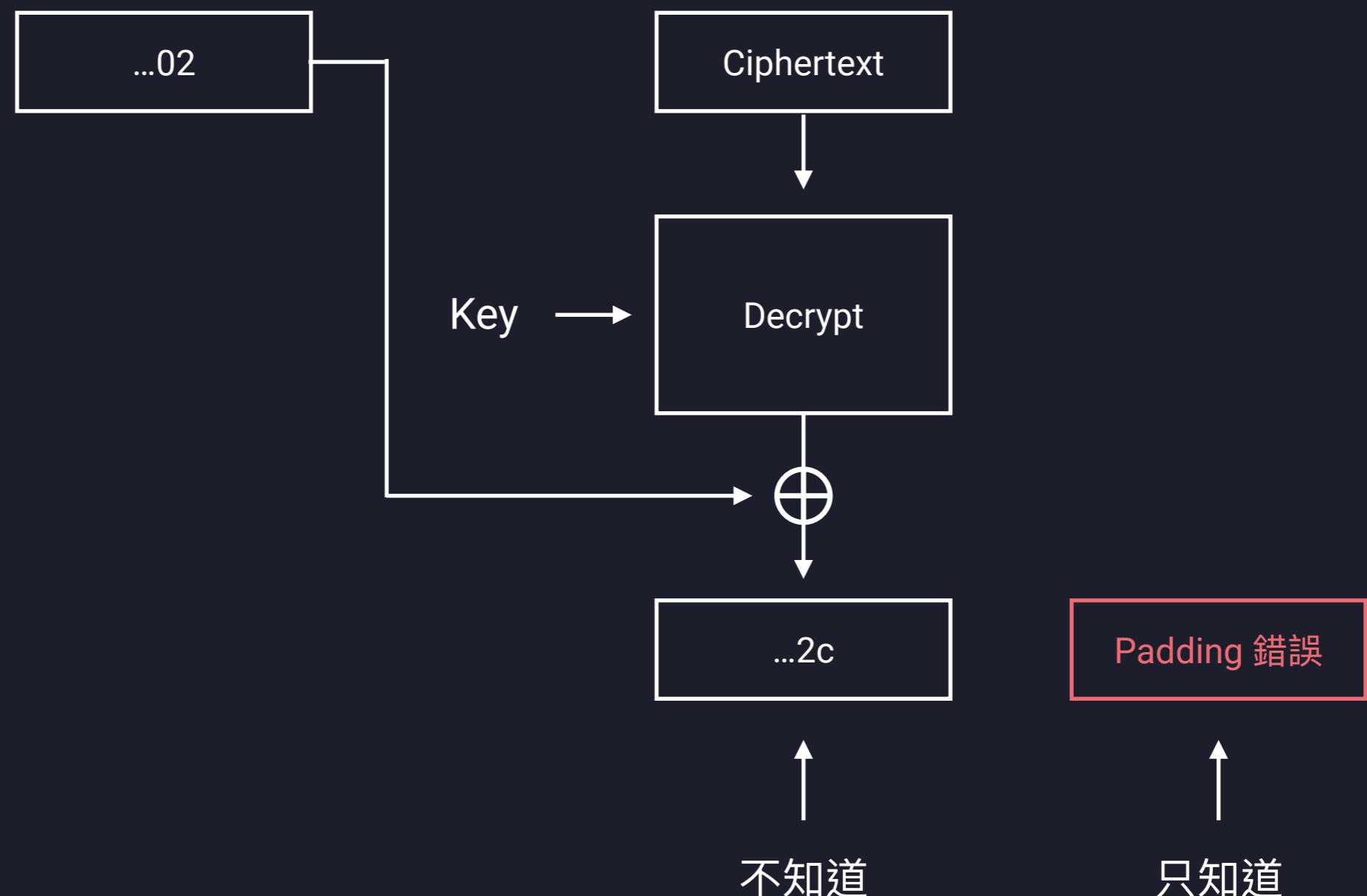
Padding Oracle Attack

暴力嘗試最後一個 byte



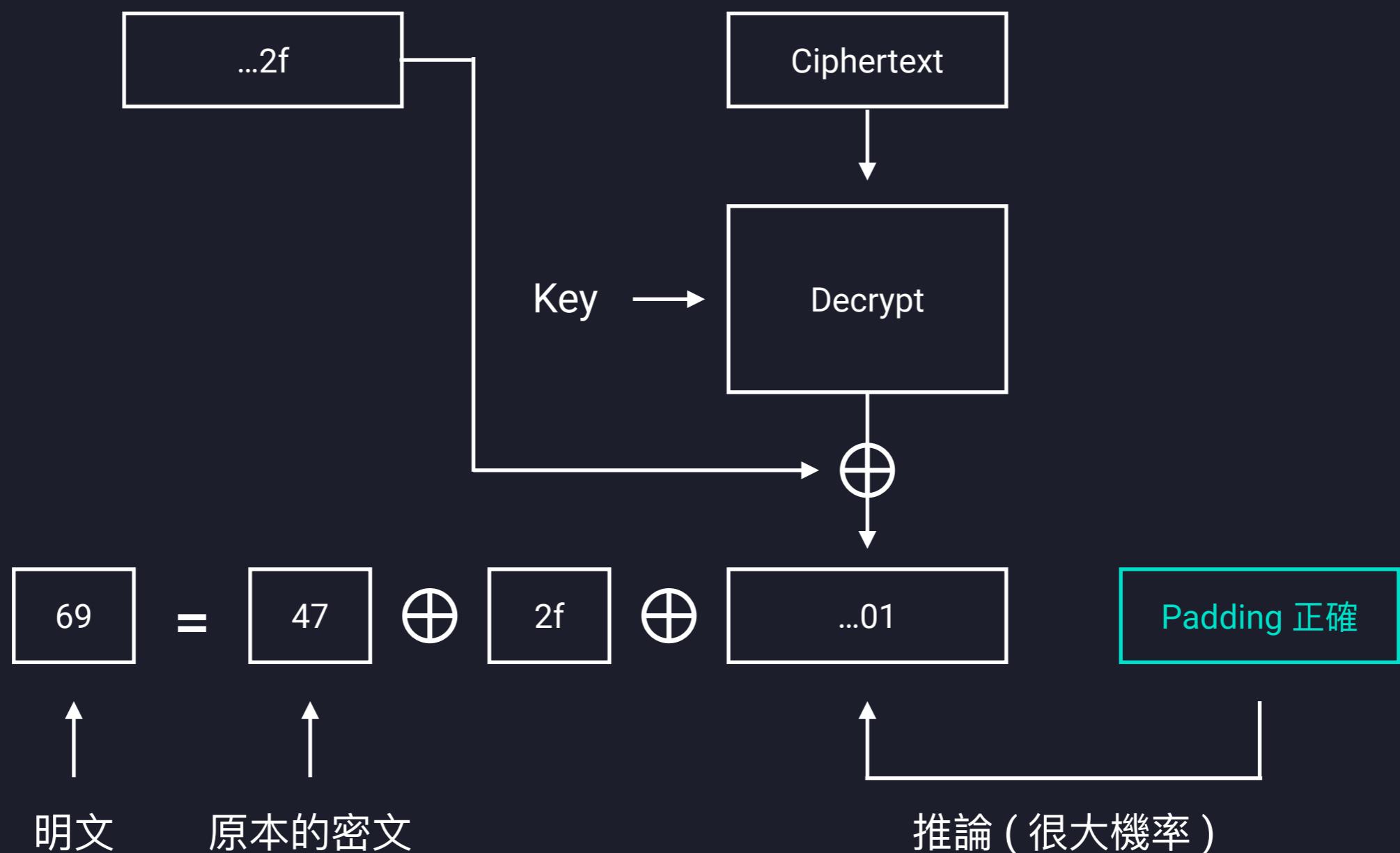
Padding Oracle Attack

暴力嘗試最後一個 byte



Padding Oracle Attack

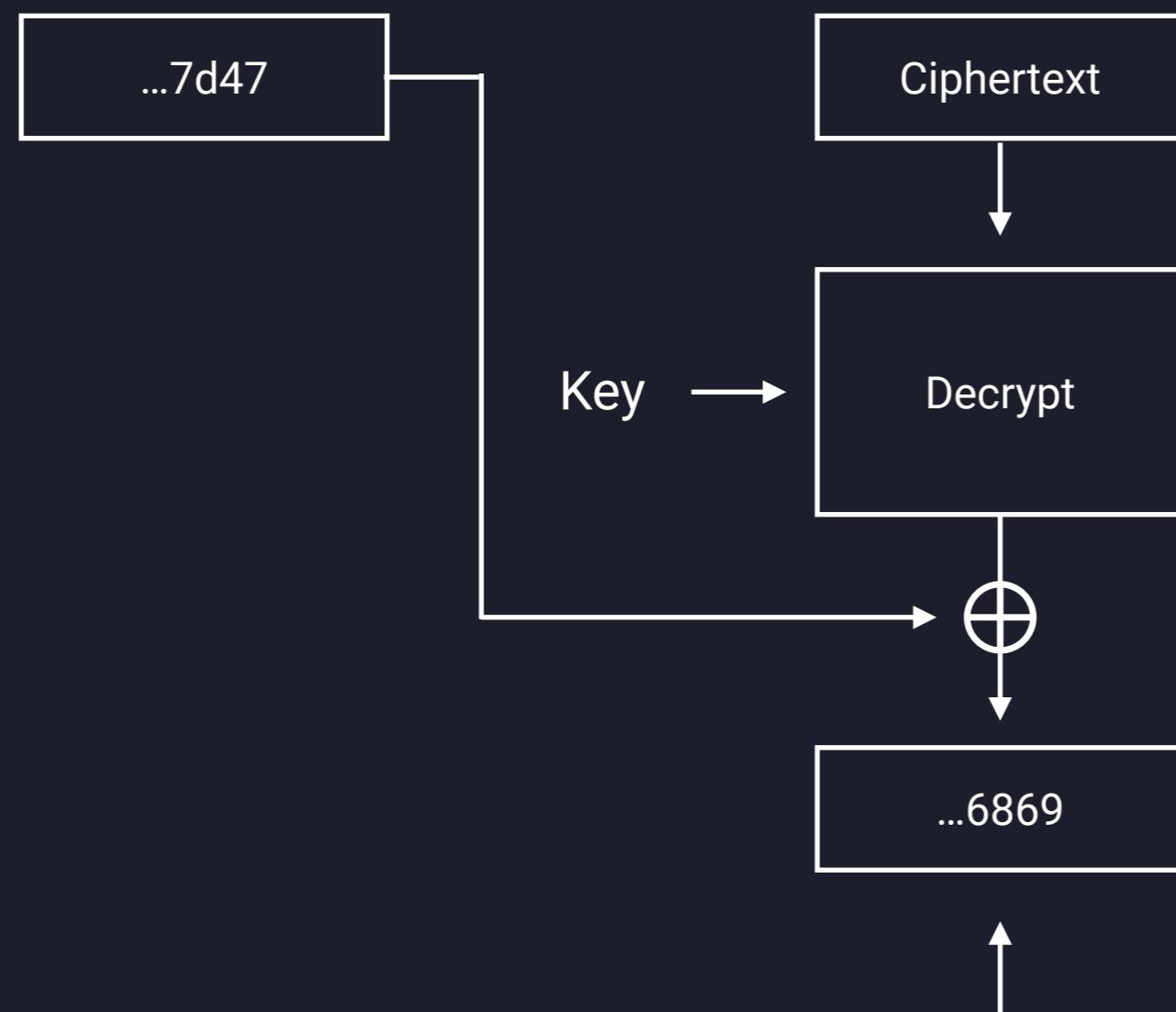
暴力嘗試最後一個 byte



解 plaintext[-2]

Padding Oracle Attack

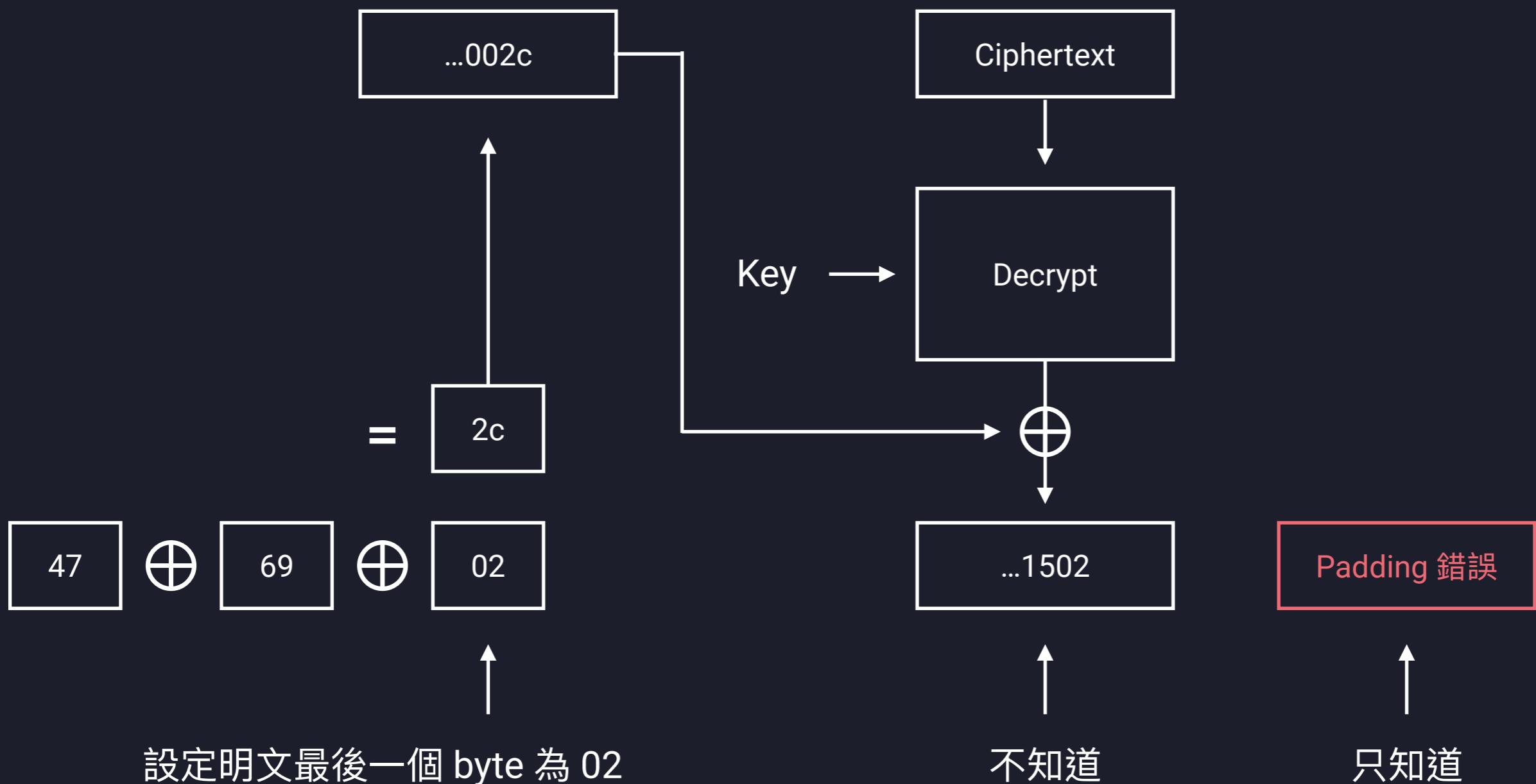
已知的密文



部分未知的明文

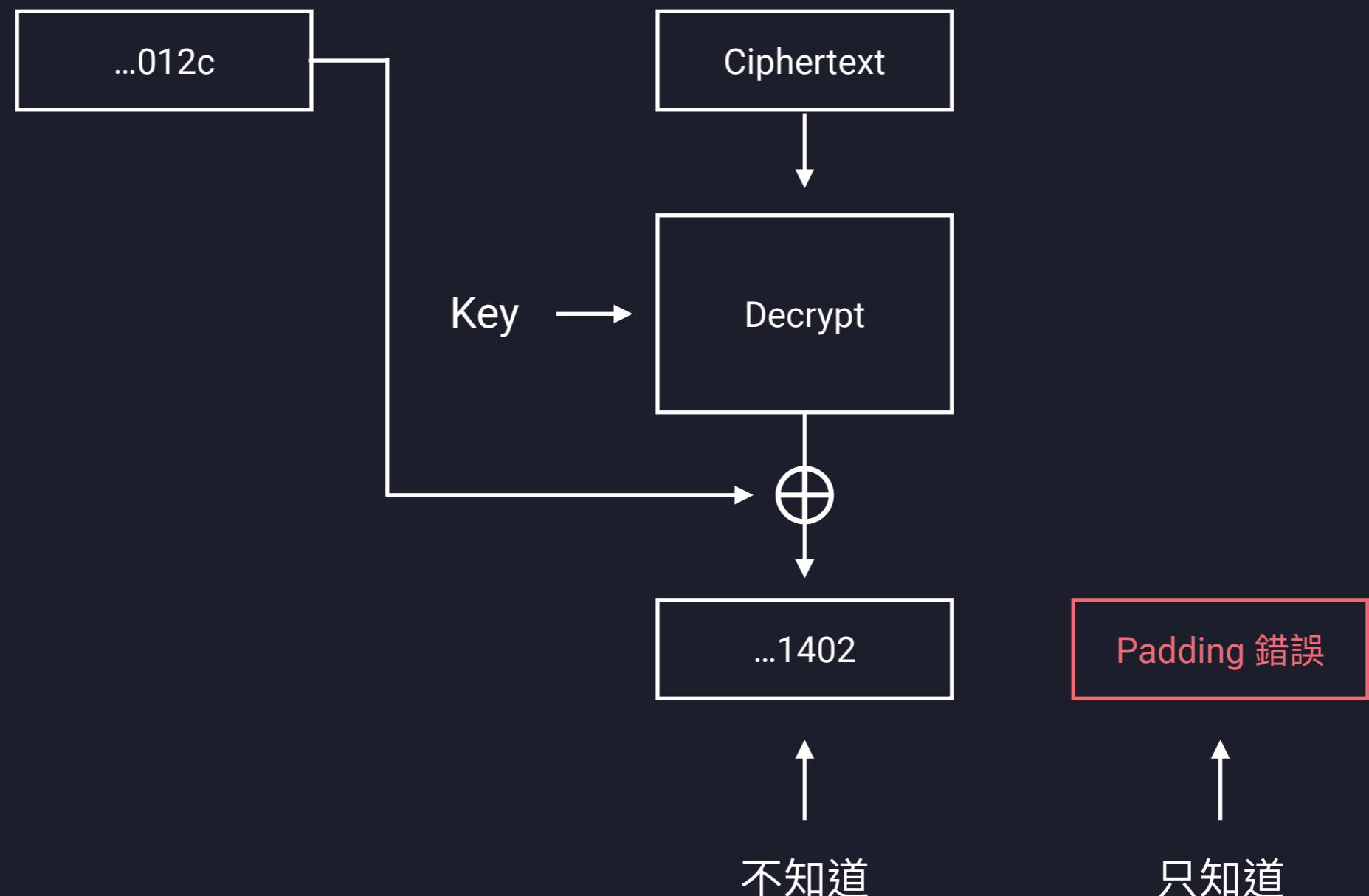
Padding Oracle Attack

暴力嘗試倒數第二個 byte



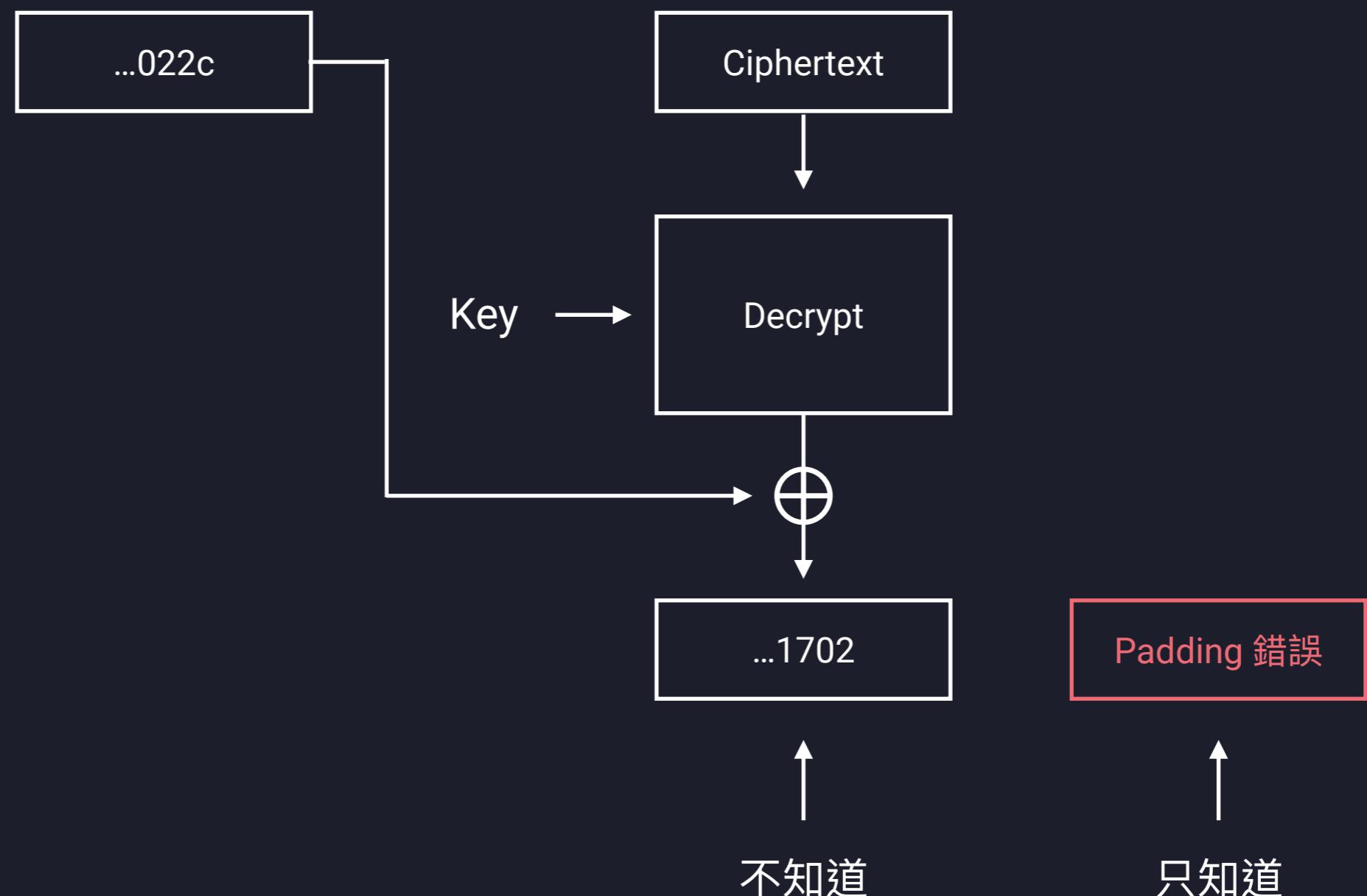
Padding Oracle Attack

暴力嘗試倒數第二個 byte



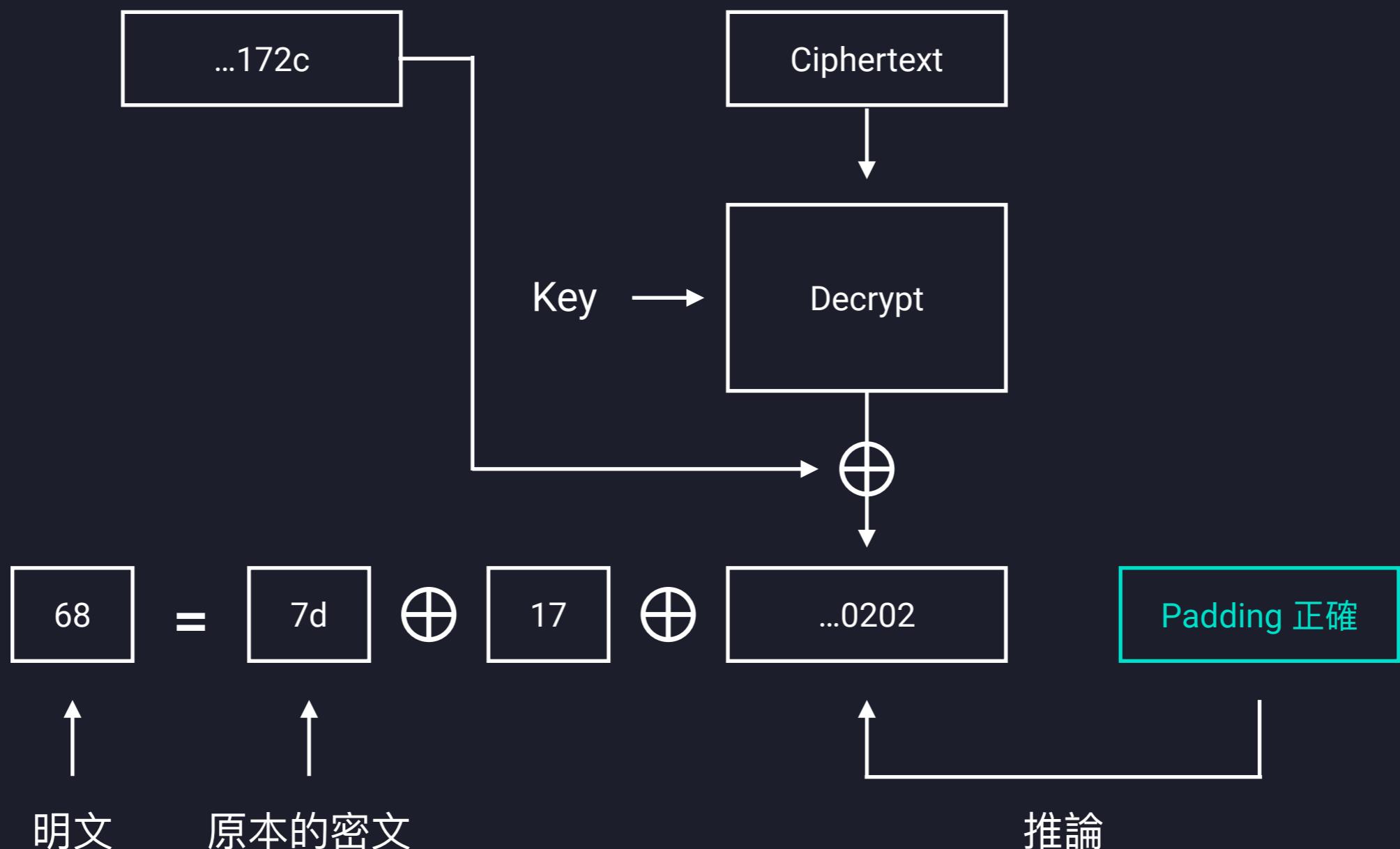
Padding Oracle Attack

暴力嘗試倒數第二個 byte



Padding Oracle Attack

暴力嘗試倒數第二個 byte



Summary

- 總共有三層迴圈
 - 猜 0 - 255 直到猜出一個 byte
 - 一次解出一個 byte 直到解完一個 block
 - 一次解出一個 block 直到解完所有 blocks
- 解出一個 block 最多需要 4096 次嘗試

CTF Challenges

原汁原味 padding oracle attack :

- [CSAW CTF 2016 Quals - Neo](#)
- [HITCON CTF 2016 Quals - Hackpad](#)
- [BAMBOOFOX CTF 2018 - mini-padding](#)

padding 相關攻擊技巧 :

- [HITCON CTF 2017 Quals - Secret Server](#)
- [HITCON CTF 2017 Quals - Secret Server Revenge](#)
- [BAMBOOFOX CTF 2018 - baby-lea-revenge](#)
- [BAMBOOFOX CTF 2018 - baby-lea-impossible](#)

GCM Mode / Forbidden Attack

GCM Mode / Forbidden Attack

- 又叫 Nonce Repeating Attack
- 也就是當 IV 重複用的情況
- 假設可以拿到任意明文加密的結果
- 這個攻擊可以做到偽造簽章

GCM Mode / Forbidden Attack

- 在 Authentication 中做的 xor 就是 GF(2^{128}) 中的加法
- Tag 可以表示成在 GF(2^{128}) 中的式子如下

$$\begin{array}{c} \text{Tag} \quad \boxed{\quad} \quad \text{Ciphertext} \\ \cdot \quad T = A\text{H}^4 + C_1\text{H}^3 + C_2\text{H}^2 + L\text{H} + E_k(J_0) \\ \text{Associated Data} \quad \boxed{\quad} \quad [len(A)]_{64} \parallel [len(C)]_{64} \end{array}$$

未知的只有藍色的部分

GCM Mode / Forbidden Attack

- 做兩次加密拿到兩個 Tag
 - 兩個相減後只剩 H 未知，解方程式求根可得 H
 - 有 H 帶回原式可得 $E_k(J_0)$ 就可以偽造 Tag 了
-
- $T_1 = A_1H^4 + C_{11}H^3 + C_{12}H^2 + L_1H + E_k(J_0)$
 - $T_2 = A_2H^4 + C_{21}H^3 + C_{22}H^2 + L_2H + E_k(J_0)$
 - $T_1 - T_2 = (A_1 - A_2)H^4 + (C_{11} - C_{21})H^3 + (C_{12} - C_{22})H^2 + (L_1 - L_2)H$

CTF Challenges

- [UTCTF 2020 - Galois](#)
- [VolgaCTF 2018 Quals - Forbidden](#)