密碼學(Cryptography)

恩，密碼學

Outline

- 前言（Intro）
- 編碼（Encode）
- 古典密碼學（Classical）
- 雜湊（Hash）
- 流密碼（Stream）
- 對稱式密碼（Symmetric）

前言（Intro）

我是誰？

- ccbeginner
- 之前是打競程的
- 負責雜耍
- 我知道的不多，但密碼學…不用知道太多
- https://www.instagram.com/ccbeginner/

想學密碼學的原因？

- 喜歡破解密碼
- 喜歡研究數學
- 不喜歡，但學就對了

不想學密碼學的原因？

- 沒興趣
- 滿滿數學
- 不想寫程式

對了推這個練習資源：
https://cryptohack.org/

課程安排：Crypto 1

- 編碼
- 古典密碼
- 穿插「一」點數學
- 流加密
- 對稱式加密

課程安排：Crypto 2

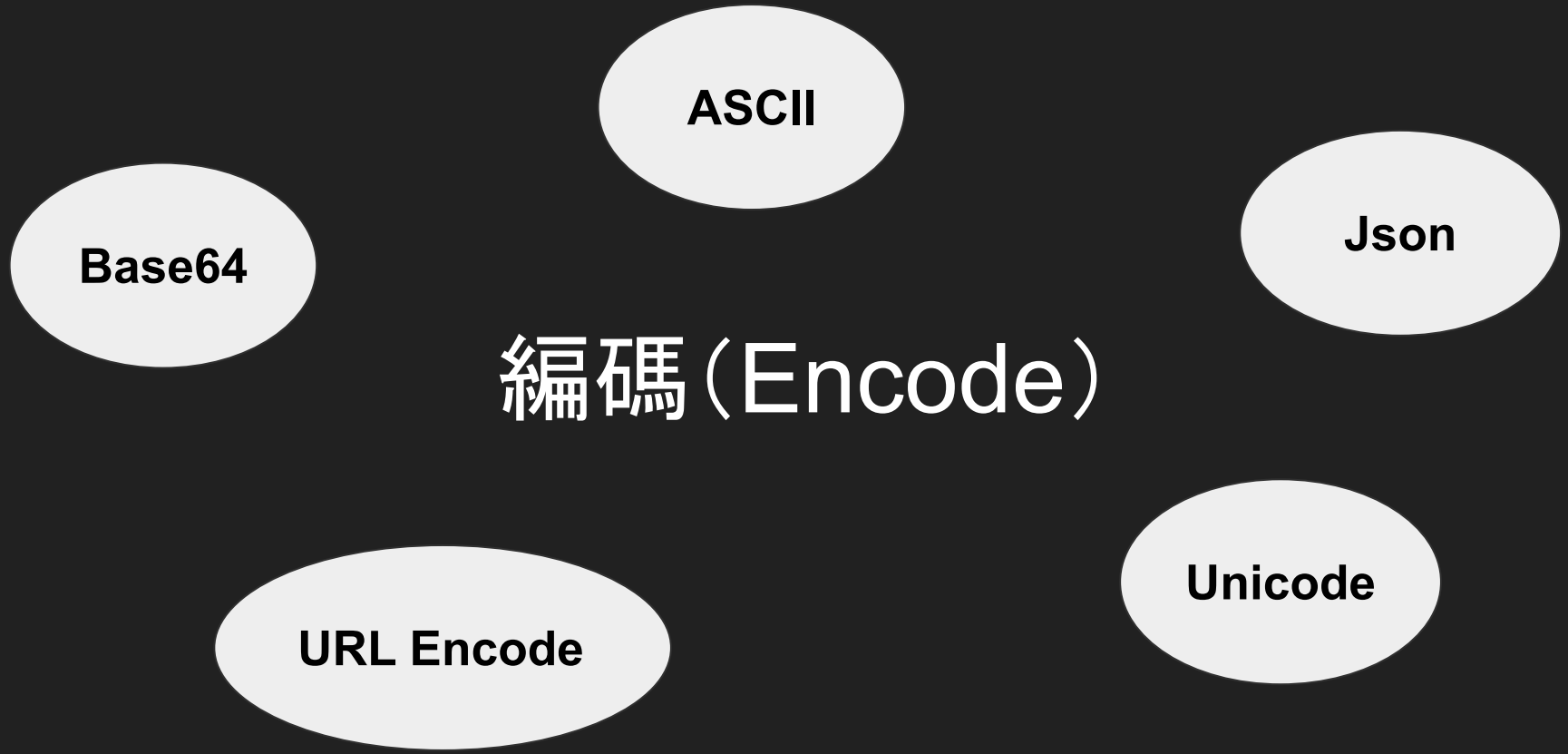
- 「億」點數學
- HASH
- 非對稱式加密

Python安裝了嘛

那我們開始嘍～！

對了先安裝個套件

pip3 install -U PyCryptodome


ASCII
Base64
Json
編碼（Encode）
URL Encode
Unicode

什麼是編碼？

「編碼器（英語：Encoder）是一種將資訊由一種特定格式轉換為其他特定格式的感測器、軟體或是演算法，轉換的目的可能是由於標準化、速度、保密性、保安或是為了壓縮資料。」
——維基百科

簡而言之，編碼的概念是

舉個栗子

目的:
儲存文字

轉換訊息的方法:
ASCII, Unicode, UTF-8,
GB2312, Big5, Shift-JIS…..

舉個栗子

目的:
方便傳輸

轉換訊息的方法:
Base64, URL Encode,
MessagePack……

常用文字編碼

ASCII（American Standard Code for Information Interchange）:

- 1個byte來表示字母、數字、標點符號和一些控制字符。
- 最初針對英語發明的。
- Unicode字符集的最初128個字符與ASCII相同，這使得ASCII字符在Unicode中有相同的編碼。

UTF-8（Unicode Transformation Format - 8-bit）:

- 使用1到4個bytes來表示字符。
- 容納了各國語言。是Unicode的一種實現。
- ASCII字符使用一個位元組表示，而其他字符使用更多位元組。ASCII能表示的，用UTF-8編碼是一樣的結果。

常用數據編碼

HEX (16進制編碼):

- 方便表達二進位數字的方法，使用0~9, 字母 A~F 表達數字

Base64:

- 基於64 個可列印字元來表示二進位資料的表示方法，常見編碼結果後面有 =,==

關於編碼

種類太多了，講不完，也沒必要講完

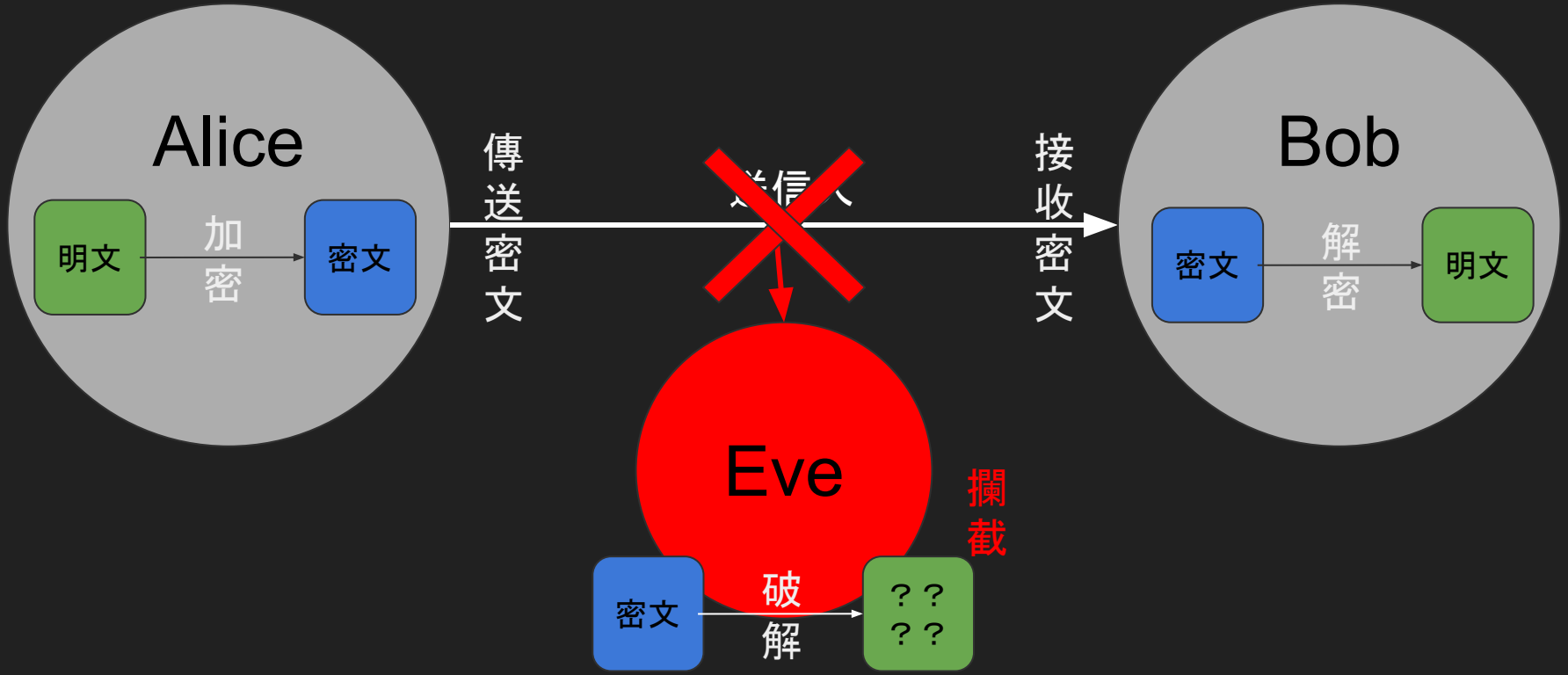
有需要再查查那是啥，大都能查到工具來轉換

LAB Time.

(Complex Encoder,F**k Encoder)

古典密碼學（Classical）

加解密的用途



古典加解密的方法

- 凱薩密碼
  - ROT13
  - 曹操密碼
- 簡易替換密碼
- 維吉尼亞密碼
- 波雷費密碼(Playfair)
- 柵欄密碼(Rail fence)

凱薩密碼(Caesar)

假設密鑰，也就是偏移量=3，字母表位移3格

明文字母表：ABCDEFGHIJKLMNOPQRSTUVWXYZ

密文字母表：DEFGHIJKLMNOPQRSTUVWXYZABC

明文：NCKUCTF

密文：QFNXFWI

ROT13

- 明文字母表
  :ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 密文字母表
  :DEFGHIJKLMNOPQRSTUVWXYZABC
- 其實就是凱薩偏移量13

曹操密碼

- 明文字母表
  :ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 密文字母表
  :ZYXWVUTSRQPONMLKJIHGFEDCBA
- 就是反轉字母表而已

簡易替換密碼（Simple Substitution cipher）

把密文字母表打亂

明文字母表：ABCDEFGHIJKLMNOPQRSTUVWXYZ

密文字母表：EBDYARUNKMGQPWSOHXLCVFZJTI

明文：NCKUCTF

密文：WDGVDCR

如何破解替換式密碼？

暴力法：枚舉所有26!種可能性？

頻率分析(Frequency analysis)

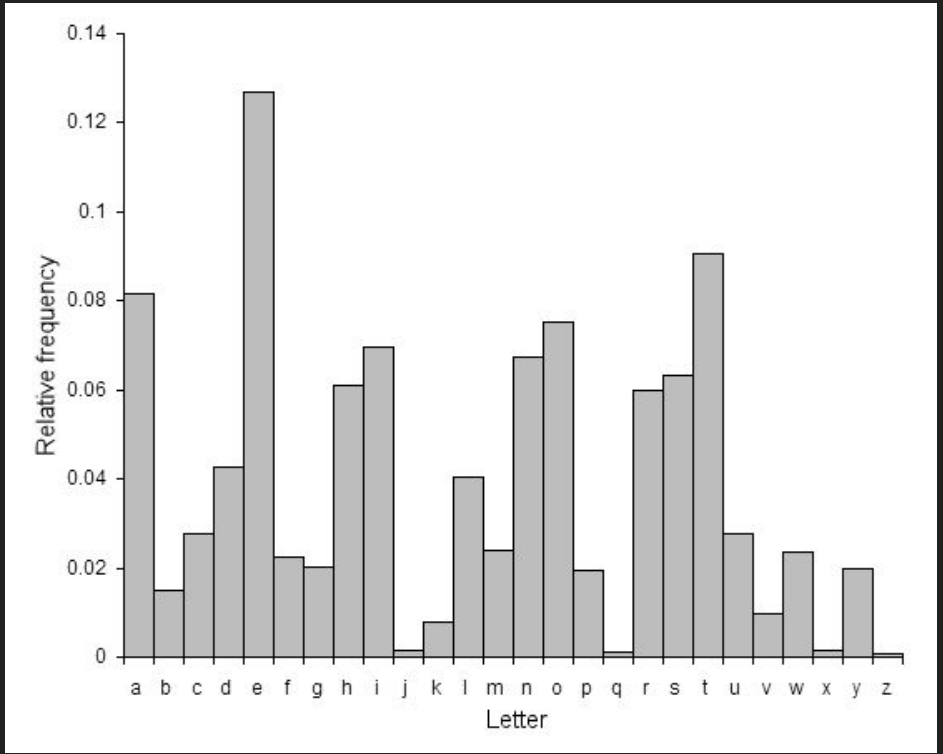
一般的文章中，把每個字母的

頻率抓出來，長這樣 =>

然後一一對應密文的頻率，

可以大概解出

破解：https://quipqiup.com/



維吉尼亞密碼(Vigenere)

- 多幾個偏移量，輪流用，假設 n=[3,10,16,9,22]，對應字母：DKQJW
- 明文：NCKUCTF
- 密文：QMADYWP
- 密碼表：

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

如何破解維吉尼亞？

頻率分析...好像不太對（偏移量不一樣）

卡希斯基實驗（Kasiski examination）

利用常見的英文單字 (例如 the) 來推算**密鑰長度**

密鑰：ABCDABCDABCDABCDABCDABCDABCD

明文：CRYPTOISSHORTFORCRYPTOGRAPHY

密文：CSASTPKVSIQUTGQUCSASTPIUAQJB

距離 16 個字母，可以得知密鑰長度為 16 的因數

把密文看成多次凱薩加密的結果，最後用**頻率分析**即可

現成解法：https://www.mygeocachingprofile.com/codebreaker.vigenerecipher.aspx

波雷費密碼(Playfair)

字母兩兩一組，如果

- 同一橫排:取右邊
- 同一直排:取下面
- 都不同排:橫座標對調

密碼表:=>

P	L	A	Y	F
I	R	E	X	M
B	C	D	G	H
K	N	O	Q	S
T	U	V	W	Z

明文:NCKUCTF => (NC KU CT FX)

密文:UNNTBUYM

1914 被 Joseph Mauborgne 破解

柵欄密碼（Rail fence）

把字母排成柵欄的樣子，然後橫排寫過去

明文：NCKUCTF IS WONDERFUL

N___C___S___D___U_

_C_U_T_I_W_N_E_F_L

__K___F___O___R___

密文：NCSDUCUTIWNEFLKFOR

暴力破解：枚舉柵欄的數量（柵欄數量<=明文長度）

LAB Time.
(替換加密、維基尼亞加密)

小白都須知道的「一」點點數學

XOR

邏輯運算的單元, 常常用在密碼學領域

對了, 0=False, 1=True

重要性質:

- 滿足 **交換律** 跟 **結合律**
- XOR 是自己的反運算子
- 運算很快

p	q	$p \oplus q$
True	True	False
True	False	True
False	True	True
False	False	False

MOD 模運算

模運算，就是除以某個數字之後取餘數，在整數域做運算，密碼學很常用。

如果a,b除以m的餘數相同，寫作 a ≡ b (mod m)

MOD 性質：

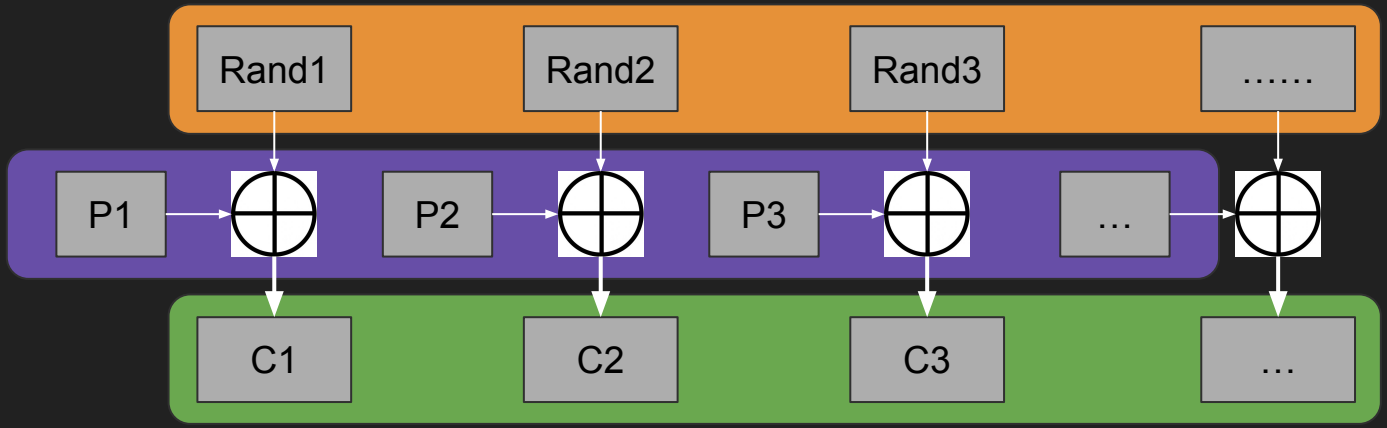
- 加、減、乘都跟暴通得四則運算差不多
- 除法需要取模反元素，python code: pow(a, -1, m)
- 如果 a ≡ 0 (mod m)，代表 a 是 m 的倍數
- 運算較慢

LAB Time.
(xor warmup)

流密碼（Stream cipher）

流密碼(Stream cipher)

- 像流水一樣，一次加密一個bit或是一個byte
- 用一個亂數器生成許多數字來加密，亂數器是重點
- 密鑰通常是亂數器的 seed



流密碼Outline

Outline:

- 偽隨機數 PRNG
- 線性同餘生成器 LCG
- 反饋位移生成器 FSR

偽隨機數 PRNG

- 用計算的方法得出隨機的數字
- 通常有一個 **seed**（初始值），跟一個 **數學函數** 來生成一個數列
- 是密碼學領域常用的

真隨機數

- 利用噪音、放射性衰變之類，難以預測的東西來產生數字
- 是真正的隨機

偽隨機數

- 知道了seed，基本上就知道整個數列了
- 函數大都具備「有限狀態空間」，當狀態開始重複時，數列開始循環
- 週期：對於所有可能的 seed，生成的數列循環長度最小的
- 例子：

$$X[n] = 2 * X[n-1] + 5 \bmod 23$$

狀態（X[n-1]）只可能是0~22，可知循環長度<=23

$$X[n] = 2 * X[n-1] + 3 * X[n-2] + 5 \bmod 23$$

狀態（X[n-1], X[n-2]）只可能是(0~22, 0~22)，可知循環長度<=23^2=529

偽隨機數可能的問題

在某些情況下，產生的隨機數列的週期會比較小。

連續值之間關聯密切，可用一部分的值，來推算其他部份的值。

輸出序列的值的大小不均勻。

線性同餘生成器 LCG (Linear congruential generator)

$$N_{j+1} \equiv (A \times N_j + B) \pmod{M}$$

公式很簡單, 就長這樣。

* 符號 ≡ 意思差不多是等於, mod 是取餘數的意思
* 就是 ≡ 的左右兩邊都對 M 取餘數時, 等式會成立

線性同餘生成器 LCG (Linear congruential generator)

如果知道一些連續的X {X0,X1,X2,X3,X4,...}:

可以試著解出參數

先試著解出 M:

$X1 \equiv A * X0 + B \pmod{M}$

$X2 \equiv A * X1 + B \pmod{M}$

兩式子相減，消掉 B:

$X2-X1 \equiv A * (X1-X0) \pmod{M}$

$X3-X2 \equiv A * (X2-X1) \pmod{M}$

線性同餘生成器 LCG (Linear congruential generator)

如果知道一些連續的X {X0,X1,X2,X3,X4,...}:

接著 上面式子乘 (X2-X1)，下面式子乘 (X1-X0)，兩式相減，消掉 A:

$$(X3-X2) * (X1-X0) - (X2-X1) * (X2-X1) \equiv 0 \pmod{M}$$

$$(X4-X3) * (X2-X1) - (X3-X2) * (X3-X2) \equiv 0 \pmod{M}$$

…

然後我們就有很多個 M 的倍數，取 gcd (最大公因數)就很容易找到M

線性同餘生成器 LCG (Linear congruential generator)

如果知道一些連續的X {X0,X1,X2,X3,X4,...}:

找到 M 之後可以找 A 了:

$X2-X1 \equiv A * (X1-X0) \pmod{M}$

$\Rightarrow A = (X2-X1) * \text{mod\_inverse}((X1-X0)) \pmod{M}$

最後找到 B:

$X1 \equiv A * X0 + B \pmod{M}$

$B = X1 - A * X0 \pmod{M}$

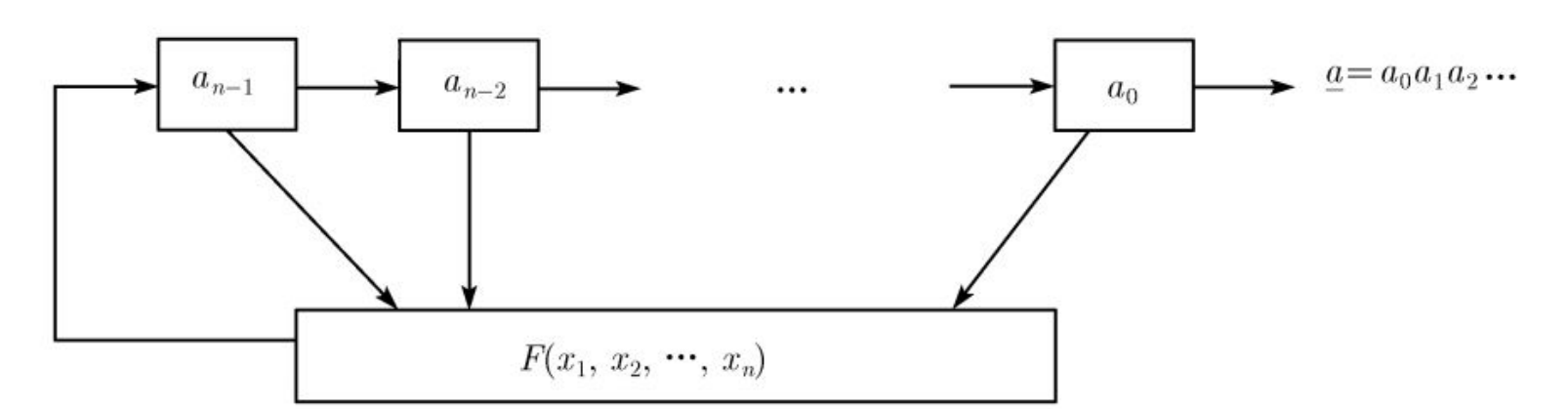
其他常見的方法

- PCG 置換同餘生成器
  - LCG的優化，更短的code更出色的表現
- Mersenne twister 梅森旋轉算法
  - 改變加數
  - 週期很長，甚至接近 cpu週期
  - MT19937
- XorShift 非常快的算法
  - v ^= v << a
  - v ^= v >> b
  - v ^= v << c

反饋位移生成器 FSR

包含一個數學函式 F，初始狀態是前 n 項 {a[0], a[1], ..., a[n-1]}

FSR 是個利用前 n 像去推導下一項的生成器



線性反饋位移生成器 LFSR

就是那個 F 是線性函數

$$a_{i+n} = \sum_{j=1}^{n} c_j a_{i+n-j}$$

如果知道其中連續的2n 項，可以用高斯消去解出所有參數

a[n+1] = c[1]a[n] + c[2]a[n-1] + … + c[n]a[1]

a[n+2] = c[1]a[n+1] + c[2]a[n] + … + c[n]a[2]

a[n+3] = c[1]a[n+2] + c[2]a[n+1] + … + c[n]a[3]

…

a[2n] = c[1]a[2n-1] + c[2]a[2n-2] + … + c[n]a[n]

解聯立 :)

LAB Time.
(EOF-LFSR、EOF-almost baby prng, Easy LCG)

對稱密碼（Symmetric-key algorithm）

對稱式密碼

對稱式密碼 Outline

Confusion and defusion

演算法：DES, AES

Padding：0pad, PKCS#5 和 PKCS#7, Bit Padding…

分組模式：ECB, CBC, CTR…

常見攻擊：padding oracle, bit flipping

基本策略（Confusion & Defusion）

Confusion混淆：

混淆 **密文** & **密鑰** 之間的關係，使得難以靠密文推測密鑰

例如：SBox

Defusion擴散：

改變一點點**明文**會影響許多部份的**密文**

例如：置換、Shift

Feistel Network

K0~Kn是密鑰，F是Feistel函數，不須可逆

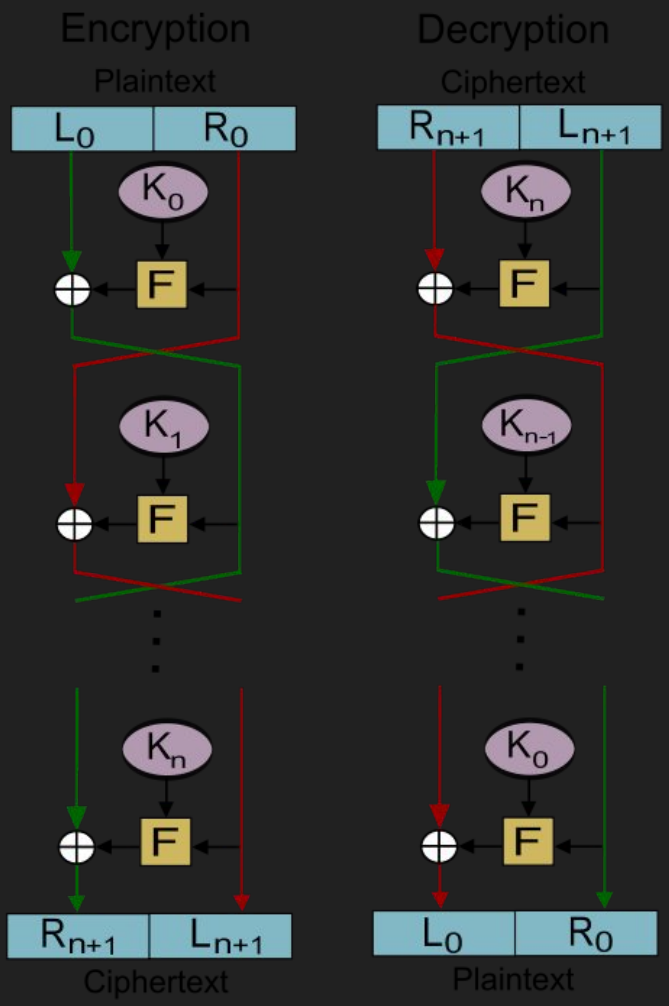
將明文塊拆分為兩個等長的塊，然後加密

對第 i 輪操作，i=1,2,3,...,n

$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus \mathrm{F}(R_i, K_i).$$

簡單好實做

* DES



DES (Data Encryption Standard)

使用 Feistel Network，總共 16 次迭代

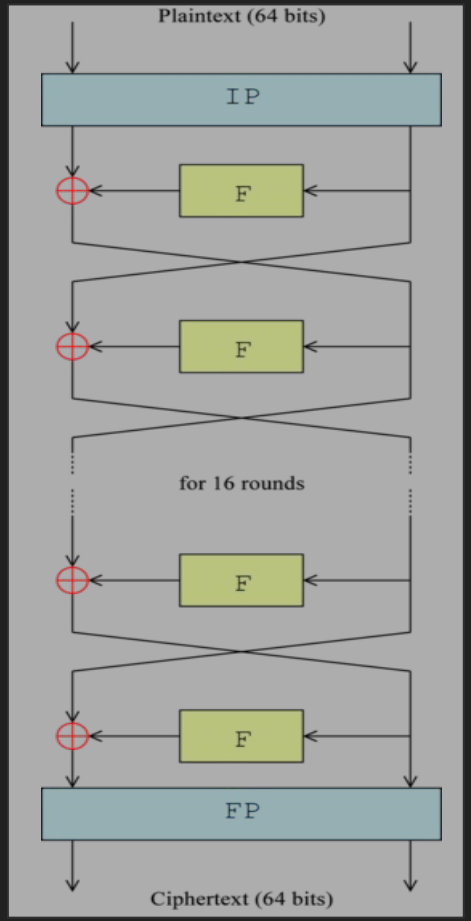
發明者：IBM 1977首次發布

key length: 64bits

block size: 64bits

IP(Initial Permutation)，就是先做一次置換

FP(Final Permutation)，也是IP的反置換



DES (Data Encryption Standard)

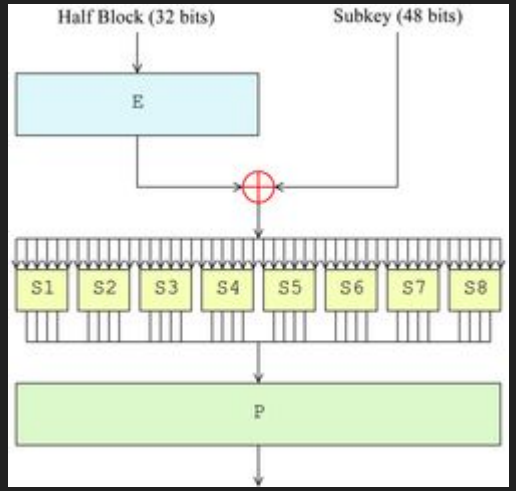
DES使用的F函數:

E(Expansion) 把那半塊括充成48bits

K(Key mixing) 然後跟 Subkey 做 XOR

S(Substitusion) 接著分成 8 個 6bits 大小的塊, 丟進S-box

P(Permutation) 重新排列組合



DES (Data Encryption Standard)

DES每一輪的的密鑰

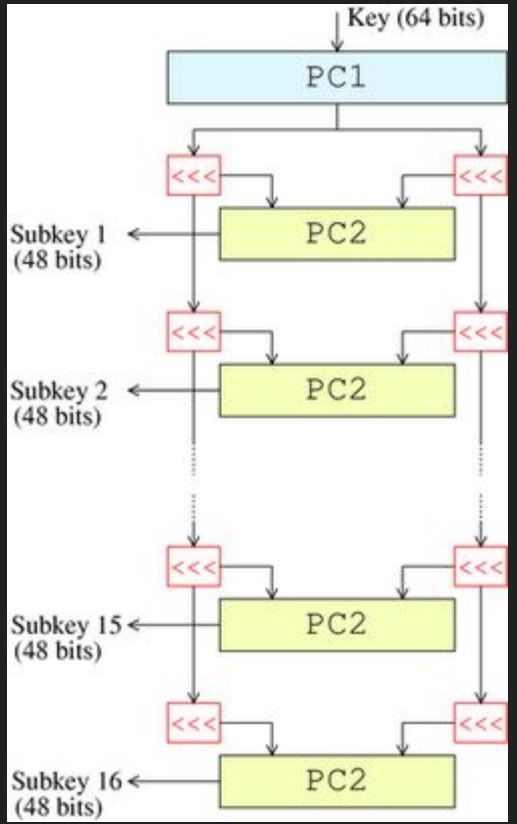
PC(Permutation Choice)

PC1 選出 56 位金鑰，然後分成兩半各 28 位

每一回合：

先將兩半金鑰左移1~2位

然後 PC2 選出 48 位子金鑰



3DES (Triple Data Encryption Standard)

＊ DES已能在一天內破解，所以出現了 3DES

使用3個key

加密過程：Cipher = Enc(Dec(Enc(Plaintext)))

解密過程：Plaintext = Dec(Enc(Dec(Cipher)))

Substitution-Permutation Network

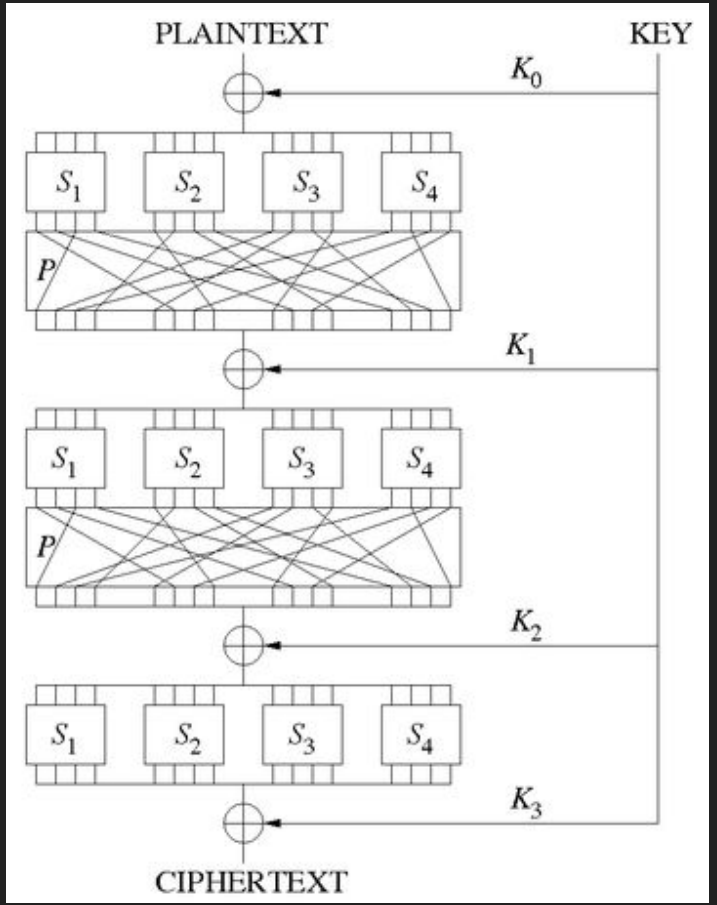
SPN，也叫SP-Network

S(Subtitusion) 裡面有好多SBox

P(Permutation) 置換

加密一共n輪

＊AES，Square, SHARK…



AES

使用 Substitution-Permutation Network，總共 16 次迭代

發明者：Vincent Rijmen、Joan Daemen

key length: 128, 192, 256bits

block size: 128bits

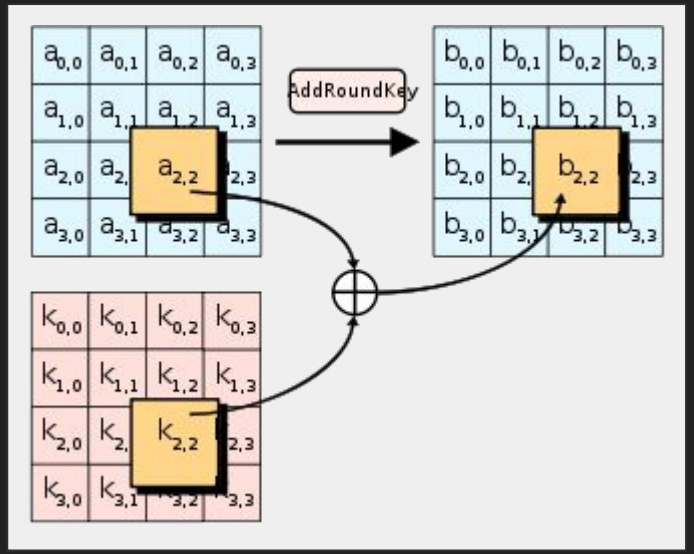
在4x4的矩陣上做運算

步驟：AddRoundKey，SubBytes，ShiftRows，MixColumns

AES

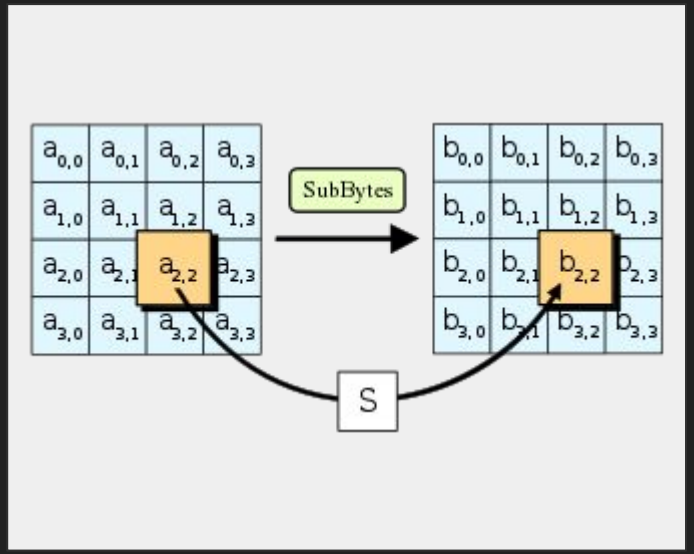
AddRoundKey

先跟 key 做 xor



SubBytes

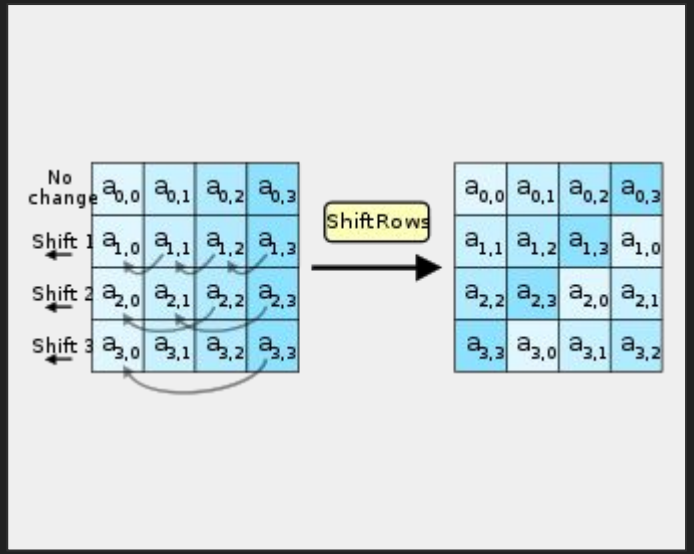
把給一格丟進 SBox



AES

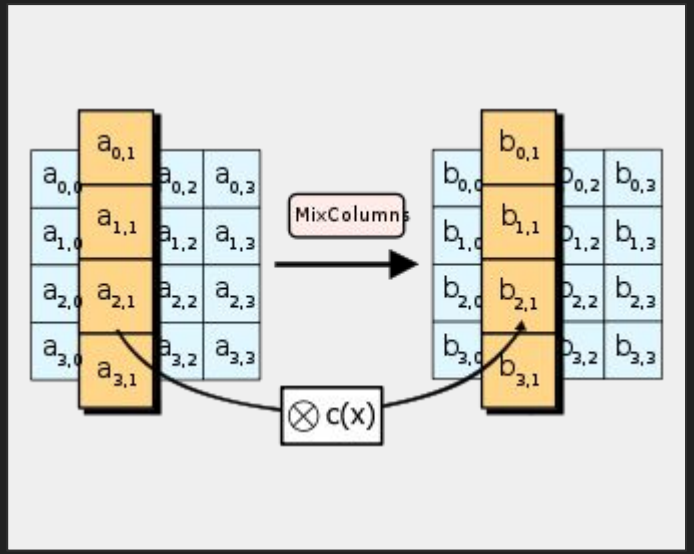
ShiftRows

第 n 橫排左移 n 格



MixColumns

多項式乘法



總得來說

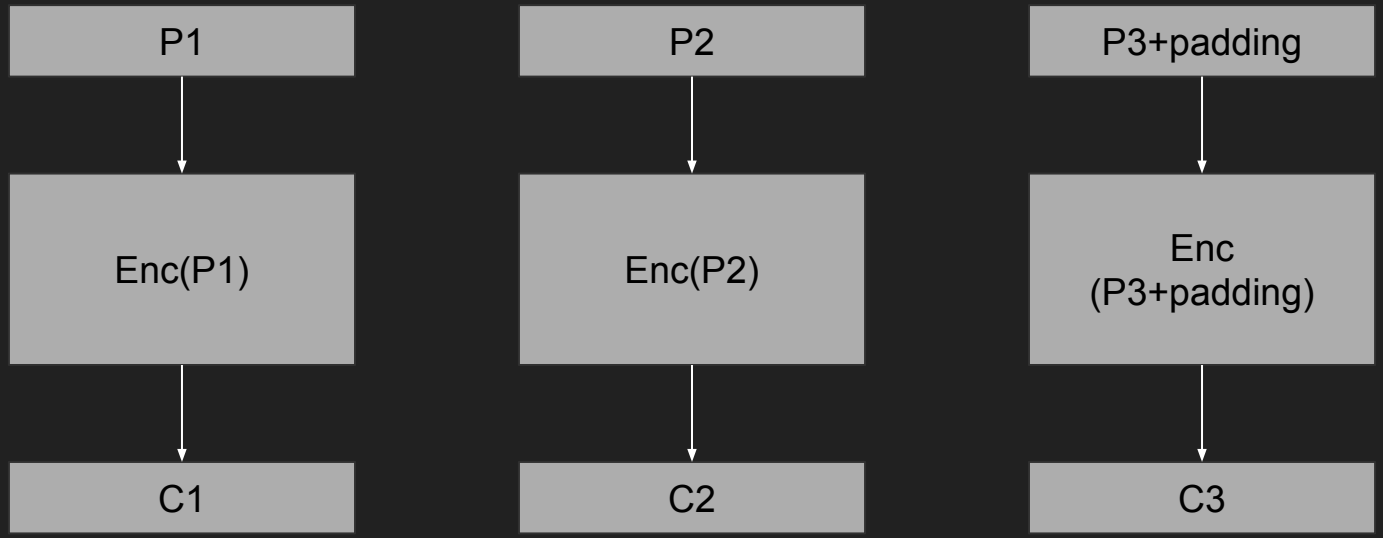
過程都很複雜，解題時通常不需要知道 DES 跟 AES 的詳細算法，只要會用工具

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

def encrypt_aes(plaintext, key):
    cipher = AES.new(key, AES.MODE_ECB)
    ciphertext = cipher.encrypt(pad(plaintext, AES.block_size))
    return ciphertext

def decrypt_aes(ciphertext, key):
    cipher = AES.new(key, AES.MODE_ECB)
    decrypted = unpad(cipher.decrypt(ciphertext), AES.block_size)
    return decrypted
```

接下來的東西比較重要w

分塊模式（Block mode）

把明文分成許多個等長的 Block 的加密方法



常見Padding方法

Zero padding：全部填 0

- ……00000000

Bit padding：加個 1 然後後面都填 0

- ……80000000

PKCS#5, PKCS#7：全部填padding 長度

- ……04040404
- ……..030303

ANSI X9.23：前面填 0，最後一個byte填 padding 長度

- ……00000004
- ……..000003

ISO 10126：前面填nonce，最後一個byte填 padding長度

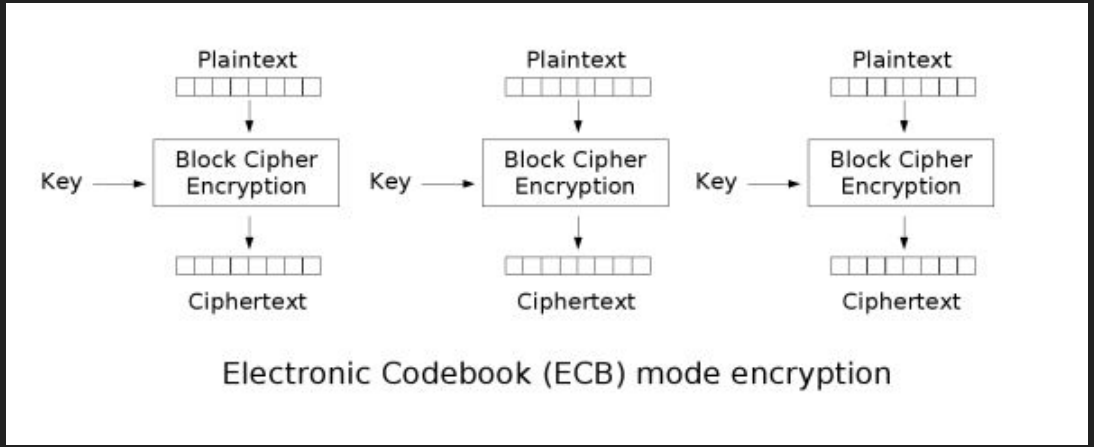
- ……81A62304
- ……..81A603

ECB mode

(Electronic Codebook), 每個 BLOCK 獨立加密

優點:好實做, 區塊間沒有相依性, 可並行處理

缺點:對每個BLOCK, 相同的輸入會產生相同的輸出。



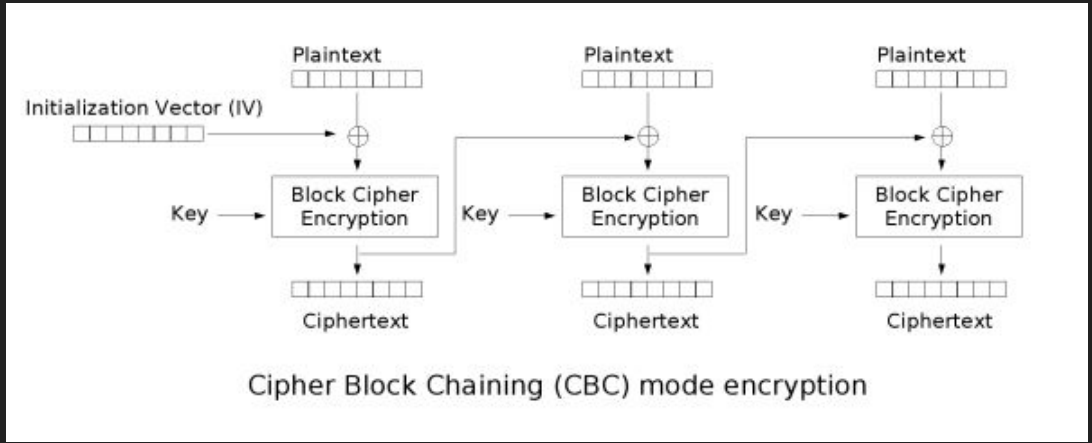
CBC mode

(Cipher-block chaining), 每個 BLOCK 的明文先 XOR 前一個 BLOCK 的密文

優點:有隨機性

缺點:不易並行處理, bit-flipping

多了一個 初始化向量IV



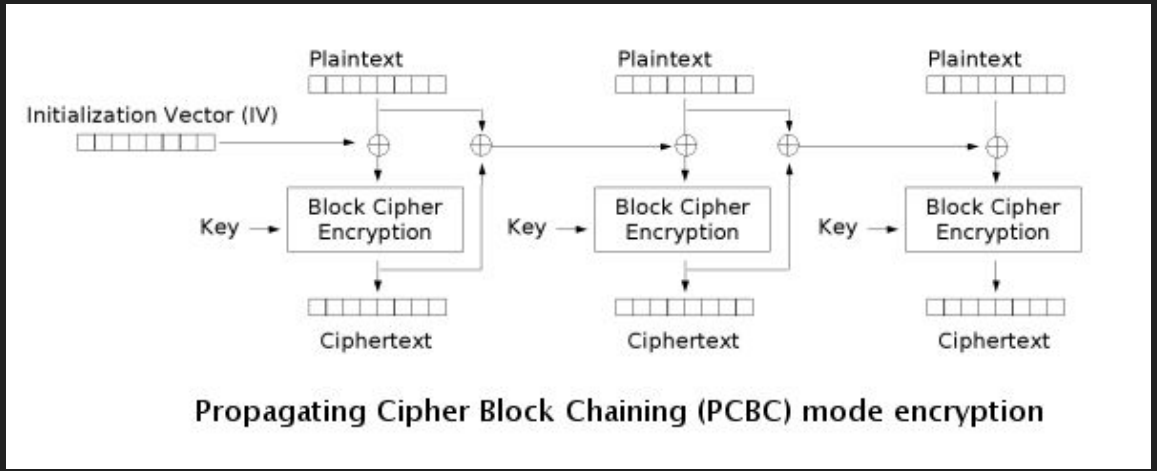
Cipher Block Chaining (CBC) mode encryption

PCBC mode

(Plaintext/Propagate cipher-block chaining), 在下一個BLOCK之前跟明文 XOR

優點:改動一點點密文就會讓明文大部分錯誤

缺點:不易並行處理, 互換相鄰兩塊時不會對其他塊造成影響

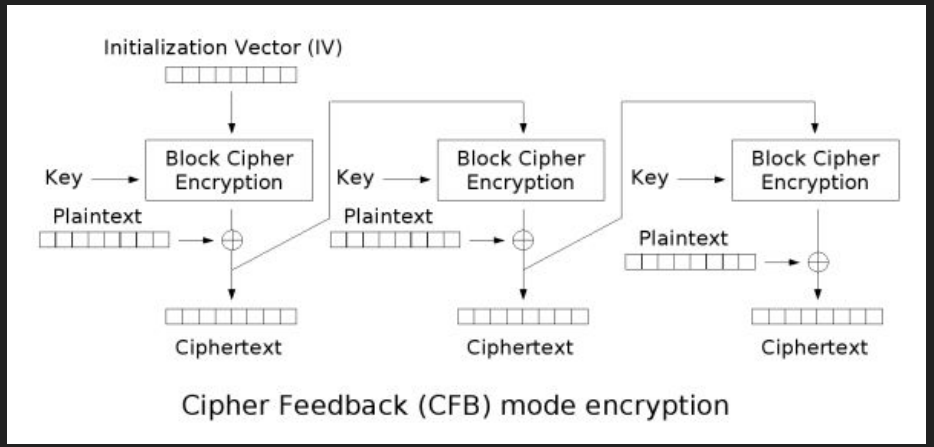


CFB mode

(Cipher feedback), 很像CBC, 然後加密完 IV 再跟明文 XOR

優點:有隨機性, 可以用於流加密

缺點:不易並行處理



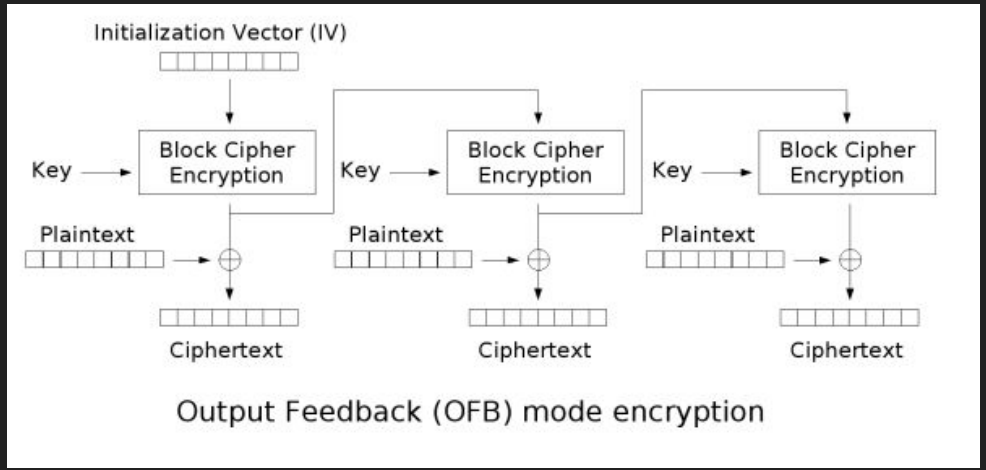
OFB mode

(Output feedback), 明文密文的關係只有 XOR

優點:有隨機性, 可以用於流加密。

可以事先對 IV 進行加密, 最後跟明文或密文 XOR 就好。

缺點:不易並行處理



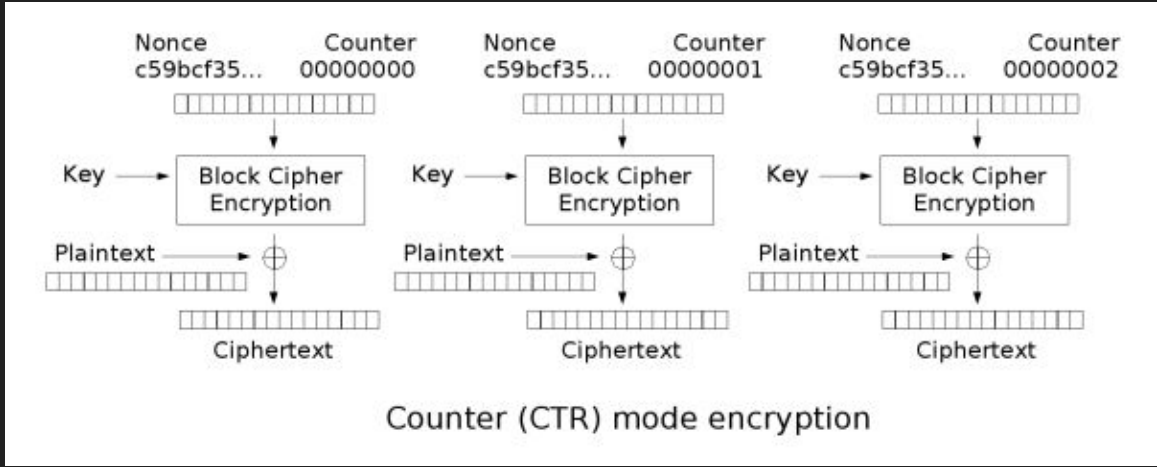
CTR mode

(Counter), 每個 BLOCK 各自獨立, 但比 ECB 多了個計數器

優點:計數器保證不會有 ECB 的問題, 可用於串流加密

缺點:bitflipping

* Nonce 功能跟 IV 差不多



CBC padding oracle

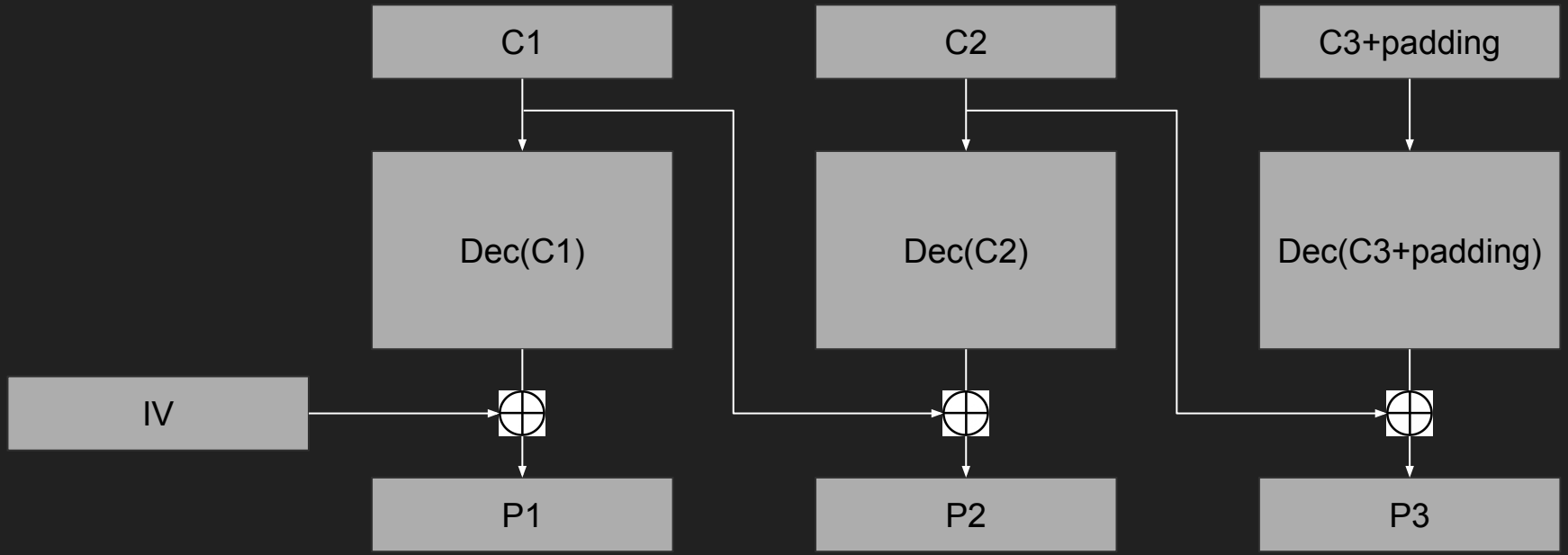
使用條件:

- 分組模式 CBC
- padding 使用 PKCS #7
- 可以不停讓對方解密
- 可以得知解密結果是否符合 PKCS #7

就可以在不知道KEY的條件下, 慢慢翻出明文！

CBC padding oracle

CBC解密長這樣

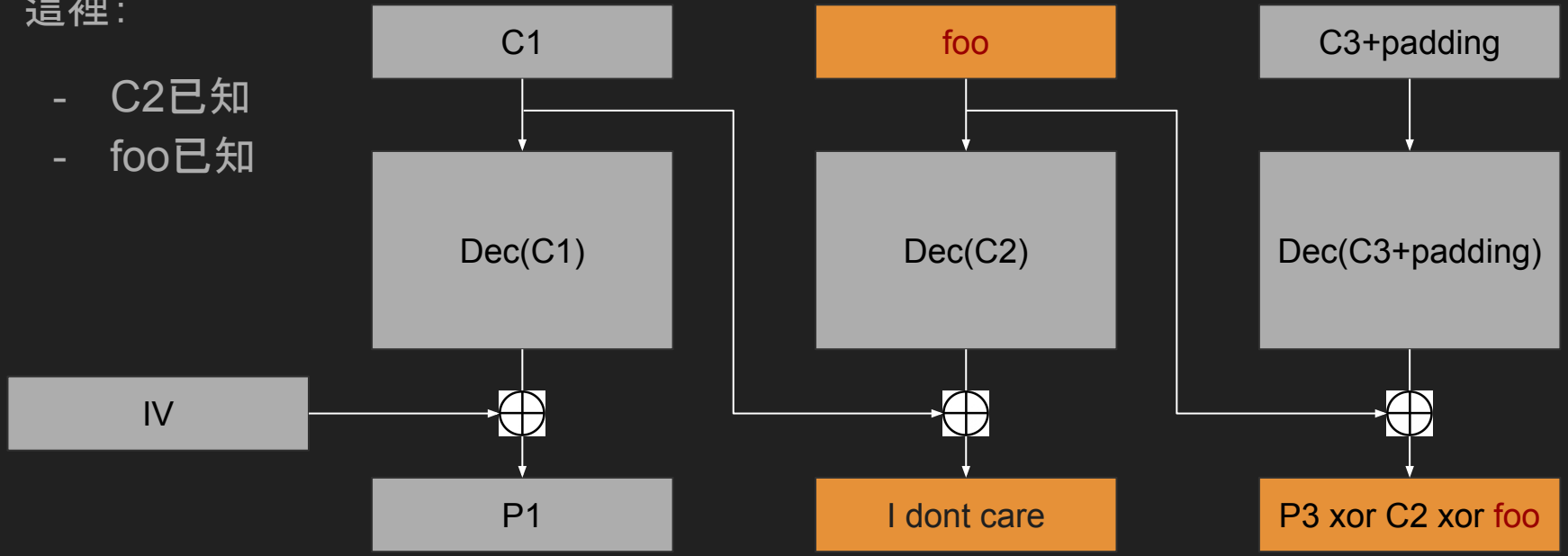


CBC padding oracle

把 C2 改成 foo，會發生

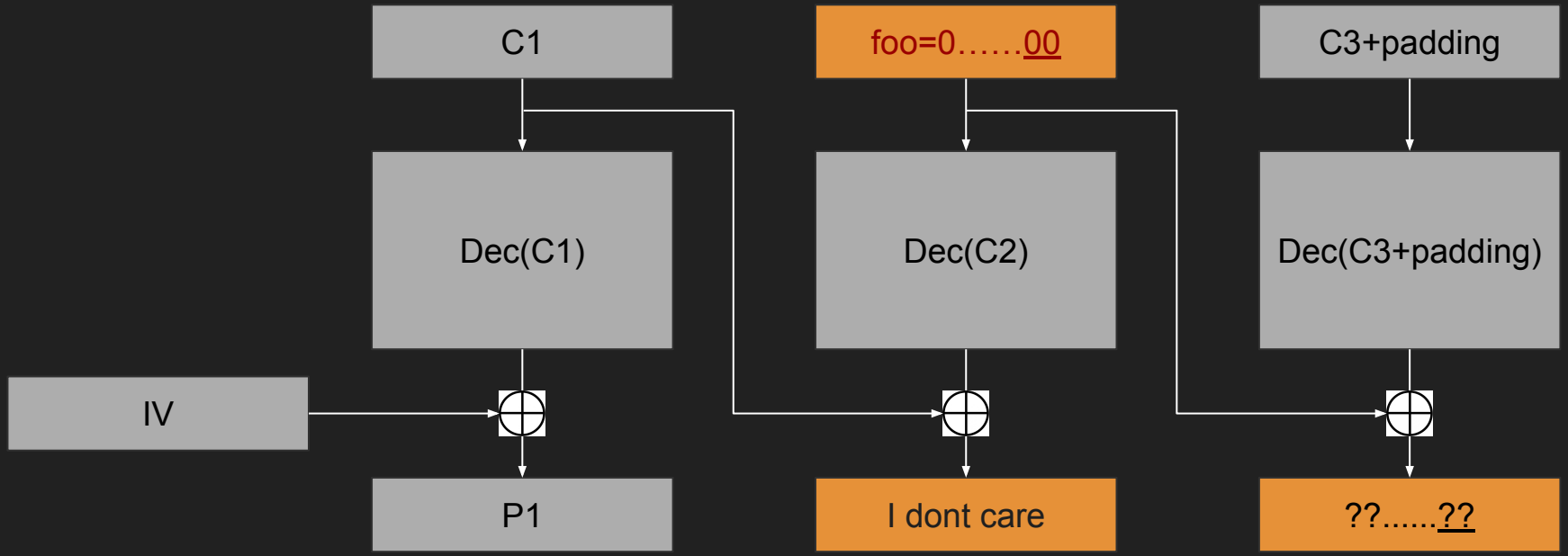
這裡：

- C2已知
- foo已知



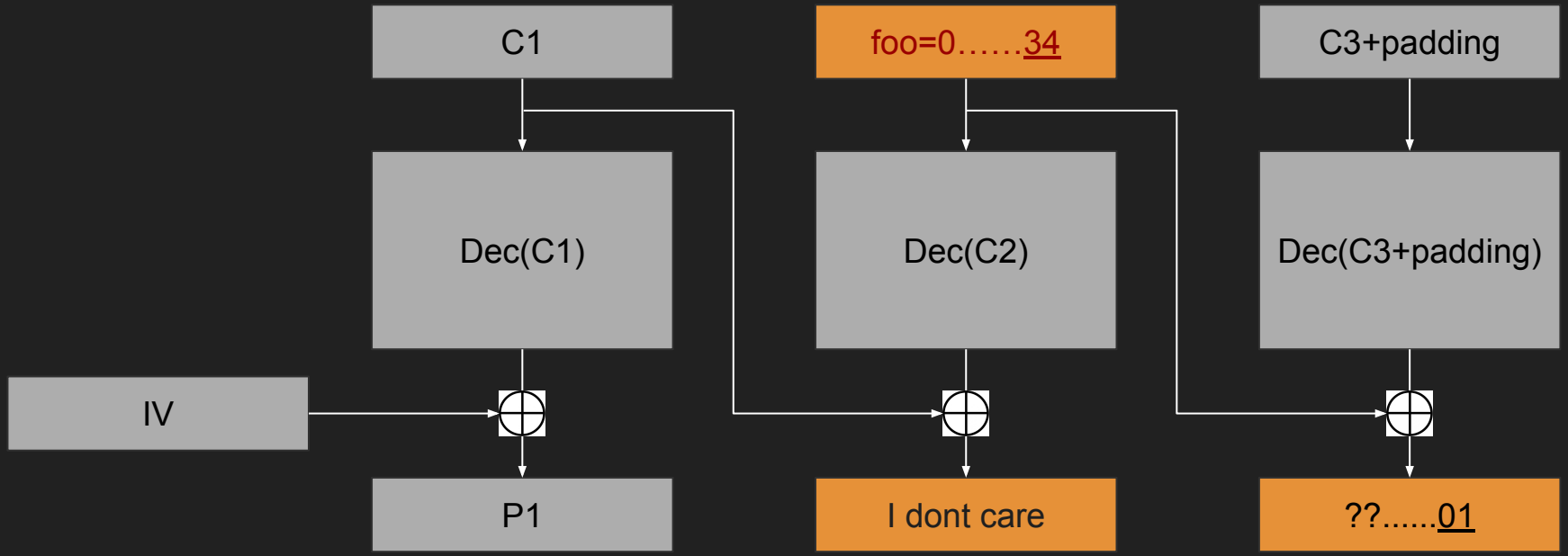
CBC padding oracle

把 foo 全部改成 0



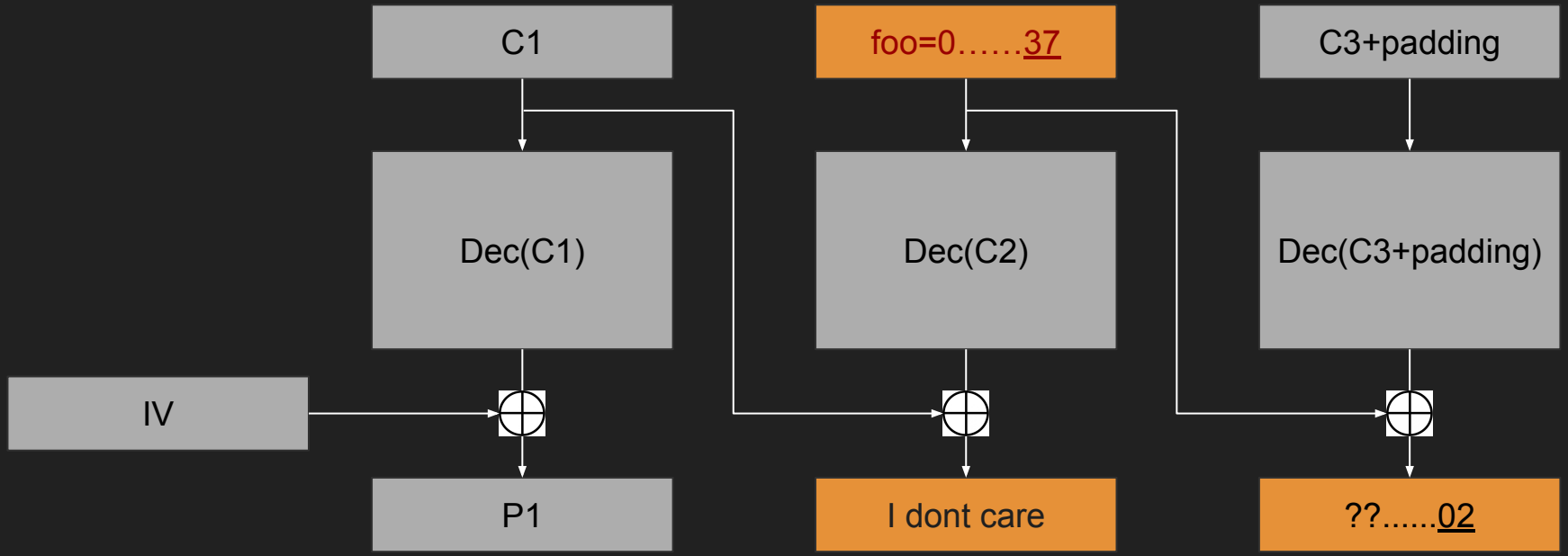
CBC padding oracle

暴力舉 foo 的最後一個 byte，直到 padding 正確，P3 最後一個 byte 可計算



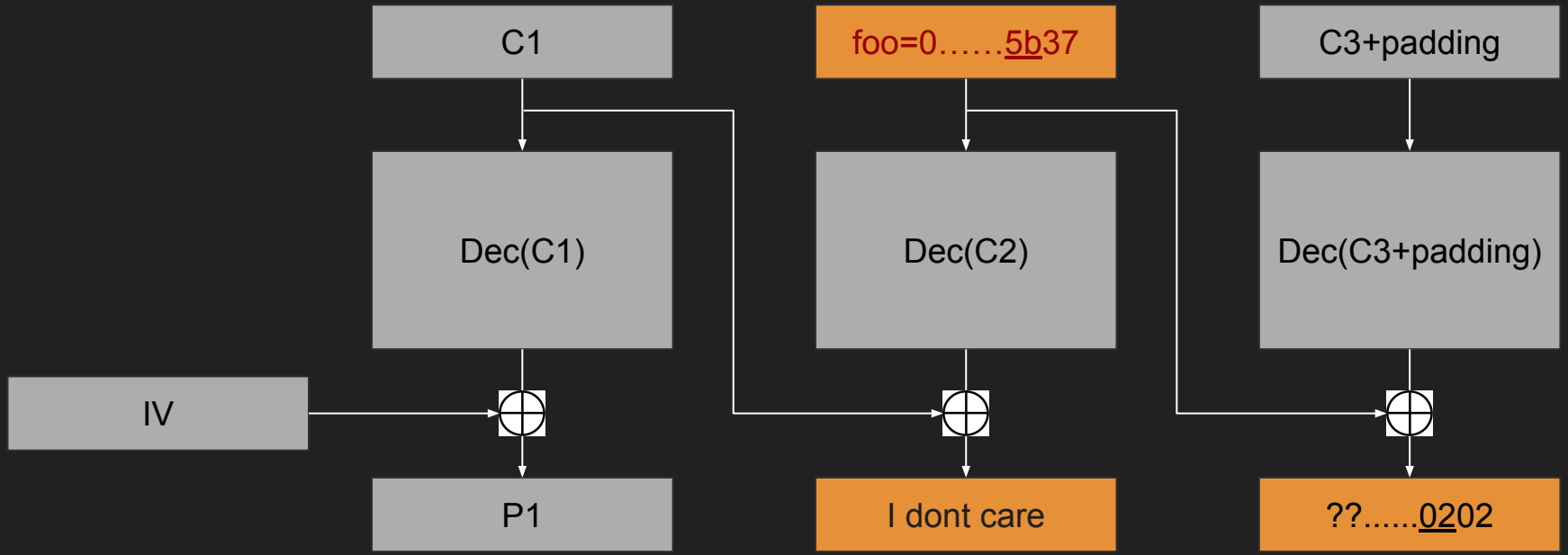
CBC padding oracle

把 34 換成 34 xor 01 xor 02 = 37



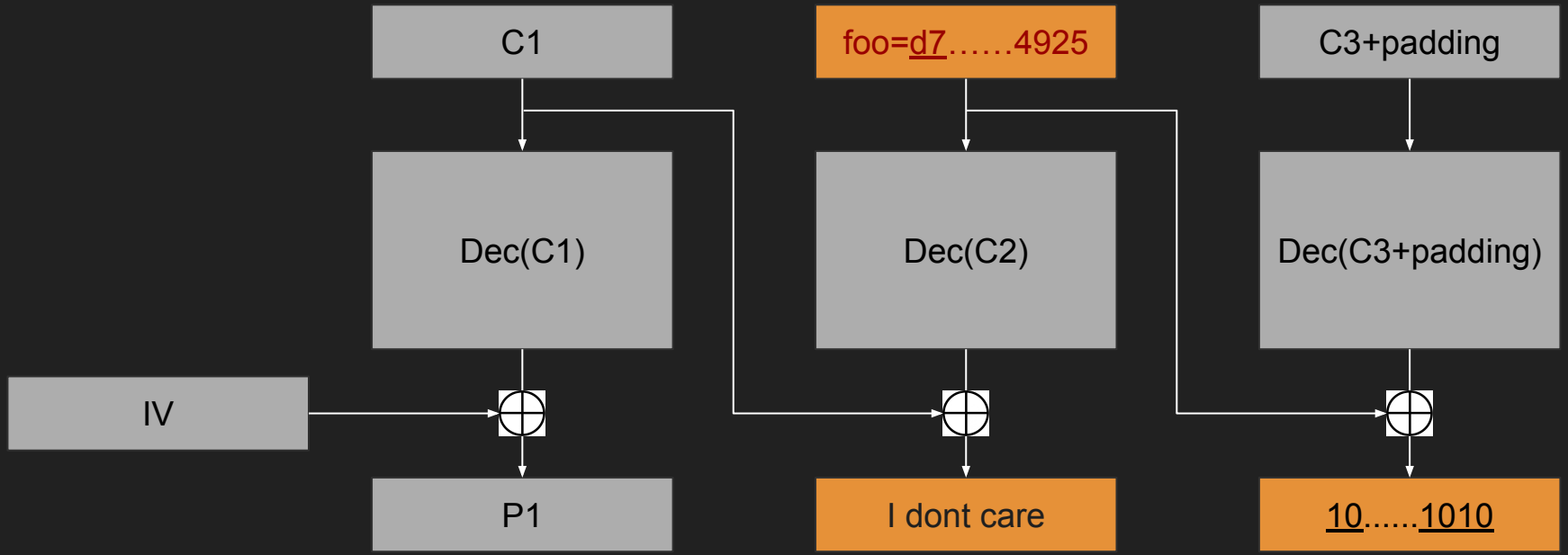
CBC padding oracle

暴力舉 foo 的倒數第 2 個 byte，直到 padding 正確，P3 最後兩個 byte 可計算



CBC padding oracle

翻完之後, P3 = foo xor C2 xor 10……1010



LAB Time.
(ECB CBC oracle、CBC oracle)

雑湊函數（Hash function）

雜湊函數Outline

- 介紹
- SHA（Secure Hash Algorithm ）
- 長度擴充攻擊 LEA
- HMAC

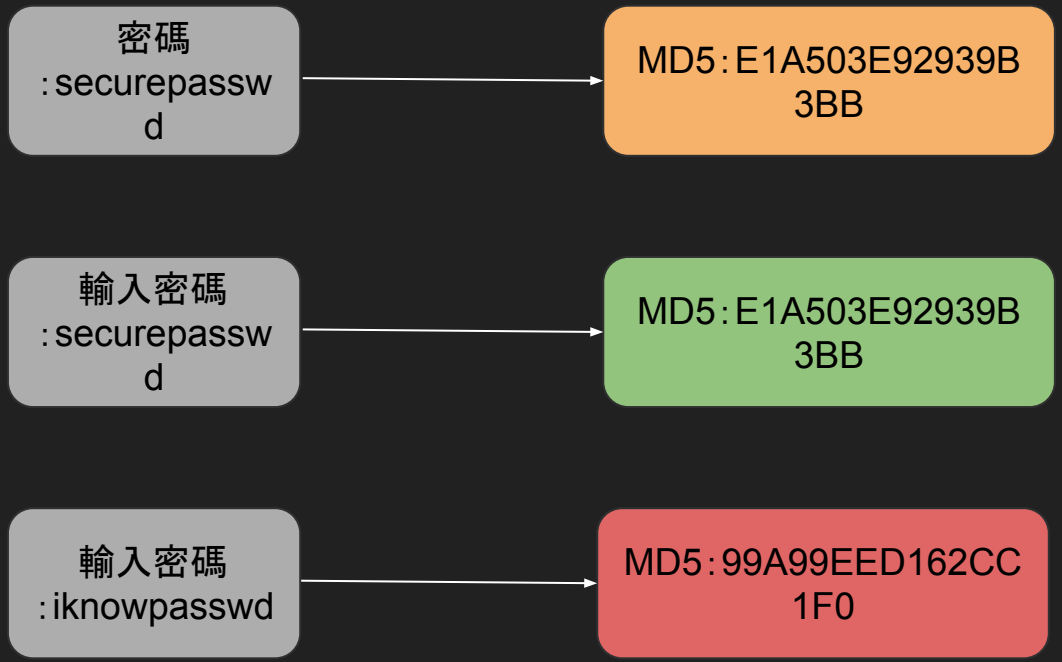
什麼是雜湊函數

它是:

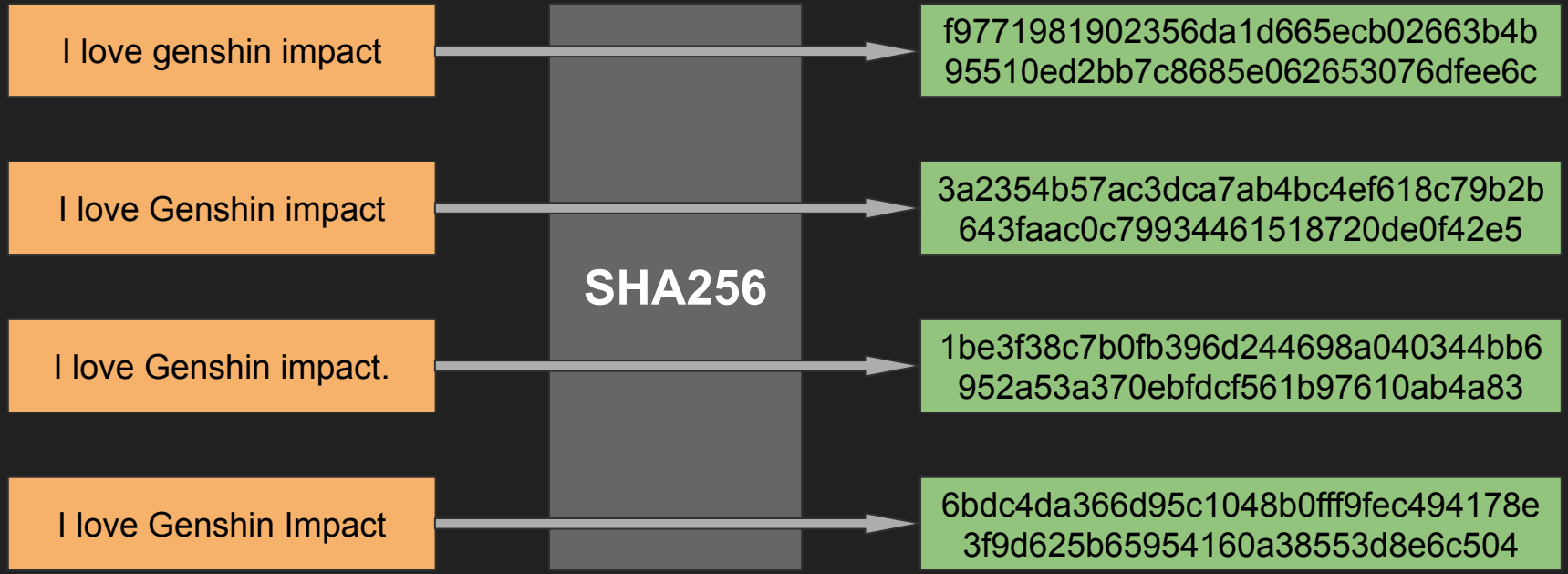
- 多對一
- 不可逆

用途:「比對」

- 文本比對
- 驗證密碼



理想中的雜湊函數


I love genshin impact
I love Genshin impact
I love Genshin impact.
I love Genshin Impact
SHA256
f9771981902356da1d665ecb02663b4b
95510ed2bb7c8685e062653076dfee6c
3a2354b57ac3dca7ab4bc4ef618c79b2b
643faac0c79934461518720de0f42e5
1be3f38c7b0fb396d244698a040344bb6
952a53a370ebfdcf561b97610ab4a83
6bdc4da366d95c1048b0fff9fec494178e
3f9d625b65954160a38553d8e6c504

SHA（Secure Hash Algorithm）

- 包含
  - MD5, SHA0, SHA1
  - SHA2 (SHA-224, SHA-256, SHA-384, SHA-512)
  - SHA3 (SHA-224, SHA-256, SHA-384, SHA-512)
- SHA0, SHA1, … SHAx 我解讀成第 x 版的演算法
- SHA-256, SHA-512… SHA-y 就是HASH的結果總共有 y 個 bits
- 演算法很複雜，有興趣的自己查

碰撞攻擊(Collision)

- 不同的輸入但有一樣的 Hash值
- 例子:找到一個跟securepasswd有一樣MD5的結果
- 理想的雜湊函數 => 不會碰撞
- 真實的雜湊函數 => 可能碰撞

原像攻擊(Preimage)

- 用已知的Hash值推斷可能的輸入
- 例子:找到一個輸入使得 MD5的結果是E1A503E92939B3BB

MD5

2009被破解，可在普通電腦上碰撞攻擊

指定前綴
：https://github.com/zhijieshi/md5collgen

指定 iv：

https://github.com/s1fr0/md5-tunneling

SHA1

2017被google工程師解出Collision

SHA1 Collision：https://shattered.io/

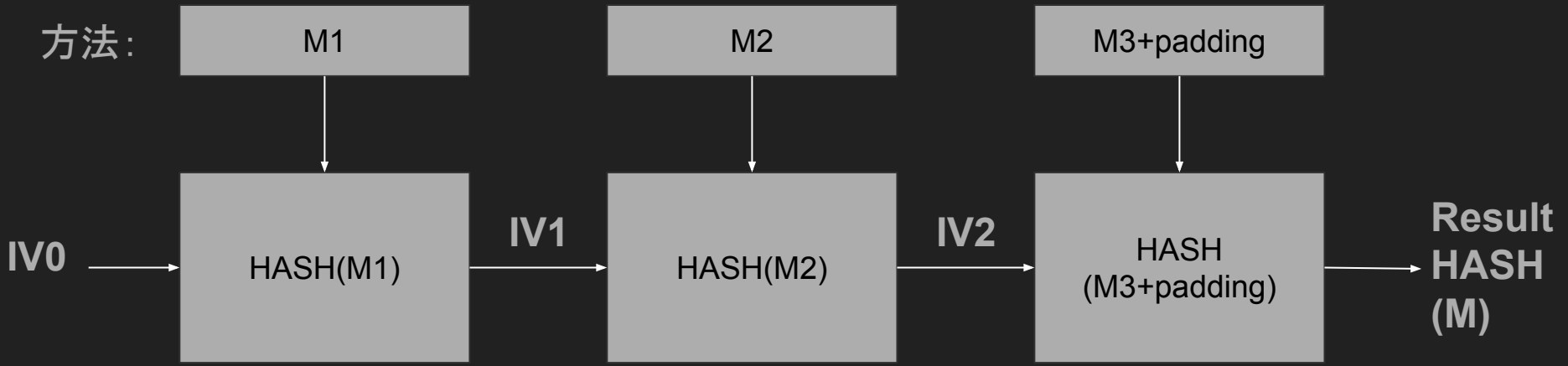
＊SHA2 跟 SHA3 目前還沒被破解

MD結構（Merkle–Damgård construction）

當明文太長的時候，怎麼辦？

將明文分成等長的塊。最後不夠長的補padding。

padding 使用 10……00 的那種方法。



長度擴展攻擊 Length Extension Attack (LEA)

使用情境:

- 已知明文的雜湊值 H(M)
- 已知明文長度 |M|

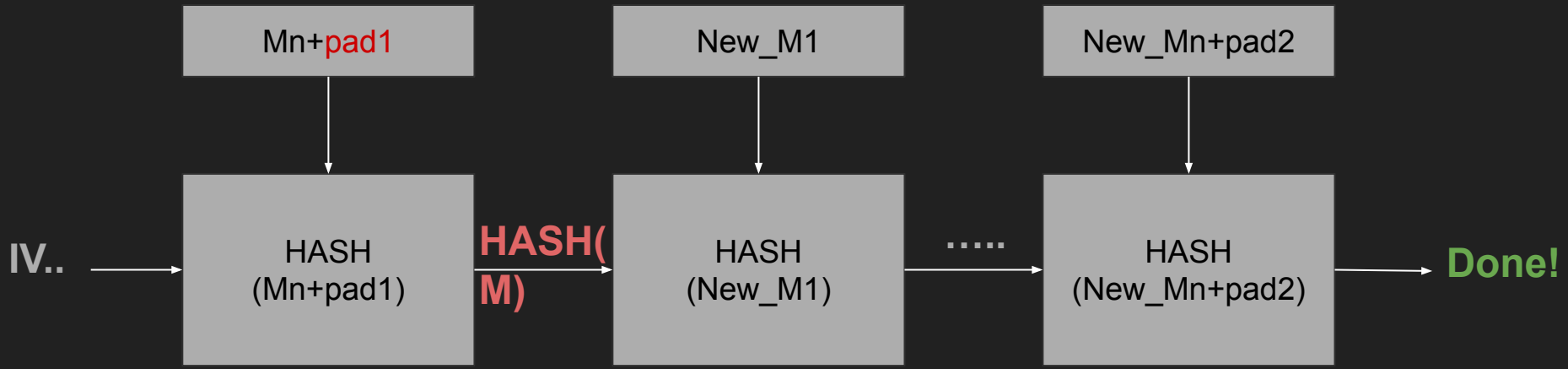
在不知道明文的情況下,

可以擴充出:

- H(M || padding || New_M)
- H(“admin=0” || idc || “ OR admin=1”)

長度擴展攻擊 Length Extension Attack (LEA)

- 需要知道 padding1 填啥所以需要知道明文 M 長度
- 填完 padding 之後填新的訊息 New_M



HMAC (hash-based message authentication code)

金鑰雜湊訊息鑑別碼，也是現代常用的。

使用已知的 HASH 算法像是 SHA-256

- 變成 HMAC-SHA-256

可抵禦各種手段的攻擊，包含 LEA

HMAC (hash-based message authentication code)

根據RFC 2104, HMAC的數學公式為:

$$HMAC(K, m) = H\Big((K' \oplus opad) \,||\, H\big((K' \oplus ipad) \,||\, m\big)\Big)$$

其中K是密鑰, m是明文訊息, K'是

- H(K), 如果 |K| > block size
- zero_pad(K), 如果 |K| <= block size

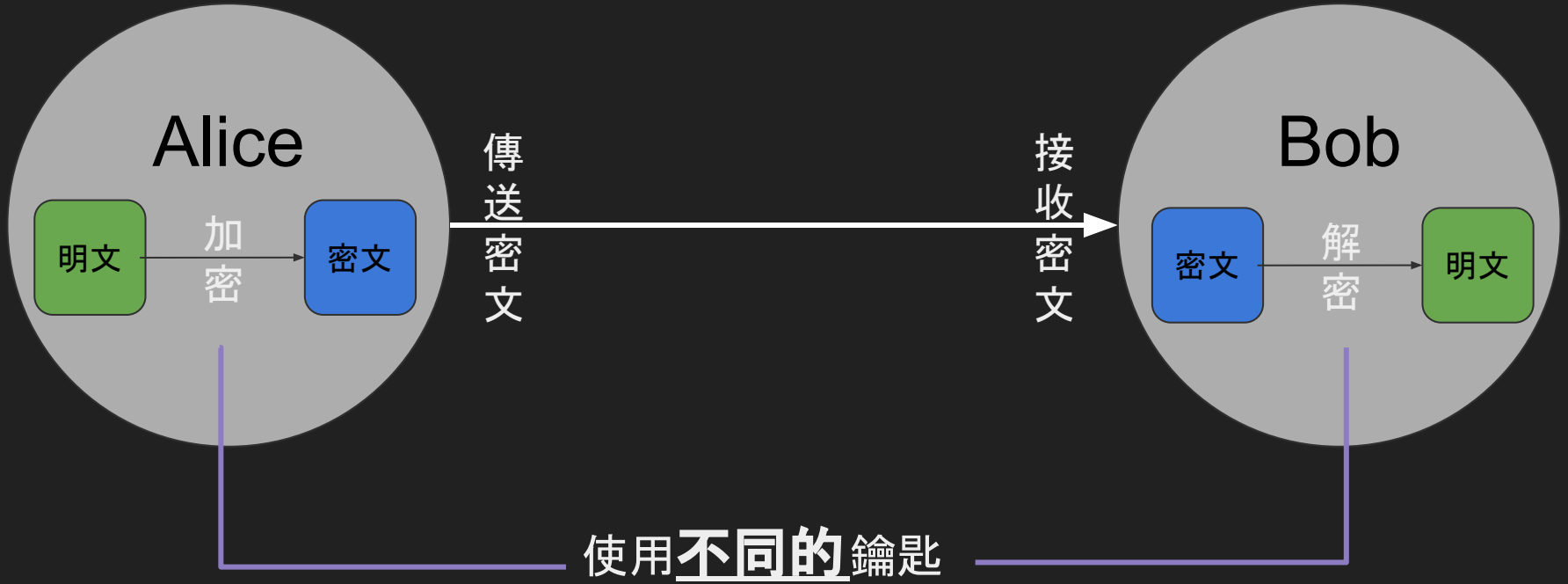
opad 是 0x5c5c5c…5c5c

ipad 是 0x363636…3636

LAB Time.
(Echo lea, Real lea)

非對稱式密碼（Publickey cipher）

非對稱式密碼



非對稱式密碼

「億」點數學

- 最大公因數
- 歐幾里得算法
- 擴展歐幾里得算法
- 費瑪小定理
- 歐拉函數

RSA & 各種條件攻擊

離散對數相關

- 離散對數難題
- Deffie-Hellman 金鑰交換
- Elgamel

離散對數相關

離散對數難題

Deffie-Hellman 金鑰交換

Elgamel

最大公因數（Greatest Common Divisor）

對於整數 a,b 的最大公因數，寫成 gcd(a,b)

gcd 等等會一直看到:）

```
import gmpy2
print(gmpy2.gcd(20,50))
```

歐幾里得算法（Euclidean algorithm）

其實就是輾轉鄉除法

gcd(a,b) = gcd(b, a mod b)

可以快速算出 a,b 的最大公因數

擴展歐幾里得算法(Euclidean algorithm)

貝祖定理:

- 對於整數 a,b, 存在整數 x,y 使得 ax + by = gcd(a,b)

找出 x,y:

- gcd(a,b), x, y = gcd(b, a % b), y, x - [a//b] × y

模反元素:

- a 的模反元素 d, 有 ad ≡ 1 (mod m)
- ad = 1 + km (k 是整數) => ad - km = 1
- 找出 b, -k, 使得 ad+m(-k) = 1, 就找到了

費瑪小定理（Fermat's little theorem）

對於質數 p 跟整數 a，如果 gcd(a,p)=1, 則 $a^{p-1} \equiv 1 \pmod{p}$ 恆成立

證明：

- 集合 {1, 2, 3,...,p-1} ≡ {a, 2a, 3a, …, (p-1)a} (mod p)
- $1 \times 2 \times 3 \times ... \times (p-1) \equiv a \times 2a \times 3a \times ... \times (p-1)a \equiv 1 \times 2 \times 3 \times ... \times (p-1) \times a^{p-1} \pmod{p}$
- => $a^{p-1} \equiv 1 \pmod{p}$

延伸：

- $a^{-1} \equiv a^{p-2} \pmod{p}$
- $a^x \equiv a^y \pmod{p} \Rightarrow x \equiv y \pmod{p-1}$

歐拉定理（Euler’s theorem）

對於整數 n 跟整數 a，如果 gcd(a,n)=1，則 $a^{\varphi(n)} \equiv 1 \pmod{n}$ 恆成立。

一般化的費瑪小定理。

歐拉函數 φ(n)：

- 質因數分解 $n = p1^{k1} \times p2^{k2} \times \dots \times pr^{kr}$
- $\varphi(n) = p1^{k1-1} \times p2^{k2-1} \times \dots \times pr^{kr-1} \times (p1-1) \times (p2-1) \times \dots \times (pr-1)$

常用性質：

- 質數 p => φ(p) = p-1
- 兩整數 m,n 且 gcd(m,n)=1 => φ(mn) = φ(m) × φ(n)

歐拉定理（Euler’s theorem）

對於整數 n 跟整數 a，如果 gcd(a,n)=1，則 $a^{\varphi(n)} \equiv 1 \pmod{n}$ 恆成立。

證明有點複雜，但跟費瑪小定理差不多的想法。

延伸：

- $a^x \equiv a^y \pmod{n} \Rightarrow x \equiv y \pmod{\varphi(n)}$

RSA

發明者:

- Ron Rivest
- Adi Shamir
- Leonard Adleman

密鑰長度:2048~4096 bits

數學難題:質因數分解

RSA

金鑰生成:

- 找出兩個超級大質數 p,q, 計算 $N=pq$
- 計算 $\varphi(N) = (p-1)(q-1)$
- 選一個 $e < \varphi(N)$, 且滿足 $\gcd(e, \varphi(N)) = 1$
- 計算 $d \equiv e^{-1} \pmod{\varphi(n)}$, 使得 $ed \equiv 1 \pmod{\varphi(n)}$
- 公鑰為 (N,e), 私鑰為 (N,d), p跟q可以銷毀

RSA

加密：

- $c \equiv m^e \pmod N$

解密：

- $m \equiv c^d \pmod N$

它是正確的：

- $m \equiv c^d \equiv (m^e)^d \equiv m^{ed} \pmod N$
- $ed \equiv 1 \pmod{\varphi(N)} \Rightarrow m^{ed} \equiv m \pmod N$

RSA 分解n相關

分解n相關:

- n 夠小 -> 分解 http://factordb.com/
- 共用 p（$\gcd(n_1,n_2) > 1$）-> $p=\gcd(n_1,n_2)$
- |p-q| 很大 -> 暴力解較小的p或q
- |p-q| 很小 -> 費瑪分解
- p-1 夠smooth -> pollard p-1
- p+1 夠smooth -> william p+1

RSA e=3

$c \equiv m^3 \pmod{n}$

$\Rightarrow m^3 = c + kn$

如果 m 很小，或許可以枚舉 k 直到 m 被算出來

RSA d夠小（$d < (1/3) \times N^3$）

Wiener’s attack

https://github.com/orisano/owiener

RSA 共模攻擊

擁有

- $c_1 \equiv m^{e1} \pmod n$
- $c_2 \equiv m^{e2} \pmod n$
- 已知c_1, c_2, e_1, e_2, n
- $\gcd(e_1, e_2) = 1$

可以找出m

- $c_1^{\,x} \equiv m^{e1x} \pmod n$
- $c_2^{\,y} \equiv m^{e2y} \pmod n$

先用擴展歐幾里得找出x,y讓

$e_1x + e_2y = 1$

然後

$m \equiv m^{e1x + e2y}$

$\equiv c_1^{\,x} \times c_2^{\,y} \pmod n$

離散對數

正常的對數:

$a^b = x \Rightarrow b = \log_a(x)$

離散對數難題:

$a^b \equiv x \pmod n \Rightarrow b \equiv \log_a x \pmod n$

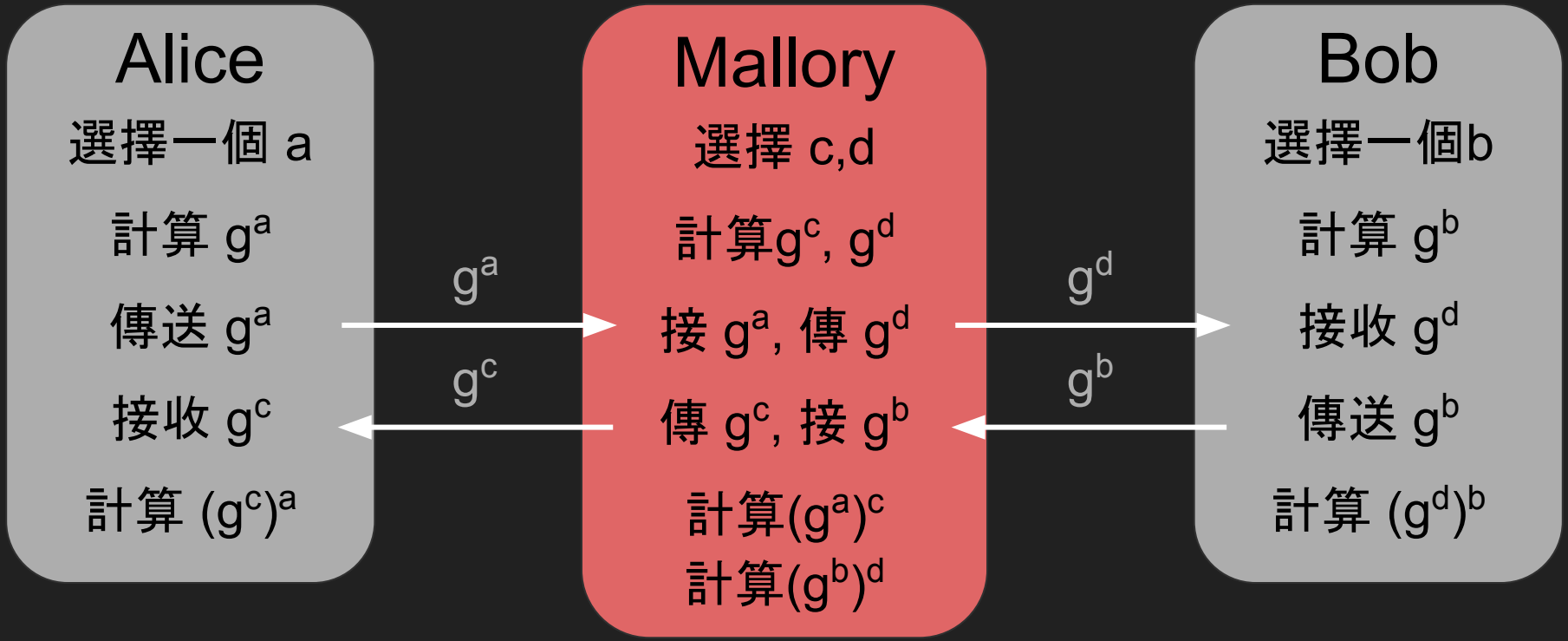
已知 a, x, 目前無演算法在多項式時間找出 b

Deffie-Hellman 金鑰交換

在攔截者無法得知訊息的情況下，讓傳送訊息的雙方擁有共享訊息

Deffie-Hellman 金鑰交換

＊容易受到中間人攻擊



離散對數相關的其他方法

Elgamel

- Deffie-Hellman 改編而來
- 也用於數位簽章

ECC

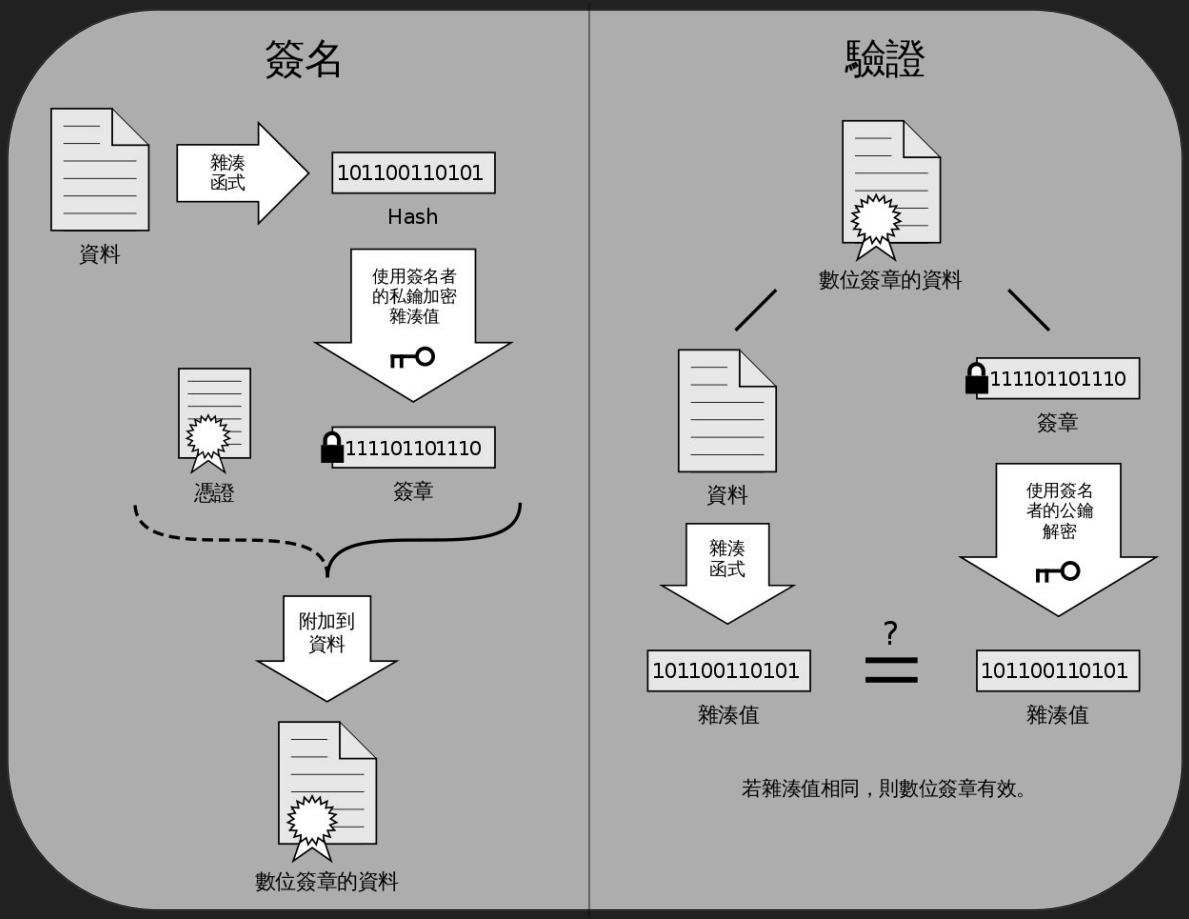
- 橢圓曲線加密
- $y^2 = x^3 + ax + b \pmod p$

數位簽章（Digital Signature）

數位簽章

確保訊息沒被修改的方法。

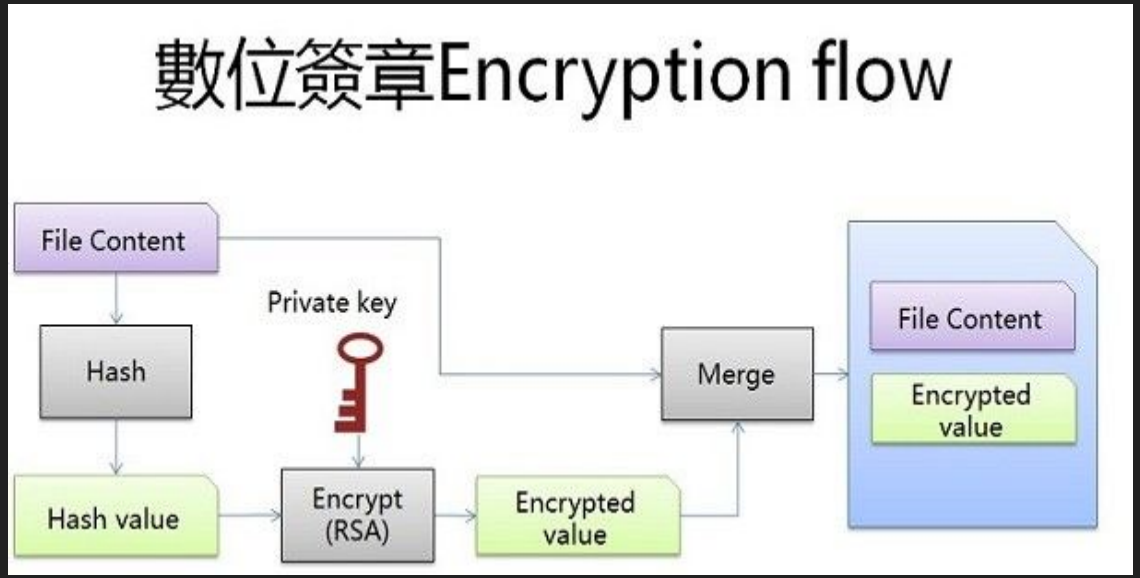
- 不可偽造
- 不可複製
- 不可竄改
- 簽名者不可否認
- 任何人都能驗證



數位簽章

用非對稱密碼生成

- 公鑰(大家驗證用)
- 私鑰(自己簽名用)



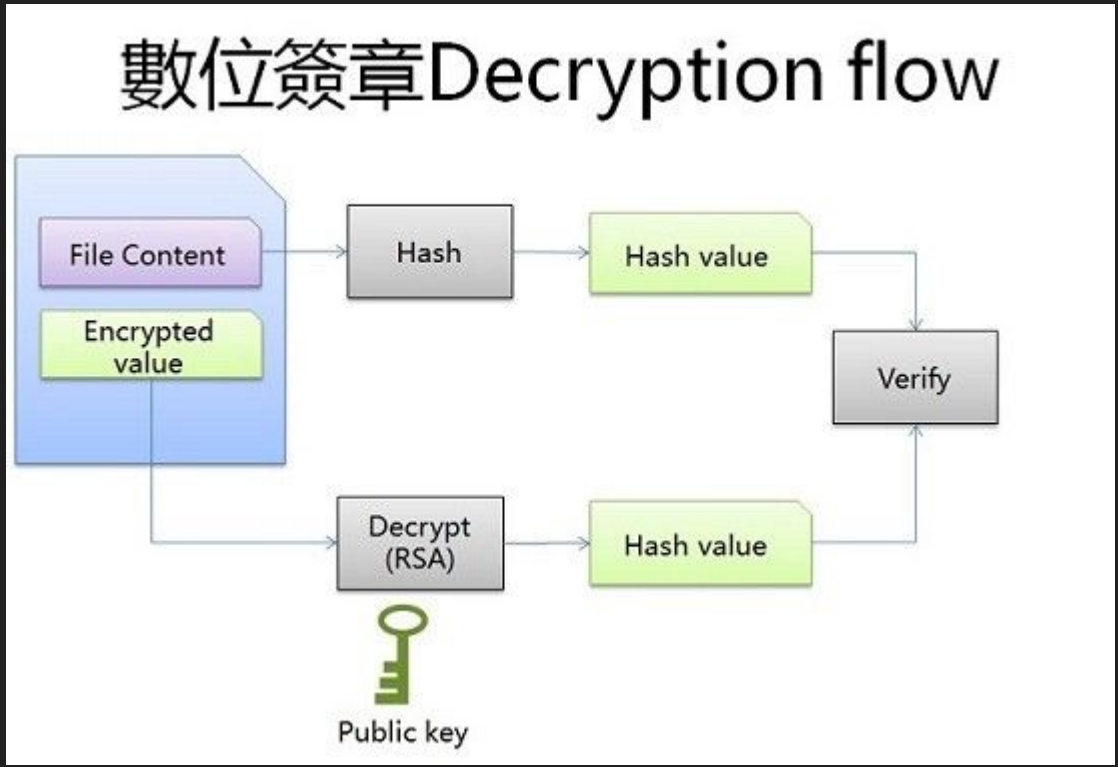
圖片來源:

https://ithelp.ithome.com.tw/articles/10188465

數位簽章

用非對稱密碼生成

- 公鑰(大家驗證用)
- 私鑰(自己簽名用)



圖片來源:

https://ithelp.ithome.com.tw/articles/10188465

就先醬啦

感謝聆聽～

LAB Time.
(RSA easy、RSA simplest)
(RSA reusen、midhle)