

# Cryptography 1

lce1187

# Speaker

黃俊嘉 (Ice1187)

- ▶ TNFSH 百八級
- ▶ Master's student at NTU CSIE NSLab
- ▶ Member of Balsn CTF Team
- ▶ Member of UNDEFINED
- ▶ Intern Researcher at CyCraft
- ▶ Speaker of CyberSec, SECCON



✉ [hcc001202@gmail.com](mailto:hcc001202@gmail.com)

GitHub <https://github.com/Ice1187>

# 大綱

- 古典密碼學
- 現代密碼學
- 現代密碼學：串流加密
- 雜湊
- 編碼、加密、雜湊的差異



# 聲明

本課程目的在提升學員對資安產業之認識及資安實務能力。本課程所授與的知識和技巧僅做為資安實務教育訓練目的。

所有課程學習內容應於雙方知情、同意且合法的情況下進行實作和練習，並且不得從事非法攻擊或違法行為，以免受到法律制裁。請勿利用所習得之技術從事非法或惡意的攻擊及入侵行為！

# Python 複習

## 算術運算子

```
a + b # 相加  
a - b # 相減  
a * b # 相乘  
a / b # 相除  
a // b # 相除取整數  
a ** b # a的b次方  
a % b # a除b的餘數
```

## 邏輯運算子

```
a > 1 or b <= 2 # OR  
a > 1 and b <= 2 # AND  
not (a > 1) # NOT
```

# Python 複習

For 迴圈

```
for i in range(1, 5):
    print(i)
# output: 1, 2, 3, 4
```

條件判斷

```
if a > 1 and b == 3:
    print(a)
elif a < 1 or b == 5:
    print(b)
else:
    print(c)
```

# Python 複習

- str
  - 對字串的抽象表達，底層為 UTF-8
  - 不能直接對 str 的元素做運算
- bytes
  - 儲存 raw data 的 bytes
  - 操作概念與 C 中的 char 相同
  - 可以直接對 bytes 的元素做運算
- bytearray
  - mutable：可以對內容做修改的 bytes

變數型別

```
a = 1                      # int
b = [1, 2, 3]                # list
c = 'hello'                  # str
d = b'hello'                 # bytes
e = bytes([0x65])            # bytes
f = bytearray()               # bytes (mutable)
```

成員運算子

```
0      in [1, 2, 3]    # False
'a'    in [1, 'a']     # True
'he'   in 'hello'      # True
'heo'  in 'hello'      # False
```

# Python 複習

- str.index(c)
  - 回傳 c 在 str 中的第幾個位置
- list.append(e)
  - 將 e 添加為 list 的最後一個元素
- int.to\_bytes(n, endian)
  - 將 int 以 endian 轉成長度為 n 的 bytes
- int.from\_bytes(bytes, endian)
  - 將 bytes 以 endian 轉成 int

## 常用的型態內建 Methods

```
str.index(c)
list.append(e)
int.to_bytes(n, endian)
int.from_bytes(bytes, endian)
```

## 範例

```
'abcde'.index('d')          # 3
[1, 2].append(3)            # [1, 2, 3]
(16).to_bytes(1, 'big')      # 'b\x10'
int.from_bytes(b'\x10', 'big') # 16
```

# Python 複習

- `chr(int)`
  - 將 int 轉成 ASCII 對應的 str 字元
- `ord(str)`
  - 將 str 字元轉成 ASCII 對應的 int
- `len(str), len(list)`
  - 取得 str、list 的長度
- `print()`
  - 將參數印出到螢幕

常用內建函數

```
chr(65)      # 'A'  
ord('A')     # 65  
len('abc')   # 3  
len([1, 2, 3]) # 3  
print('hello') # hello
```

# Python 複習

- string

- ascii\_lowercase : 小寫字母
- ascii\_letters : 小、大寫字母

```
from string import ascii_lowercase, ascii_letters, digits
ascii_lowercase # 'abc...z'
ascii_letters   # 'abc...zABC...Z'
digits         # '0123456789'
```

- digits : 數字

- random

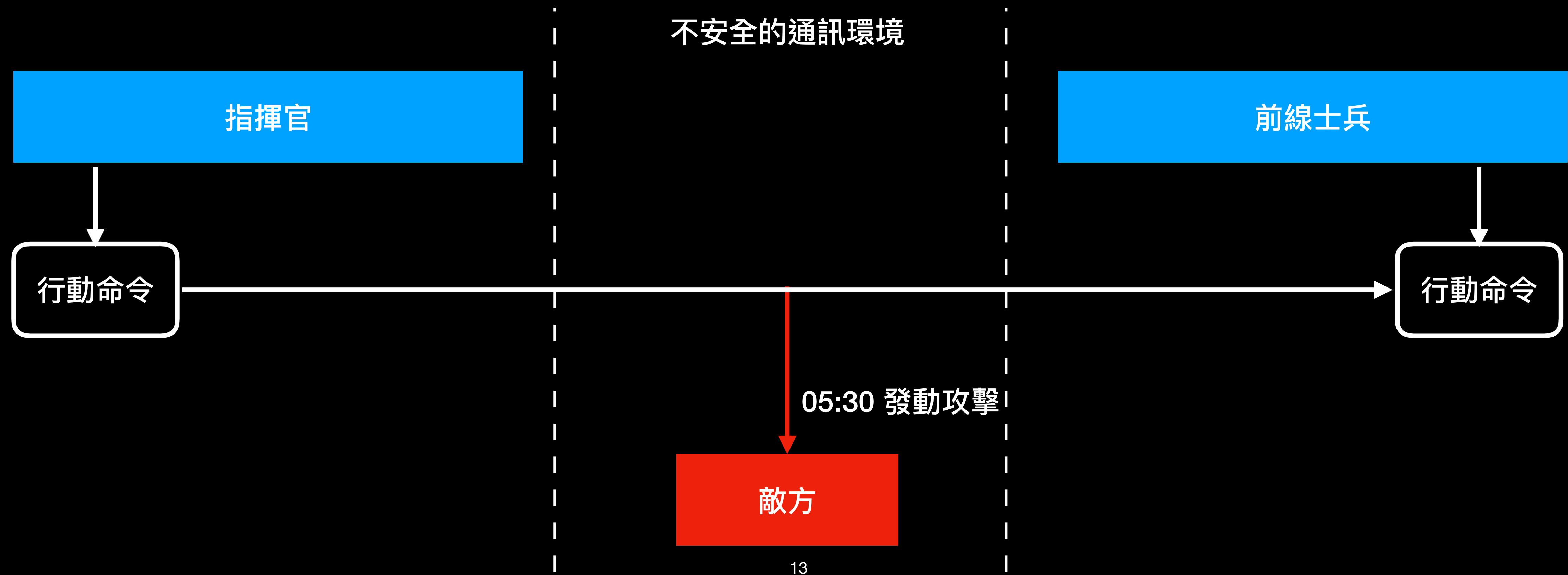
- choices(list, k=1) : 從 list 中選 k 個
- randint(a, b) : 回傳 n ,  $a \leq n \leq b$

```
import random
random.choices(list, k=1)
random.randint(a, b)
```

# 密碼學簡介

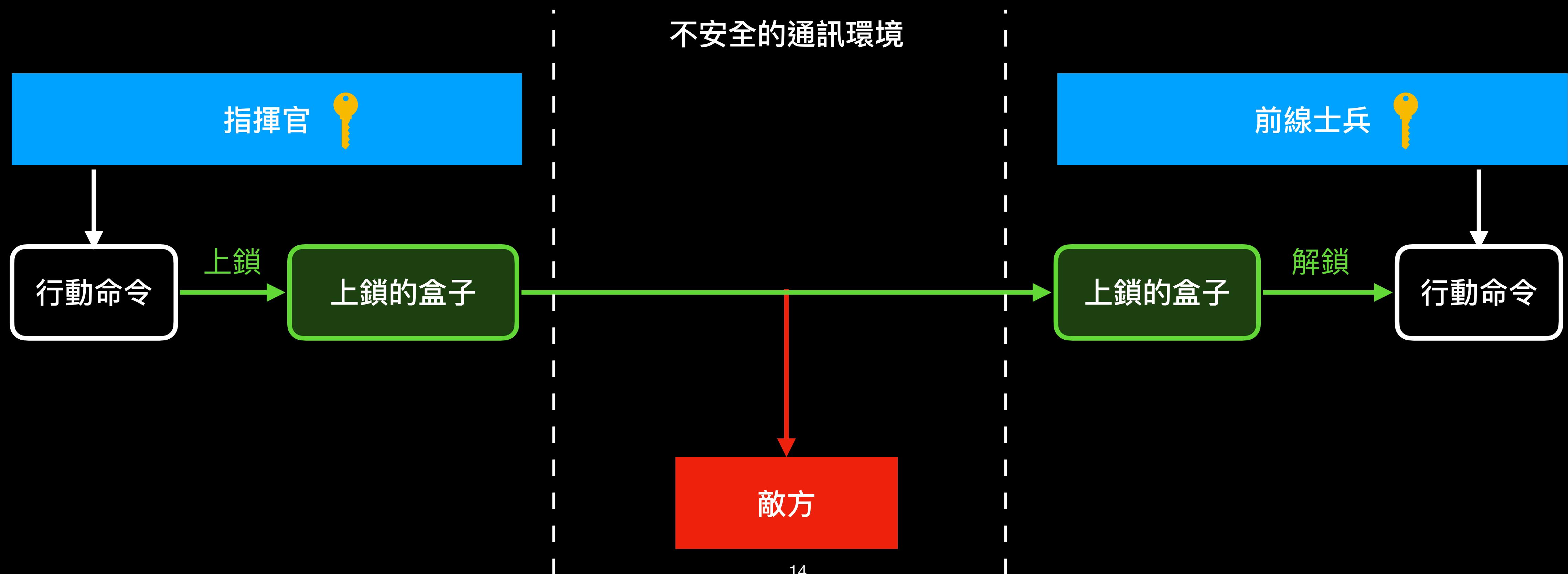
# 密碼學簡介

- 在不安全的環境中建立安全通訊管道的技術，如：戰爭時的軍事通訊



# 密碼學簡介

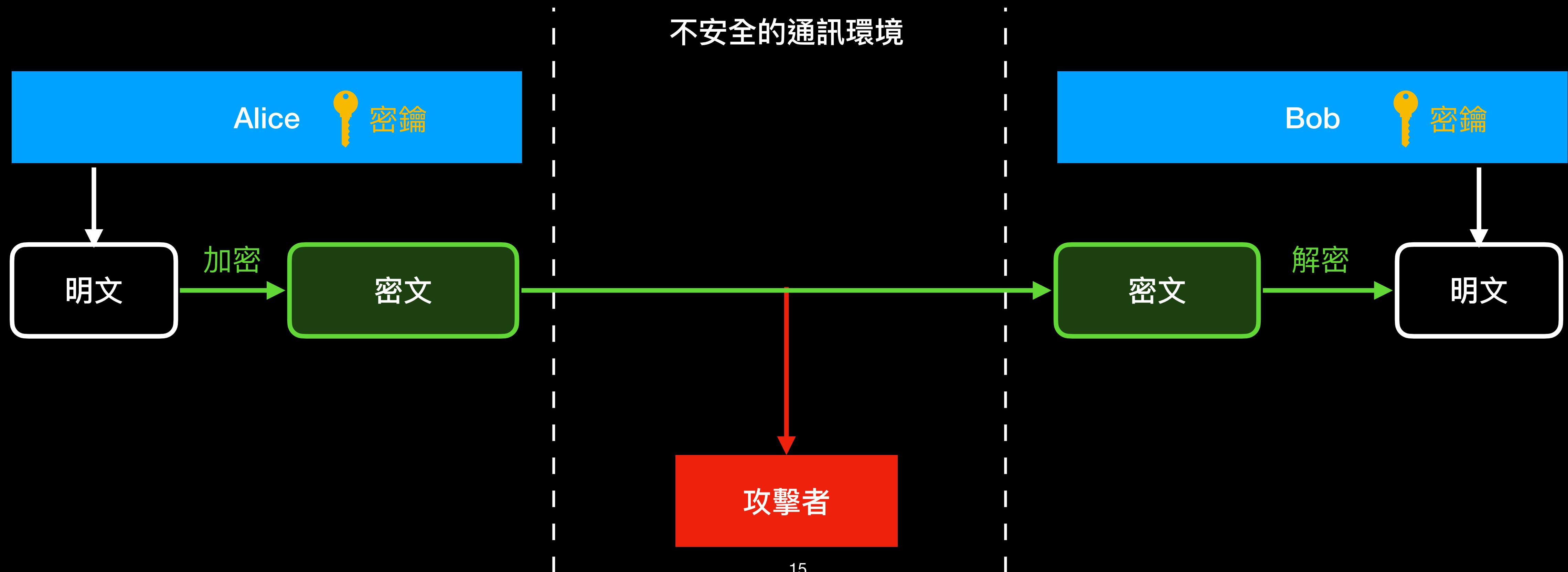
- 在不安全的環境中建立安全通訊管道的技術，如：戰爭時的軍事通訊



# 密碼學簡介

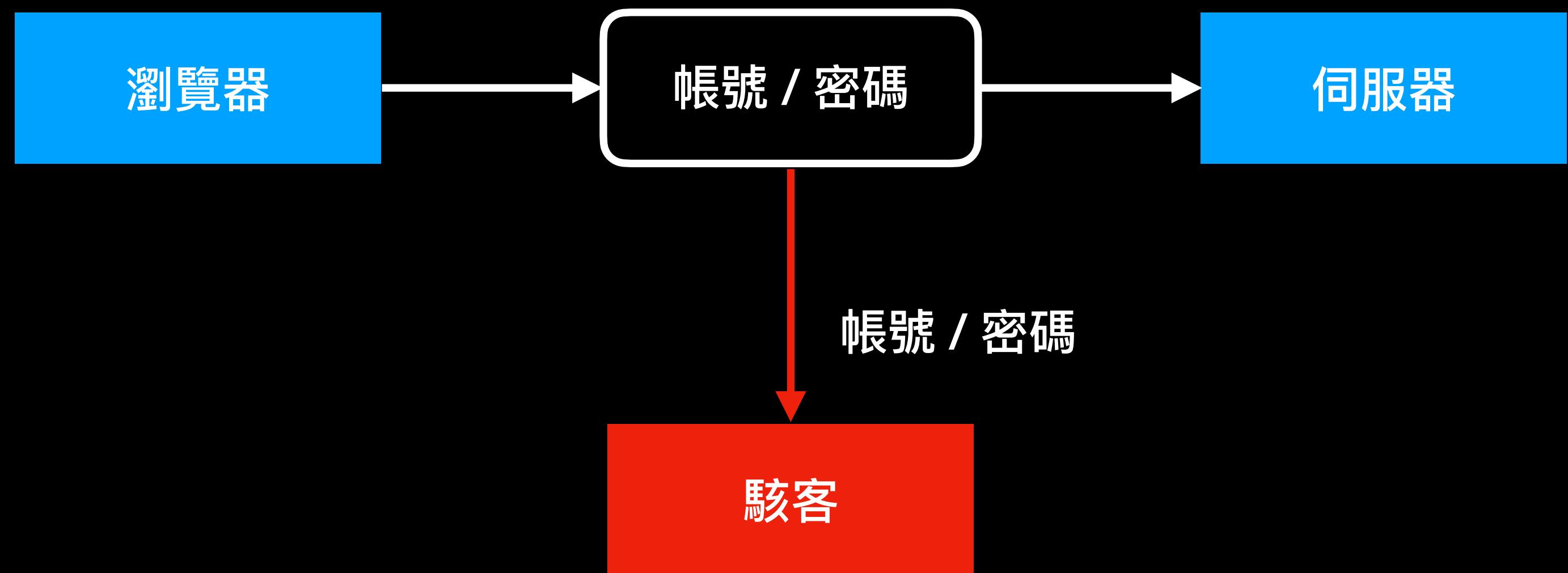
## Terminology

- 在不安全的環境中建立安全通訊管道的技術，如：戰爭時的軍事通訊



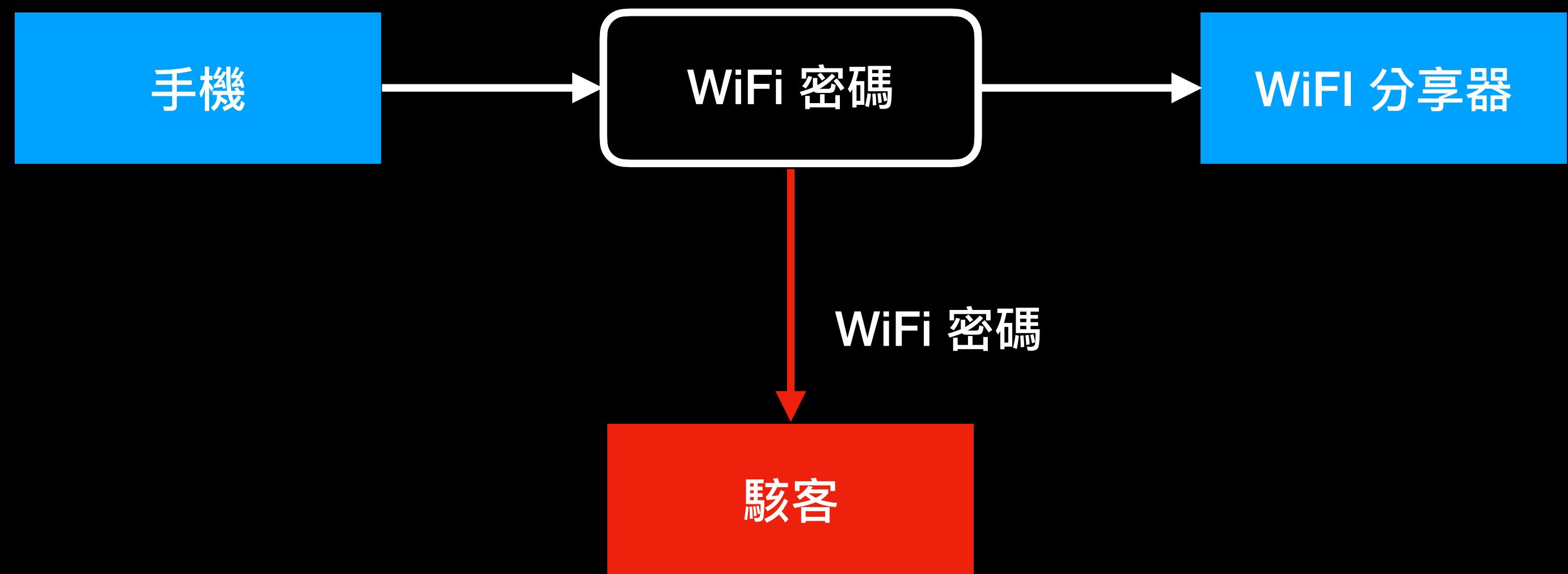
# 密碼學在生活中的實際應用

- 瀏覽網頁 (HTTPS)
- WiFi 連線
- 線上刷卡/轉帳 (OTP)
- VPN
- 加密硬碟



# 密碼學在生活中的實際應用

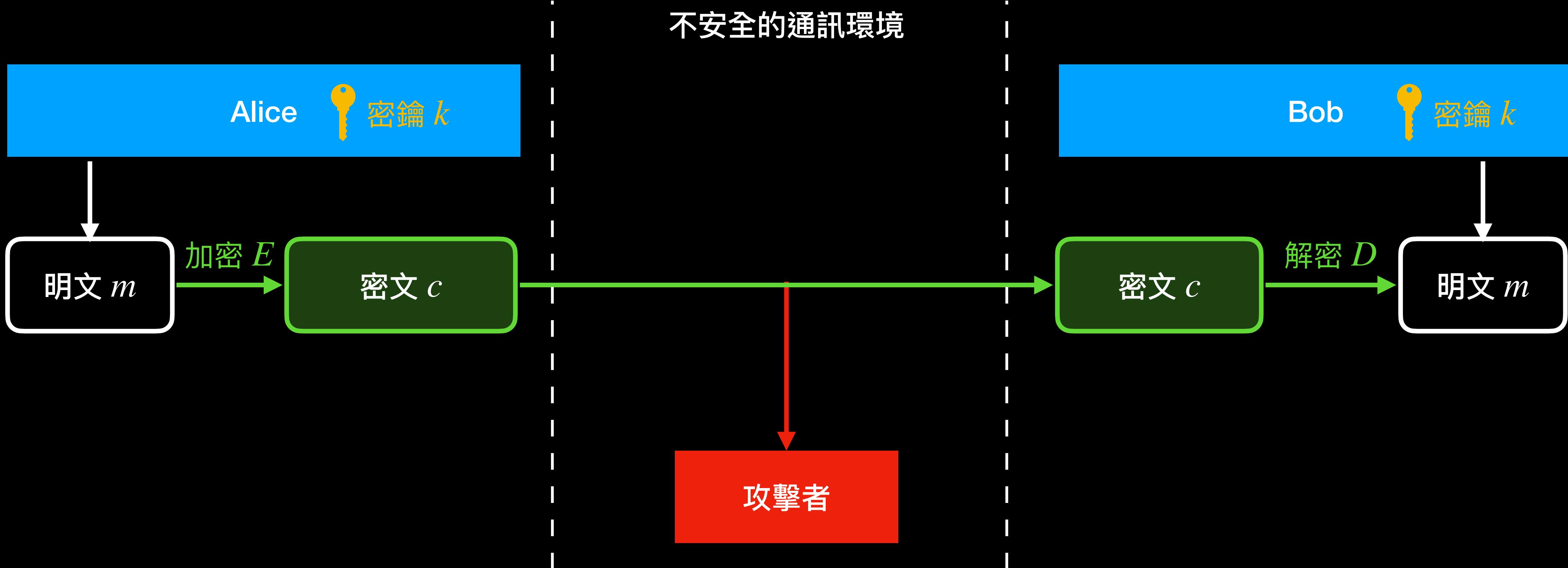
- 瀏覽網頁 (HTTPS)
- WiFi 連線
- 線上刷卡/轉帳 (OTP)
- VPN
- 加密硬碟



# 名詞與符號

- 明文  $\text{plaintext} / m$ ：要傳遞的訊息
- 密文  $\text{ciphertext} / c$ ：經過加密後的明文
- 密鑰  $\text{key} / k$ ：加解密時使用的「鑰匙」，必須避免洩漏以確保加密安全性
- 加密  $\text{encryption} / E$ ：將明文轉換成密文的程序， $E(m, k) = c$
- 解密  $\text{decryption} / D$ ：將密文轉換成明文的程序， $D(c, k) = m$

# 名詞與符號



# 柯克霍夫原則

## Kerckhoff's Principle

- 即使密碼系統的所有細節都被瞭解，只要密鑰沒有洩漏，密文就是安全的
- 設計密碼系統時，應以系統被完全公開也不影響安全性為前提
- 密鑰是密碼系統安全性的關鍵

# 密碼學的分類

- 古典密碼學
- 現代密碼學
  - 對稱式加密
    - 串流加密
    - 區塊加密
  - 非對稱式加密

# 古典密碼學

# 古典密碼學

- 歷史上曾使用過的秘密通訊方法
- 通常存在數學或實務上的弱點，較容易破解

JGNQ ECGU  
T



HELLO CAESAR

# 古典密碼學的分類

- 替換式加密
- 換位式加密
- 其他

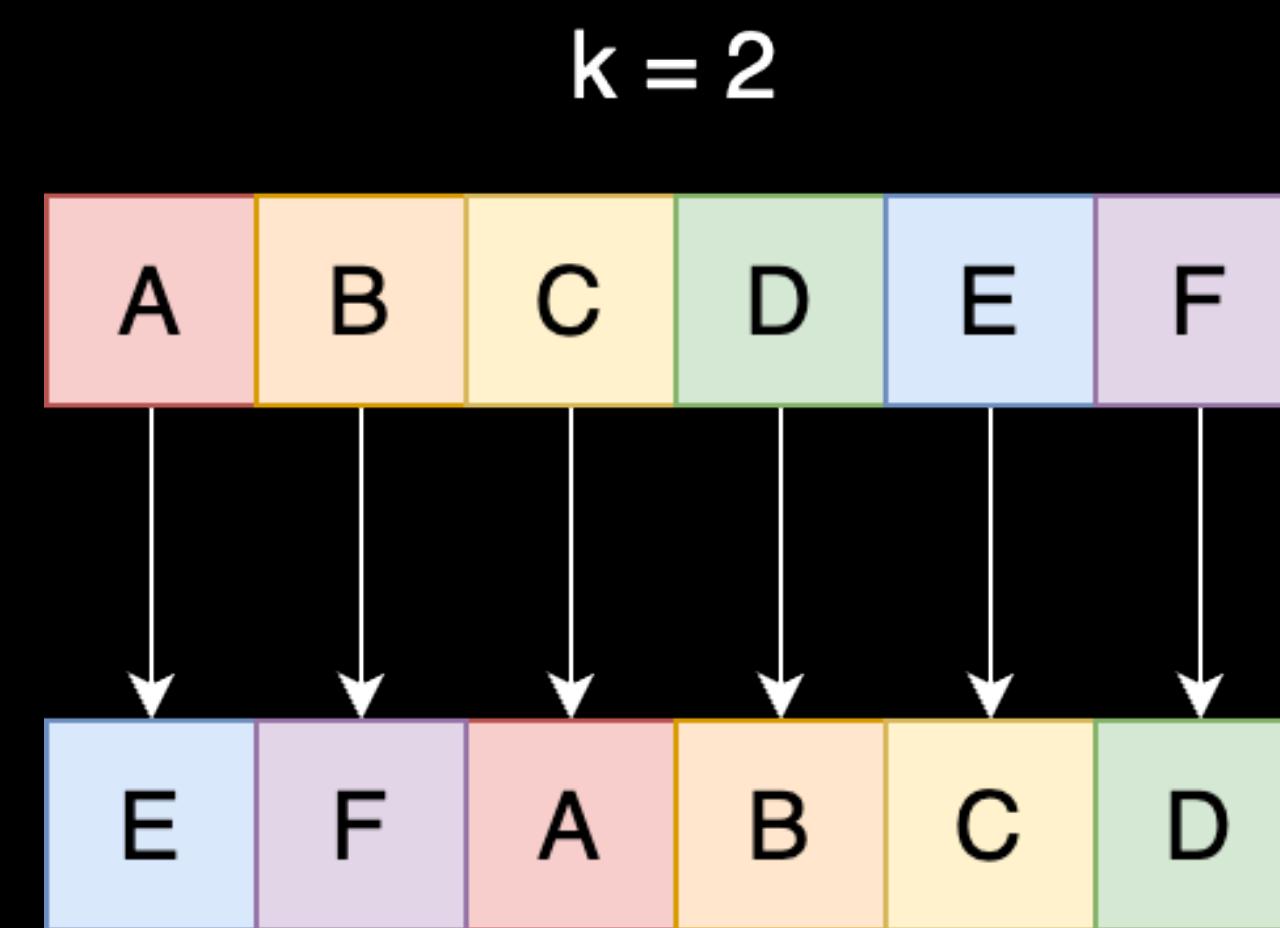
# 替換式加密

- 將明文中的字母以特定的方式進行替換
  - 組成元件：字母表、密鑰、替換方式
- 凱薩密碼
- 維吉尼亞密碼
- 仿射密碼



# 凱薩密碼 Caesar Cipher

- 密鑰： $k$
- 加密：將明文的每個字元在字母表上的位置偏移  $k$  位，得到密文
- 解密：將密文的每個字元在字母表上的位置偏移  $-k$  位，得到明文



# 凱薩密碼 Caesar Cipher

- 範例
  - 密鑰 :  $k=2$
  - 字母表 : A-Z
  - 明文 : HELLO CAESAR
  - 密文 : JGNQ ECGUCT

# Lab : ABCTF 2016 : caesar-salad-10

- Flag 開頭為 abctf{
- 字母表 : a–z
- 嘗試用題目提供的線上工具，或是練習寫 Python

## 1. ABCTF 2016 : caesar-salad-10

10

Most definitely the best salad around. Can you decrypt this for us?

xyzqc{t3\_qelrdeq\_t3\_k33a3a\_lk3\_lc\_qe3p3}

# Lab : ABCTF 2016 : caesar-salad-10

```
enc_flag = 'xyzqc{t3_qelrdeq_t3_k33a3a_lk3_lc_qe3p3}'$  
#  
# abctf{$  
$  
alphabets = 'abcdefghijklmnopqrstuvwxyz'$  
for k in range(len(alphabets)):$  
    flag = ''$  
    for c in enc_flag:$  
        if c in alphabets:$  
            flag += alphabets[(alphabets.index(c) + k) % len(alphabets)]$  
        else:$  
            flag += c$  
$  
        if 'abctf' in flag:$  
            print(flag)$
```

# 維吉尼亞密碼 Vigenere Cipher

- 密鑰：字串  $k$
- 加密：對於明文的第  $i$  個字元
  1. 密鑰的第  $i$  個字元作為字母表的 row
  2. 明文的第  $i$  個字元作為字母表的 column
  3. 字母表中對應之字元為密文
- 解密：對於密文的第  $i$  個字元
  1. 密鑰的第  $i$  個字元作為字母表的 row
  2. 該 row 中密文字元所在的 column 即為明文

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

字母表

# 維吉尼亞密碼 Vigenere Cipher

- 加密
  - 密鑰 : KEYKEY
  - 明文 : CIPHER
  - 密文 : M

	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
A	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

# 維吉尼亞密碼 Vigenere Cipher

- 加密
  - 密鑰 : KEYKEY
  - 明文 : CIPHER
  - 密文 : MM

	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
A	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

# 維吉尼亞密碼 Vigenere Cipher

- 加密：
  - 密鑰：KEYKEY
  - 明文：CIPHER
  - 密文：MMNRIP

	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
A	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

# 維吉尼亞密碼 Vigenere Cipher

- 解密：
  - 密鑰：KEYKEY
  - 密文：MMNRIP
  - 明文：C

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

# 維吉尼亞密碼 Vigenere Cipher

- 解密
  - 密鑰 : KEYKEY
  - 密文 : MMNRIP
  - 明文 : CI

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

# Lab : Vigenère cipher

- 用題目提供的線上工具試試看

## 11. Vigenère cipher

100

Crack the cipher: vhixoieemksktorywzvhxzijqni

Your clue is:

"caesar is everything. But he took it to the next level."

# 仿射密碼 Affine Cipher

- 密鑰： $a, b$
- 字母表長度： $p$
- 加密： $E(m) = am + b = c \pmod{p}$ 
  - $a = 1$  時即為凱薩密碼
- 怎麼解密？

# 反元素

## Inverse Element

- 設  $(S, *)$  為帶有二元運算 \* 的集合  $S$
- 單位元素  $e$ 
  - $e \in S$
  - $\forall a \in S, a * e = a$  且  $e * a = a$
- 反元素  $x$ 
  - $a \in S, a * x = e$  且  $x * a = e$
  - $x \in S$
  - $x$  稱作  $a$  的反元素，又寫作  $a^{-1}$
  - 反元素不一定存在

# 反元素

## Inverse Element

- $(\mathbb{R}, +)$ 
  - 單位元素 : 0
  - $a$  的反元素 :  $-a$
- $(\mathbb{R}, \times)$ 
  - 單位元素 : 1
  - $a$  的反元素 :  $\frac{1}{a}$
  - 0 沒有反元素

# 模運算的乘法反元素

## Modular Multiplicative Inverse Element

- 使得以下等式成立的整數  $x$ ，即為  $a$  的模運算乘法反元素  $a^{-1}$

$$ax \equiv 1 \pmod{p}$$

- $3 \times 2 \equiv 1 \pmod{5} \Rightarrow 3^{-1} = 2 \pmod{5}$
- $7 \times 15 \equiv 1 \pmod{26} \Rightarrow 7^{-1} = 15 \pmod{26}$
- 模反元素算法
  - 暴力硬算
  - 擴展歐幾里德 Extended Euclidean Algorithm

# 仿射密碼 Affine Cipher

- 密鑰： $a, b$
- 字母表長度： $p$
- 加密： $E(m) = am + b = c \pmod{p}$
- 解密： $D(c) = a^{-1}(c - b) = m \pmod{p}$
- 條件
  - $a, p$  必須互質，才能確保  $a^{-1}$  存在

$$\begin{aligned} D(c) &= a^{-1}(c - b) \pmod{p} \\ &= a^{-1}(am) \pmod{p} \\ &= m \end{aligned}$$

# 仿射密碼 Affine Cipher

- 加密
  - 密鑰 :  $a=7, b=13$
  - 明文 : **affine**
  - 密文 : **n**
  - $0 \times 7 + 13 \pmod{26} = 13 = n$

a	b	c	...	<b>n</b>	...	z
0	1	2	...	<b>13</b>	...	25

# 仿射密碼 Affine Cipher

- 加密
  - 密鑰 :  $a=7, b=13$
  - 明文 : affine
  - 密文 : nw
  - $5 \times 7 + 13 \pmod{26} = 22 = w$

a	b	c	...	w	...	z
0	1	2	...	22	...	25

# 仿射密碼 Affine Cipher

- 加密

- 密鑰 :  $a=7, b=13$

- 明文 : affine

- 密文 : nwwrap

```
alphabets = 'abcdefghijklmnopqrstuvwxyz'$  
$  
m = 'affine'$  
a = 7$  
b = 13$  
p = len(alphabets)$  
$  
c = ''$  
for x in m:$  
    x = alphabets.index(x)$  
    y = (a*x + b) % p$  
    c += alphabets[y]$  
$  
print('m:', m)$  
print('c:', c)$
```

```
> python3 afi.py  
m: affine  
c: nwwrap  
a   f   f   i   n   e  
0   5   5   8   13  4  
13  22  22  17  0   15  
n   w   w   r   a   p
```

# 仿射密碼 Affine Cipher

- 解密
  - 密鑰 :  $a=7, b=13$
  - 密文 :  $nwwrap$
  - 明文 :  $a$ 
    - $(13 - 13) \times 7^{-1} \pmod{26} = 0 = a$

$a$	b	c	...	$n$	...	$z$
0	1	2	...	13	...	25

# 仿射密碼 Affine Cipher

- 解密
  - 密鑰 :  $a=7, b=13$
  - 密文 : n<sub>w</sub>wrap
  - 明文 : af
  - $(22 - 13) \times 7^{-1} \pmod{26} = 5 = f$

a	b	c	...	w	...	z
0	1	2	...	22	...	25

# Lab : School CTF 2015: affine-cipher-100(教)

- $a=4$
- $b=15$
- 字母表 : a-z\_
- 密文
  - ifpmluglesecdlqp\_rclfrseljpkq
- Flag 開頭的 flag\_is\_ 要刪掉

## 6. School CTF 2015: affine-cipher-100(教)

150

Decrypt the message, which was encrypted with following rules:

for each letter of cipher text its position in the alphabet is the position of the original letter multiplied by 4 and shifted by 15 character

shift over alphabet is cyclic, so 'z' shifted by 1 is symbol \_ and symbol \_ shifted by 1 is 'a'

alphabet consists of letters from 'a' to 'z' and symbol \_

letter 'a' has position 0, symbol '\_' has position 26 ( following 'z' )

Encrypted message: ifpmluglesecdlqp\_rclfrseljpkq

What is the flag?

# Lab : School CTF 2015: affine-cipher-100(教)

```
alphabets = 'abcdefghijklmnopqrstuvwxyz_'$  
$  
a = 4$  
b = 15$  
p = len(alphabets)$  
c = 'ifpmluglesecdlqp_rclfrseljpkq'$  
$  
for x in range(p):$  
    if (a * x) % p == 1:$  
        mod_inv = x$  
        break$  
print('found mod inv:', mod_inv)$  
$  
m = ''$  
for x in c:$  
    x = alphabets.index(x)$  
    x = ((x - b) * mod_inv) % p$  
    m += alphabets[x]$  
$  
print(m)$
```

```
› python3 afi_solve.py  
found mod inv: 7  
flag_is_every_haxor_love_math
```

# 換位式加密

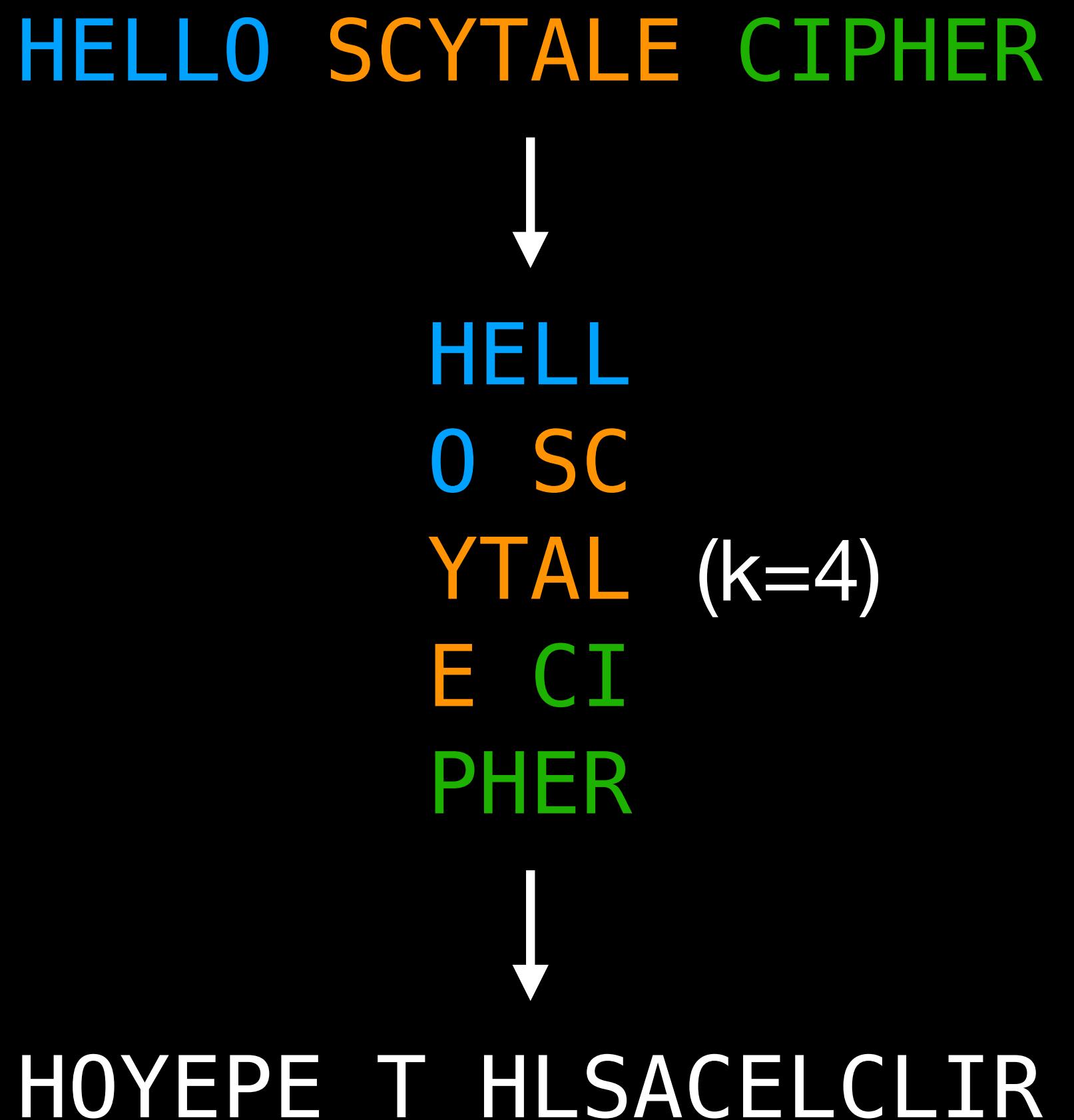
- 將字母以特定的方法重新排序
- 密碼棒
- 櫃欄密碼



<https://en.wikipedia.org/wiki/Scytale#/media/File:Skytale.png>

# 密碼棒 Scytale

- 密鑰 : k
- 加密
  - 將明文以橫向寫入 k column 的表格
  - 以直向閱讀表格得到密文
- 解密：直向寫入表格，橫向閱讀



# Lab : EKOPARTY CTF 2015: SCYTCRYPTO

- 嘗試用 Python 解題
- 答案以 EKO 開頭，是一個有意義的英文句子（沒有空白）

## 4. EKOPARTY CTF 2015: SCYTCRYPTO

50

Decrypt this strange word: ERTKSOOTCMCHYRAFYLIPL

# Lab : EKOPARTY CTF 2015: SCYTCRYPTO

```
m = 'ERTKS00TCMCHYRAFYLIPL'$  
print('len(m):', len(m))$  
$  
r, c = 7, 3$  
for i in range(c):$  
    for j in range(r):$  
        print(m[j*c + i], end='')$
```

E	R	T
K	S	0
O	T	C
M	C	H (k=7)
Y	R	A
F	Y	L
I	P	L

```
|> python3 scy.py  
len(m): 21  
EK0MYFIRSTCRYPTOCHALL
```

↓  
EK0MYFIRSTCRYPTOCHALL

# CTF 中出現過的其他古典密碼學密碼

- 柵欄密碼 Rail Fence Cipher
- 路徑密碼 Route Cipher
- 列位移密碼 Column Transposition Cipher
- 豬圈密碼 Pigen Cipher
- 跳舞小人密碼
- ...

> .|.|•|•|V >|□| □ | V •|□|>  
X M A R K S T H E S P O T



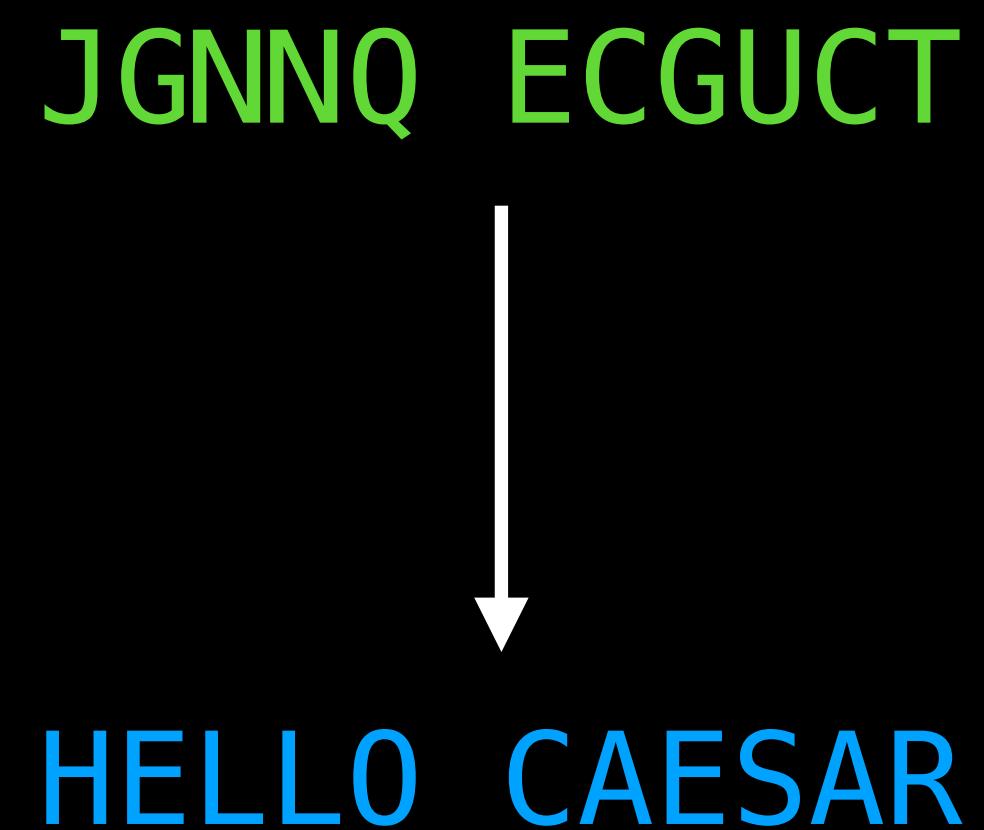
# 這麼多古典密碼學加密



# 頻率分析

## Letter Frequency Analysis

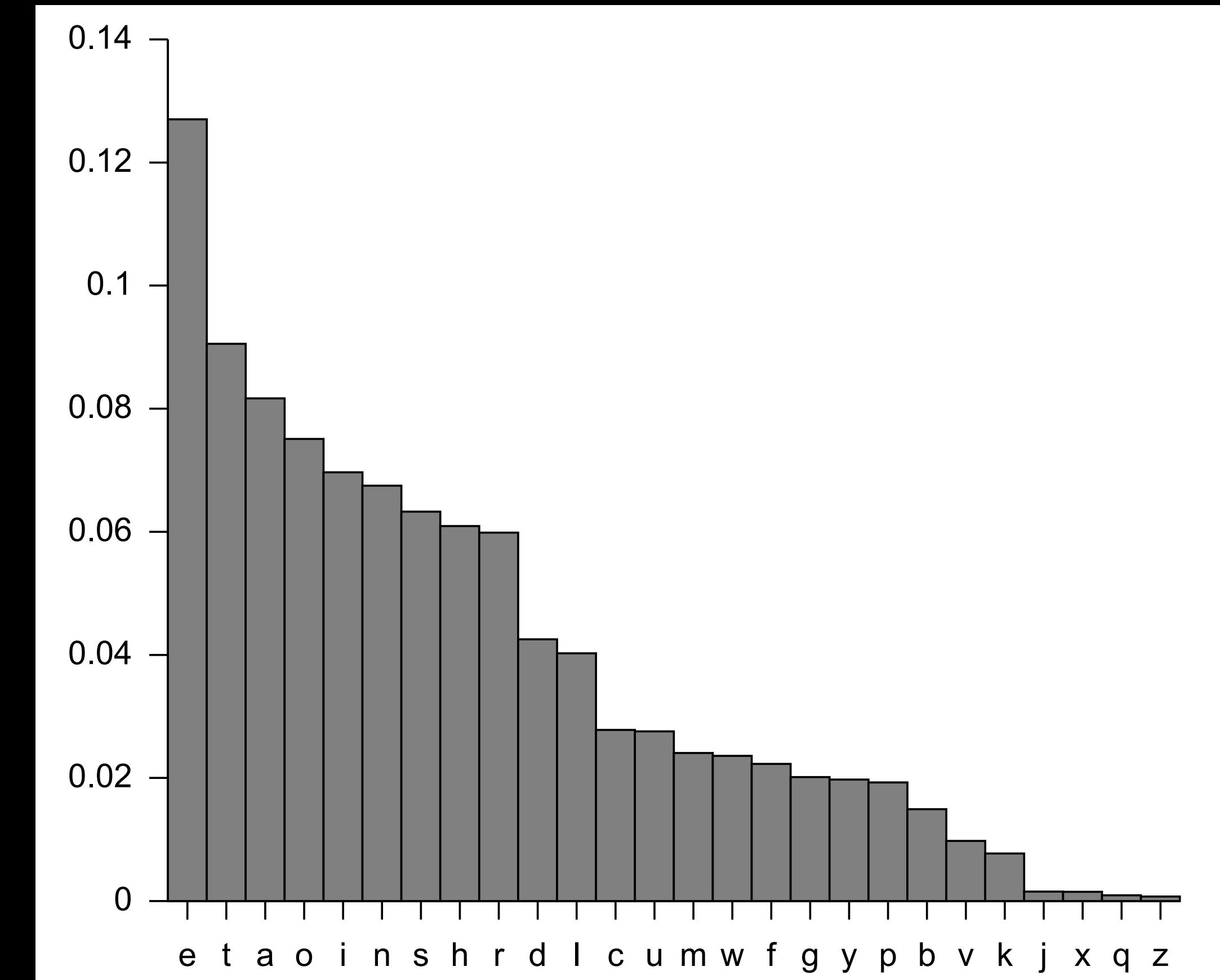
- 多數替換式密碼的特性
    - 明文與密文字母間有一對一的對應關係
    - 各字母的出現頻率不變
- 頻率分析
- 英文中最常見的字母是 e
  - 推測密文最常見的字母應該對應到 e



# 頻率分析

## Letter Frequency Analysis

- 根據統計得到字母及其組合的出現頻率
  - 字母 : e, t, a, o, i, ...
  - 2-gram : th, he, in, er, an
  - n-gram
- 從密文字母的出現頻率，回推可能的原字母
- 需要較長密文才能得到穩定的出現頻率
- 線上工具
  - [quipquip](#)



# Lab : EasyCTF-2014:A Simple Cipher

- 使用線上工具試試看

## 10. EasyCTF-2014:A Simple Cipher

20

Cryptography is hiding messages in plain sight. Although they can be viewed, they are usually unreadable without the use of a special key. Messages can be encrypted and then sent to another person who then decrypts the ciphertext (encrypted message) using their special key into plaintext (readable text). Try your hand at this Caesar cipher:

IGKYGX HKIGSK ZNK LOXYZ XUSGT MKTXGR ZU IXUYY HUZN CNKT NK  
HAORZ G HXOJMK GIXUYY ZNK XNOTK GTJ IUTJAIZKJ ZNK LOXYZ OTBGYOUT  
UL HXOZGOT.ZNKYK GINOKBKSCTZY MXGTZKJ NOS ATSGZINKJ SOROZGXE  
VUCKX GTJ ZNXKGZKTKJ ZU KIROVYK ZNK YZGTJOTM UL VUSVKE, CNU NGJ  
XKGROMTKJ NOSYKRL COZN ZNK YKTGZK GLZKX ZNK JKGZN UL IXGYYAY OT  
53 HI. COZN ZNK MGRROI CGXY IUTIRAJKJ, ZNK YKTGZK UXJKXKJ IGKYGX ZU

# 古典密碼學破解方法

1. 根據題目敘述、字母表、密文字元等推測可能的加密方式
  2. 知道加密方式：寫腳本暴力破解 key
  3. 替換式加密：頻率分析
  4. 其他：線上工具、特定試驗方法（維吉尼亞密碼）
- Tip
    - 善用 Flag 格式等已知資訊
    - 嘗試先找到 Flag 格式的開頭

# 古典密碼學破解工具

- 加解密網站：[CyberChef](#)、[Cryptii](#)
- 頻率分析：[quipqiup](#)
- Python
- Google
- ChatGPT

休息一下~

# 現代密碼學

# 現代密碼學

- 以數學為基礎設計，經過多年公開的審核與檢視，被公認為安全的加密演算法
  - 安全性
  - 計算成本
  - 計算時的其他特點（軟體優化、泛用性、複雜程度）
- 安全性並非經過數學證明，最新研究或算力提升可能使其變得不安全
  - RC4、DES

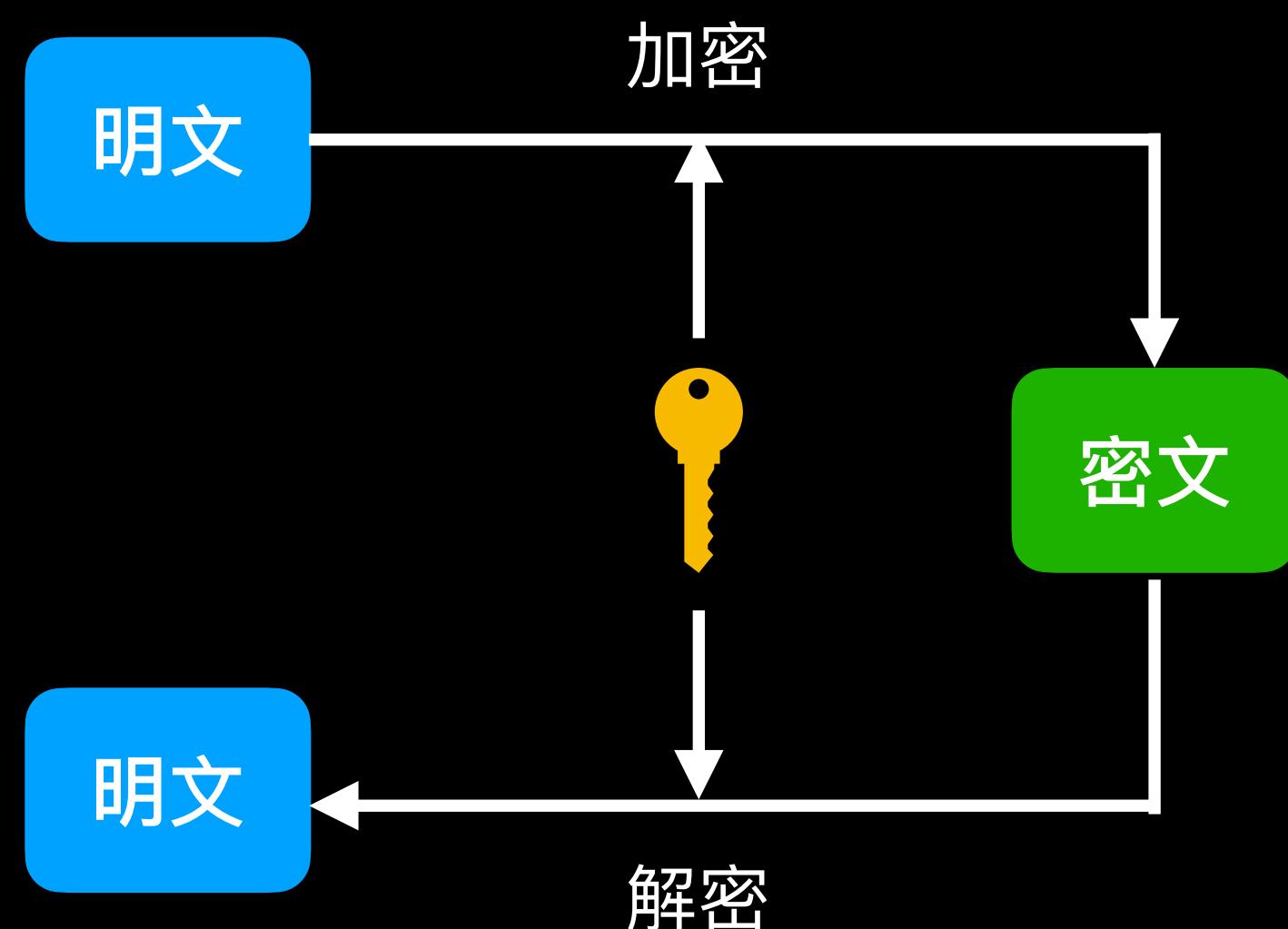
# 現代密碼學的分類

- 對稱式加密
  - 串流加密
  - 區塊加密
- 非對稱式加密

# 對稱式加密 v.s. 非對稱式加密

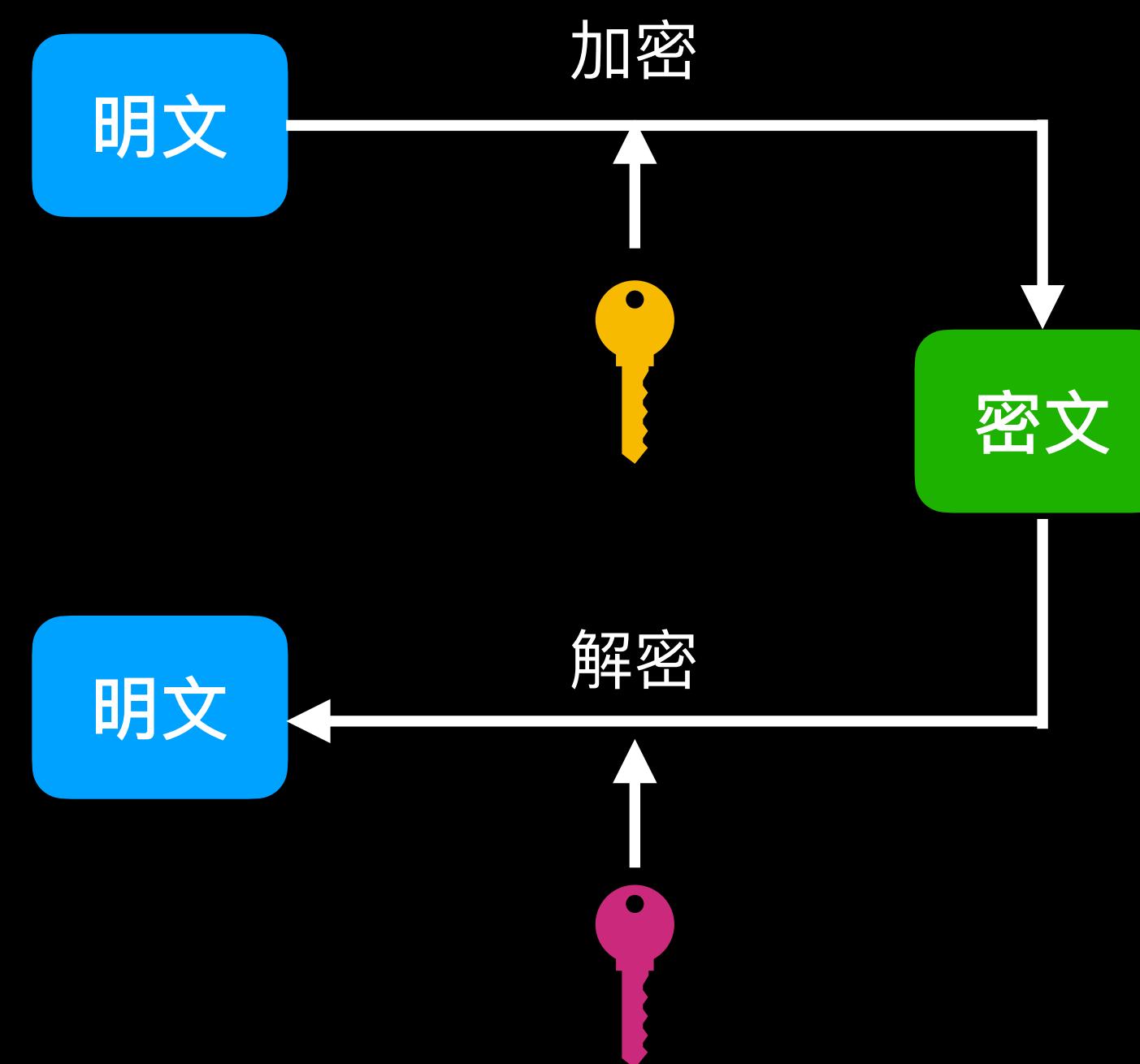
## Symmetric v.s. Asymmetric Encryption

對稱式加密



加、解密使用**相同**金鑰

非對稱式加密



加、解密使用**不同**金鑰

# 對稱式加密 v.s. 非對稱式加密

## Symmetric v.s. Asymmetric Encryption

	對稱式加密	非對稱式加密
計算速度	快	慢
金鑰	同一把密鑰	公鑰、私鑰
功能	加密訊息	金鑰交換、數位簽章、...
演算法	RC4、DES、AES	RSA、D-H、ECC

# 串流加密 v.s. 區塊加密

## Stream Cipher v.s. Block Cipher

- 串流加密

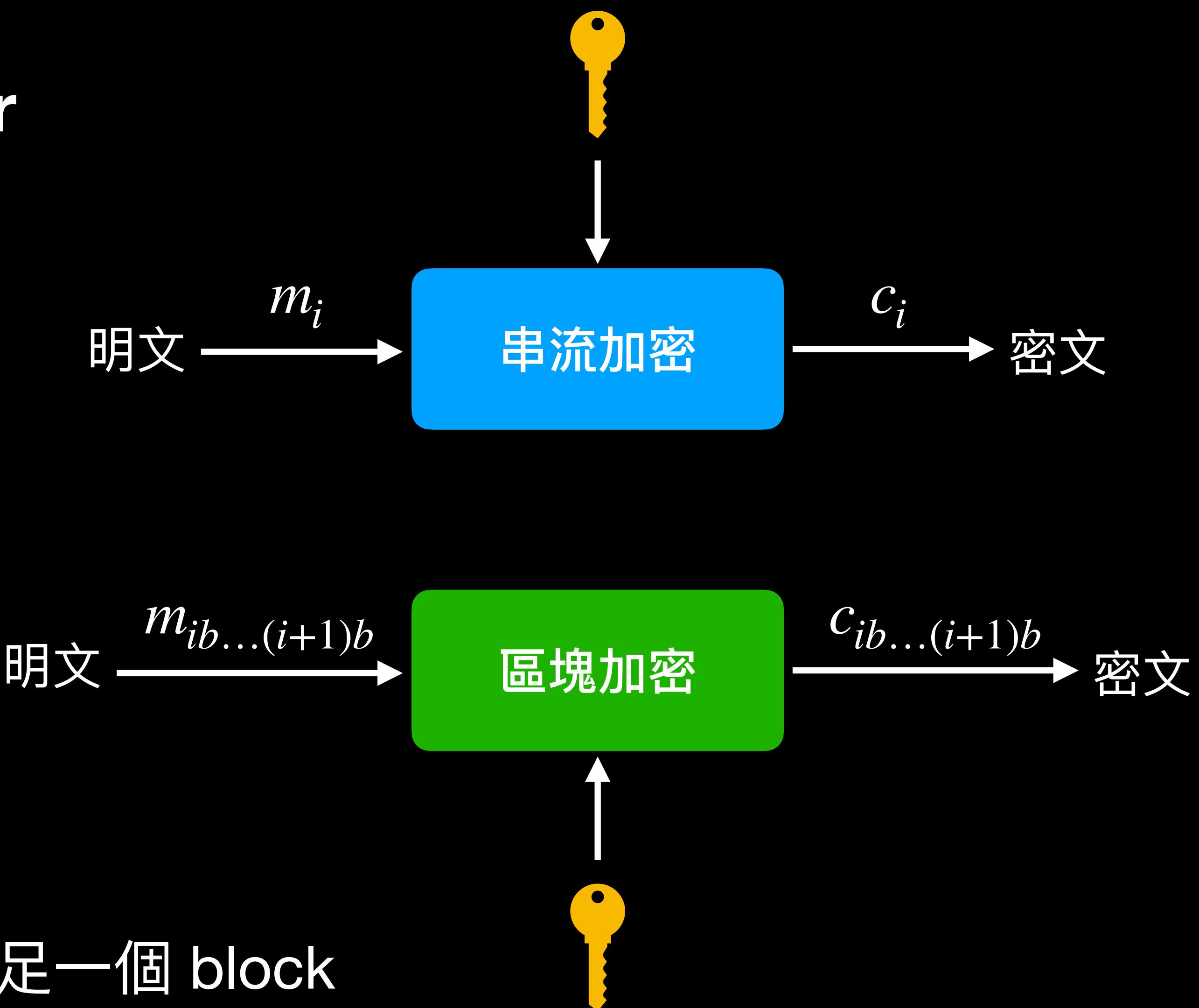
- 一次加密一個 bit

- 區塊加密

- 一次加密一個 block

- block 為演算法預先定義的大小  $b$

- 剩餘資料不足時需做 padding，補足一個 block



# 串流加密

# XOR

- 兩邊數值不同時為 True，否則為 False
- 符號
  - 數學： $\oplus$
  - 程式：`^`
- 範例
  - $001101 \oplus 101010 = 100111$

XOR	0	1
0	0	1
1	1	0

XOR 真值表

# XOR

- 現代密碼學加密系統的重要元件
  - Perfectly Balanced
  - 可逆性
  - 容易於軟硬體實作

XOR	0	1
0	0	1
1	1	0

XOR 真值表

# XOR 特性：Perfectly Balanced

- 若密鑰為完全隨機，則密文輸出 0 與 1 的機率相同
- 無法從密文中得到任何資訊

$$P(m = 1) = p, P(k = 1) = 0.5$$

$$P(m \oplus k = 0)$$

$$= P(m = 0) \times P(k = 0) + P(m = 1) \times P(k = 1)$$

$$= (1 - p) \times 0.5 + p \times 0.5$$

$$= 0.5$$

明文	密鑰	密文
0	0	0
0	1	1
1	0	1
1	1	0

# XOR 特性

- 可逆性
  - XOR 的逆運算同為 XOR
  - $m \oplus k = c$
  - $c \oplus k = m$
- 容易於軟硬體實作
  - 數個基本邏輯閘

XOR	0	1
0	0	1
1	1	0

XOR 真值表

# 串流加密

- 基於 XOR 良好特性而生的加密方式
- 加密： $m \oplus k = c$
- 解密： $c \oplus k = m$

# Lab: Echo in XOR

- 點擊 `chal.py` 下載題目程式碼
- 在 Kali 虛擬機上，使用 `nc 210.70.138.222 11001` 連上題目



# Lab: XOR Cipher Chaining

- 點擊 `chal.py` 下載題目程式碼
- 在 Kali 虛擬機上，使用 `nc 210.70.138.222 11002` 連上題目



# Lab: Echo in XOR

- $\text{enc\_flag} = \text{flag} \oplus \text{key}$
- $\text{enc\_msg} = \text{msg} \oplus \text{key}$

```
20  with open('flag', 'r') as f:  
21  |   flag = f.read()  
22  key = gen_key(flag)  
23  enc_flag = xor(flag, key)  
24  
25  #print('key:', key)  
26  print('encrypted flag:', enc_flag)  
27  
28  msg = input(f'input your msg (len == {len(flag)}): ')  
29  while len(msg) != len(flag):  
30  |   print(f'[!] len(msg) != {len(flag)})')  
31  |   msg = input(f'input your msg (len == {len(flag)}): ')  
32  enc_msg = xor(msg, key)  
33  print('encrypted msg:', enc_msg)
```

# Lab: Echo in XOR

- $\text{enc\_flag} = \text{flag} \oplus \text{key}$
- $\text{enc\_msg} = \text{msg} \oplus \text{key}$
- 解法
  - 輸入  $\text{msg} = \text{enc\_flag}$
  - $\text{enc\_msg} = \text{enc\_flag} \oplus \text{key} = \text{flag}$

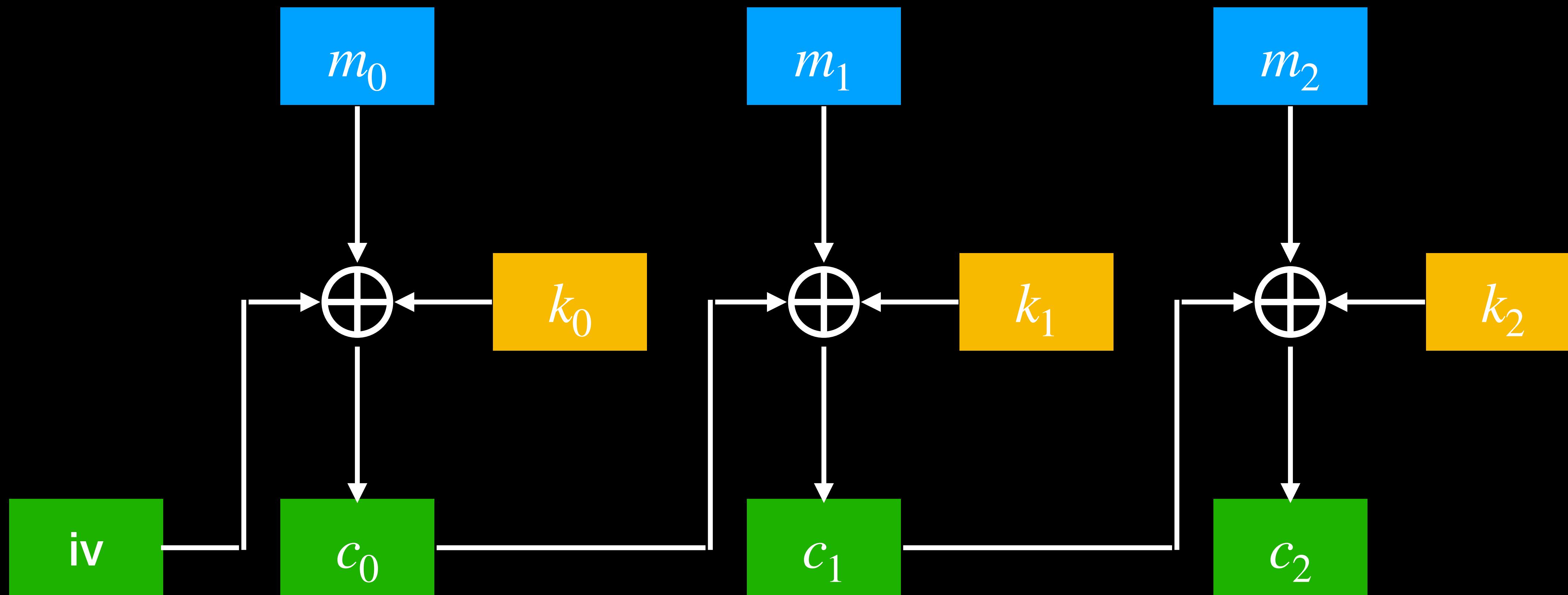
```
> nc 210.70.138.222 11001
encrypted flag: t8ur04IFm65wVq09X94eAmV99462
input your msg (len == 28): t8ur04IFm65wVq09X94eAmV99462
encrypted msg: FLAG{x0r_tw1c4_t0_g3t_flag}
```

# Lab: XOR Cipher Chaining

- $c_i = m_i \oplus k_i \oplus c_{i-1}$
- $c_{-1} = iv = 37$
- $\text{len(key)} = 5$

```
4  ✓ def gen_key(flag):
5      |   key = ''.join(random.choices(ascii_letters + digits, k=5))
6      |   return key
7
8  ✓ with open('flag', 'r') as f:
9      |   flag = f.read()
10     |   iv = 37
11     |   key = gen_key(flag)
12     |   enc_flag = []
13     |   prev_c = iv
14  ✓ for i in range(len(flag)):
15      |   c = ord(flag[i]) ^ ord(key[i % len(key)]) ^ prev_c
16      |   enc_flag.append(c)
17      |   prev_c = c
18
19     #print('key:', key)
20     print('iv:', iv)
21     print('encrypted flag:', enc_flag)
```

# Lab: XOR Cipher Chaining



# Lab: XOR Cipher Chaining

- $c_i = m_i \oplus k_i \oplus c_{i-1}$
- $iv = 37$
- $\text{len}(\text{key}) = 5$
- 解法：用 FLAG 格式 ( $m_0 \dots m_4$ ) 計算出 key
  - $k_0 = iv \oplus c_0 \oplus m_0$
  - $k_1 = c_0 \oplus c_1 \oplus m_1$

```
iv = 37
enc_flag = [80, 107, 64, 62, 4, 95, 77, 19, 78,
            # compute key
            prev_c = iv
            header = 'FLAG{'
            key = []
            for i in range(len(header)):
                m = prev_c ^ enc_flag[i] ^ ord(header[i])
                key.append(m)
                prev_c = enc_flag[i]

            # decrypt flag
            flag = header
            for i in range(len(flag), len(enc_flag)):
                m = prev_c ^ enc_flag[i] ^ key[i % len(key)]
                flag += chr(m)
                prev_c = enc_flag[i]
            print(flag)
```

# 串流加密

- 基於 XOR 良好特性而生的加密方式
- 加密： $m \oplus k = c$
- 解密： $c \oplus k = m$
- 因為 XOR 是 Perfectly Balanced，串流加密的安全性完全仰賴密鑰  $k$  是否足夠隨機
- 生成密鑰  $k$  的方式
  - 隨機數生成器
  - 以生成密鑰的方式衍生出不同的串流加密演算法

# 隨機數生成器

## Random Number Generator

- 真隨機數生成器 TRNG
- 偽隨機數生成器 PRNG
- 密碼學安全偽隨機數生成器 CSPRNG

# 真隨機數生成器 TRNG

## True Random Number Generator

- 從物理過程得到的真正隨機
  - 無法推測
  - 無法重現
- 範例
  - 硬幣、擲筊
  - 電子雜訊、熱雜訊

# 真隨機數生成器 TRNG

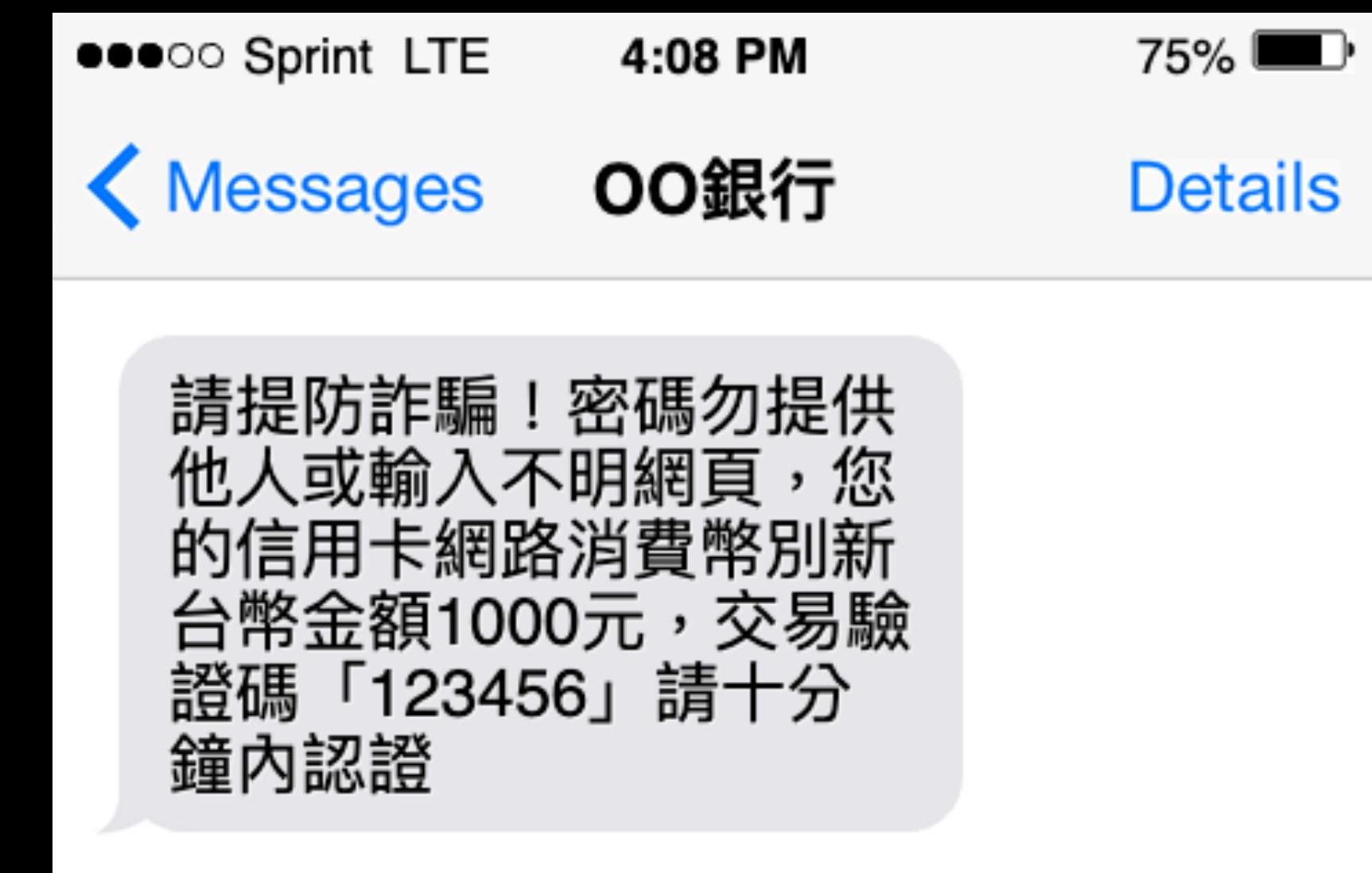
## True Random Number Generator

- 實務上不易使用
  - TRNG 需要專門的硬體
  - 無法解密
    - 無法重現結果
    - 傳送密鑰給對方
- 實際用途
  - 生成偽隨機數生成器的初始值
  - 生成不需要分享的私鑰

# 一次性密碼本 OTP

## One-Time Pad

- 數學證明上安全的加密系統（無限算力也無法暴力破解）
- 條件
  - 真正隨機產生的密鑰
  - 只有通訊雙方知道密鑰
  - 密鑰與明文一樣長，且不可重複使用
- 相似技術
  - 簡訊 6 位數驗證碼
  - Google Authenticator



# 偽隨機數生成器 PRNG

## Pseudo Random Number Generator

- 紿定初始值  $\text{seed}$ ，用數學方法連續生成一串看似隨機的數字

$$s_0 = \text{seed}$$

$$s_{i+1} = f(s_i, s_{i-1}, \dots, s_{i-t})$$

- 特性
  - 近似隨機：輸出統計上近似隨機的數字
  - 可重現性：相同的  $\text{seed}$ ，相同的輸出結果
- 範例：C 的 `rand()`, Python 的 `random`

# Lab : Just a Little Lucky

- 點擊 chal.py 下載題目程式碼
- 在 Kali 虛擬機上，使用 nc 210.70.138.222 11003 連上題目



# Lab : Just a Little Lucky

```
26 int main(void) {  
27     init();  
28  
29     srand(0);  
30  
31     unsigned user, bot;  
32     for (unsigned i = 0 ; i < 5; i++) {  
33         // bot move  
34         bot = rand() % 3;  
35  
36         // user move  
37         printf("input (0-rock 1-paper 2-scissor): ");  
38         scanf("%d", &user);  
39         if (user > 2) {  
40             printf("Bad move: %d\n", user);  
41             return 1;  
42         }
```

```
44     printf("You: %d, Bot: %d\n", user, bot);  
45     if ((user == 0 && bot == 2) || \  
46         (user == 1 && bot == 0) || \  
47         (user == 2 && bot == 1)) {  
48         printf("You win!\n");  
49     } else if (user == bot) {  
50         printf("Tie. Maybe next time.\n");  
51         return 1;  
52     } else {  
53         printf("You lose! Try again.\n");  
54         return 1;  
55     }  
56 }  
57  
58     printf("Flag: %s\n", flag);  
59     return 0;  
60 }
```

# Lab : Just a Little Lucky

- 解法 1
  - 相同的 seed，相同的輸出
  - 使用 rand()，在本地計算對方出拳

```
ksu@ksu-VirtualBox:~/crypto/fix_seed$ ./leak_ans
1
1
0
1
2
```

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    srand(0);
    for (unsigned i = 0 ; i < 5; i++) {
        printf("%d\n", rand() % 3);
    }
    return 0;
}
```

# Lab : Just a Little Lucky

- 解法 2

- 相同的 seed，相同的輸出
- 對方每次出拳都相同，多試幾次

```
ksu@ksu-VirtualBox:~/crypto/fix_seed$ nc 127.0.0.1 11003
input (0-rock 1-paper 2-scissor): 2
You: 2, Bot: 1
You win!
input (0-rock 1-paper 2-scissor): 2
You: 2, Bot: 1
You win!
input (0-rock 1-paper 2-scissor): 1
You: 1, Bot: 0
You win!
input (0-rock 1-paper 2-scissor): 2
You: 2, Bot: 1
You win!
input (0-rock 1-paper 2-scissor): 0
You: 0, Bot: 2
You win!
Flag: FLAG{5eed_15_crucial_1n_prn9}
```

# 密碼學安全偽隨機數生成器 CSPRNG

## Cryptographically Secure Pseudo Random Number Generator

- 特性
  - 近似隨機：輸出統計上近似隨機的數字
  - 可重現性：相同的 seed，相同的輸出結果
  - 無法推測：給定任意第  $i \sim i+n$  個輸出，無法推測之前或之後的輸出
- 絝大多數 PRNG 都不是 CSPRNG
- 範例：/dev/urandom

# 偽隨機數生成器演算法

## Pseudo Random Number Generator

- 線性同餘 LCG
- 線性反饋移位暫存器 LFSR

# 線性同餘偽隨機數生成器 LCG

## Linear Congruential Generator

- $a, b$  選擇恰當時，可作為 PRNG
- 曾用於實作 `rand()`
- 基於 LCG 的串流加密
  - 密鑰： $a, b, s_0$

$$s_0 = \text{seed}$$

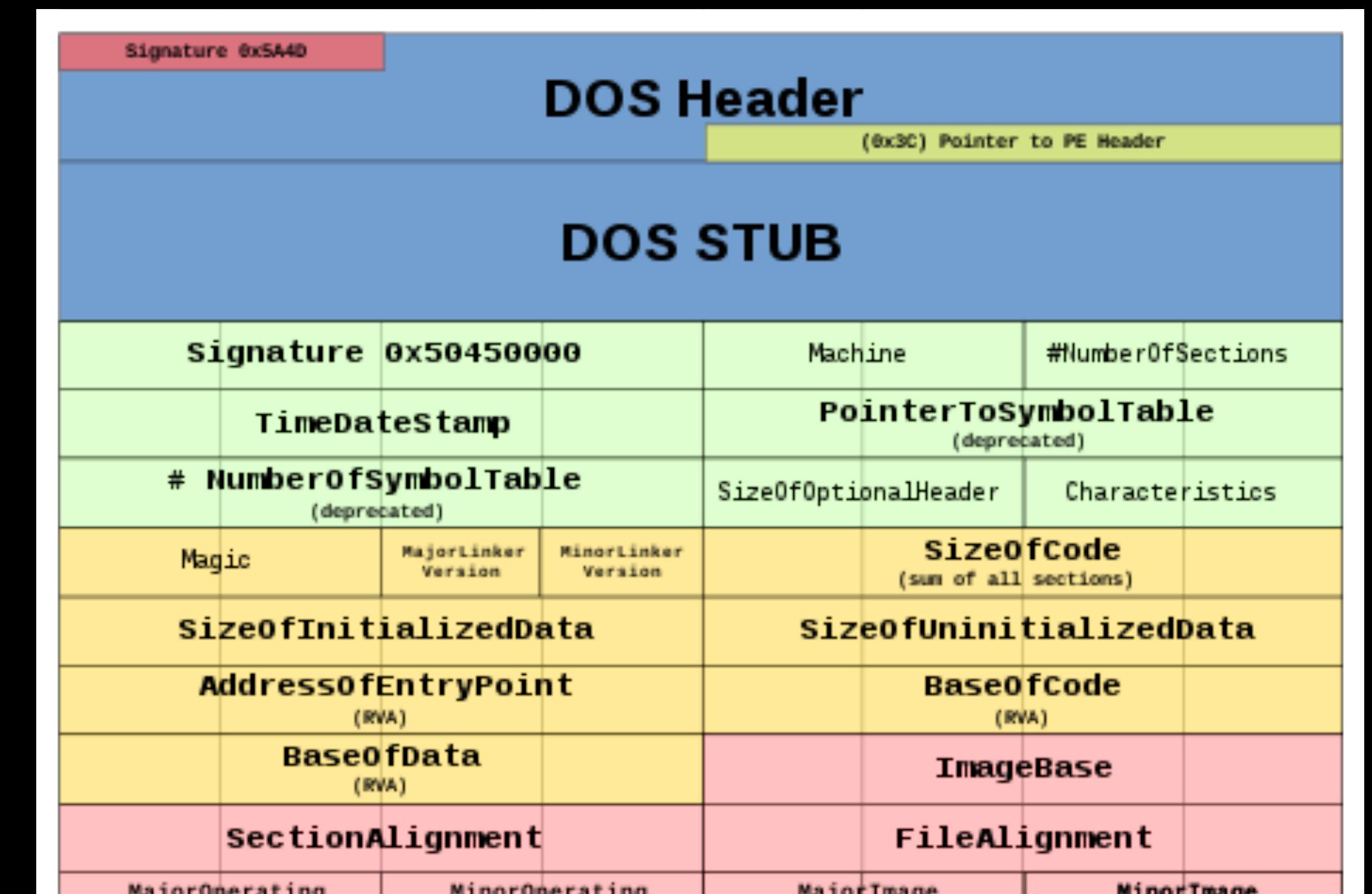
$$s_{i+1} = as_i + b \pmod{r}$$

- 加密： $m_i \oplus s_i = c_i$
- 解密： $c_i \oplus s_i = m_i$

# 基於 LCG 的串流加密的弱點

## 1. 假設已知

- 公開： $c, r$
- 檔案/通訊協定格式開頭： $m_0, m_1, m_2$



EXE 檔案格式，

src: [https://commons.wikimedia.org/wiki/  
File:Portable\\_Executable\\_32\\_bit\\_Structure\\_in\\_SVG\\_fixed.svg](https://commons.wikimedia.org/wiki/File:Portable_Executable_32_bit_Structure_in_SVG_fixed.svg)

# 基於 LCG 的串流加密的弱點

1. 假設已知

- 公開： $c, r$
- 檔案/通訊協定格式開頭： $m_0, m_1, m_2$

2. 計算得到： $s_0, s_1, s_2$

$$s_0 \equiv c_0 \oplus m_0$$

$$s_1 \equiv c_1 \oplus m_1$$

$$s_2 \equiv c_2 \oplus m_2$$

# 基於 LCG 的串流加密的弱點

1. 假設已知

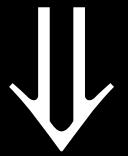
- 公開 :  $c, r$
- 檔案/通訊協定格式開頭 :  $m_0, m_1, m_2$

2. 計算得到 :  $s_0, s_1, s_2$

3. 解方程式，得到  $a, b$

$$s_1 = as_0 + b \pmod{r}$$

$$s_2 = as_1 + b \pmod{r}$$



$$a = (s_1 - s_2)(s_0 - s_1)^{-1} \pmod{r}$$

$$b = s_1 - as_0 \pmod{r}$$

# 基於 LCG 的串流加密的弱點

1. 假設已知

- 公開： $c, r$
- 檔案/通訊協定格式開頭： $m_0, m_1, m_2$

2. 計算得到： $s_0, s_1, s_2$

3. 解聯立方程式，得到  $a, b$

4. 計算  $s$ ，解密出  $m$

$$s_i = as_{i-1} + b \pmod{r}$$

$$m_i = c_i \oplus s_i$$

# Lab: Crack LCG

- 注意 FLAG 格式為 TNFSH{...}
- 提示：善用 FLAG 格式

Crack LCG  
100

閱讀題目程式碼 `chal.py`，然後嘗試從基於 LCG 的串流加密中還原出明文吧！

連線資訊：`nc 210.70.138.222 11004`

Flag 格式：`TNFSH{...}`

Hint : FLAG 格式很有用

Author: lce1187

`chal.py`

# Lab: Crack LCG

- 程式碼邏輯

1.  $r = 2^{16}$

2. 生成  $a, b, \text{seed } s_0$ ，三者都是 2-byte

3. LCG 計算  $s_1, s_2, \dots$ ，也都是 2-byte

4. 將  $s_0, s_1, \dots, s_n$  接起來，變成  $ss$

5.  $c = m \oplus ss$

```
23     r = 2**16
24     a, b = get_a_b(r)
25     seed = random.randint(0, r-1)
26
27     ss = bytearray()
28     s = seed
29     for i in range(len(flag) // 2):
30         ss += s.to_bytes(2, 'big')
31         s = lcg(s, a, b, r)
32
33     enc_flag = []
34     for i in range(len(flag)):
35         enc_flag.append(flag[i] ^ ss[i])
36
37     #print('a, b, seed:', a, b, seed)
38     print(enc_flag)
```

# Lab: Crack LCG

- 觀察
  1.  $a, b$ , seed  $s_0$ ，都是 2-byte
  2. FLAG 格式為 6-byte
- 剛好可以計算  $s_0, s_1, s_2$

$$s_0 = (m_0 \oplus c_0, m_1 \oplus c_1)$$

$$s_1 = (m_2 \oplus c_2, m_3 \oplus c_3)$$

$$s_2 = (m_4 \oplus c_4, m_5 \oplus c_5)$$

# Lab: Crack LCG

- 計算  $s_0, s_1, s_2$

$$s_0 = (m_0 \oplus c_0, m_1 \oplus c_1)$$

$$s_1 = (m_2 \oplus c_2, m_3 \oplus c_3)$$

$$s_2 = (m_4 \oplus c_4, m_5 \oplus c_5)$$

```
flag = b'TNFSH{'
enc_flag = [91, 89, 44, 219, 54, 102, 91, 133, 227, 108, 175, 161,
r = 2**16

s = [None, None, None]
for i in range(3):
    s[i] = [flag[i*2] ^ enc_flag[i*2], flag[i*2+1] ^ enc_flag[i*2+1]]
    s[i] = bytes(s[i])

print(hex(int.from_bytes(s[0], 'big')))
print(hex(int.from_bytes(s[1], 'big'))))
print(hex(int.from_bytes(s[2], 'big'))))
```

```
› python3 solve2.py
0xf17
0x6a88
0x7e1d
```

# Lab: Crack LCG

- 計算  $s_0, s_1, s_2$
- 解方程式得到  $a, b$ 
  - 需要算出  $(s_0 - s_1)$  的模運算  
乘法反元素  $(s_0 - s_1)^{-1}$

$$s_1 = as_0 + b \pmod{r}$$

$$s_2 = as_1 + b \pmod{r}$$

↓

$$a = (s_1 - s_2)(s_0 - s_1)^{-1} \pmod{r}$$

$$b = s_1 - as_0 \pmod{r}$$

# Lab: Crack LCG

- 計算  $s_0, s_1, s_2$
- 解方程式得到  $a, b$ 
  - 需要算出  $(s_0 - s_1)$  的模運算  
乘法反元素  $(s_0 - s_1)^{-1}$

$$ax \equiv 1 \pmod{r}$$

```
s[0] = int.from_bytes(s[0], 'big')
s[1] = int.from_bytes(s[1], 'big')
s[2] = int.from_bytes(s[2], 'big')
for x in range(1, r):
    if (x*(s[0] - s[1])) % r == 1:
        inv = x
        print('inv found:', inv)
        break
```

```
› python3 solve2.py
inv found: 64111
```

# Lab: Crack LCG

- 計算  $s_0, s_1, s_2$
  - 解方程式得到  $a, b$ 
    - 需要算出  $(s_0 - s_1)$  的模運算  
乘法反元素  $(s_0 - s_1)^{-1}$
    - 計算  $a, b$
- $$s_1 = as_0 + b \pmod{r}$$
- $$s_2 = as_1 + b \pmod{r}$$
- $$\downarrow$$
- $$a = (s_1 - s_2)(s_0 - s_1)^{-1} \pmod{r}$$
- $$b = s_1 - as_0 \pmod{r}$$

# Lab: Crack LCG

- 計算  $s_0, s_1, s_2$
- 解方程式得到  $a, b$
- 需要算出  $(s_0 - s_1)$  的模運算  
乘法反元素  $(s_0 - s_1)^{-1}$
- 計算  $a, b$

```
a = ((s[1] - s[2]) * inv) % r
b = (s[1] - a*s[0]) % r
print('a, b:', a, b)
```

```
> python3 solve2.py
inv found: 64111
a, b: 101 30325
```

# Lab: Crack LCG

- 計算  $s_0, s_1, s_2$
- 解方程式得到  $a, b$ 
  - 需要算出  $(s_0 - s_1)$  的模運算乘法反元素  $(s_0 - s_1)^{-1}$
- 計算  $a, b$
- 計算  $s_0, s_1, \dots, s_n$
- 解密  $m_i = c_i \oplus s_i$

```
# compute s[i]
for i in range(3, len(enc_flag) // 2):
    s_i = lcg(s[i-1], a, b, r)
    s.append(s_i)

# turn s[i] from 2-byte to 1-byte
ss = bytearray()
for s_i in s:
    ss += s_i.to_bytes(2, 'big')

# decrypt flag
flag = ''
for i in range(len(enc_flag)):
    flag += chr(enc_flag[i] ^ ss[i])
print(flag)
```

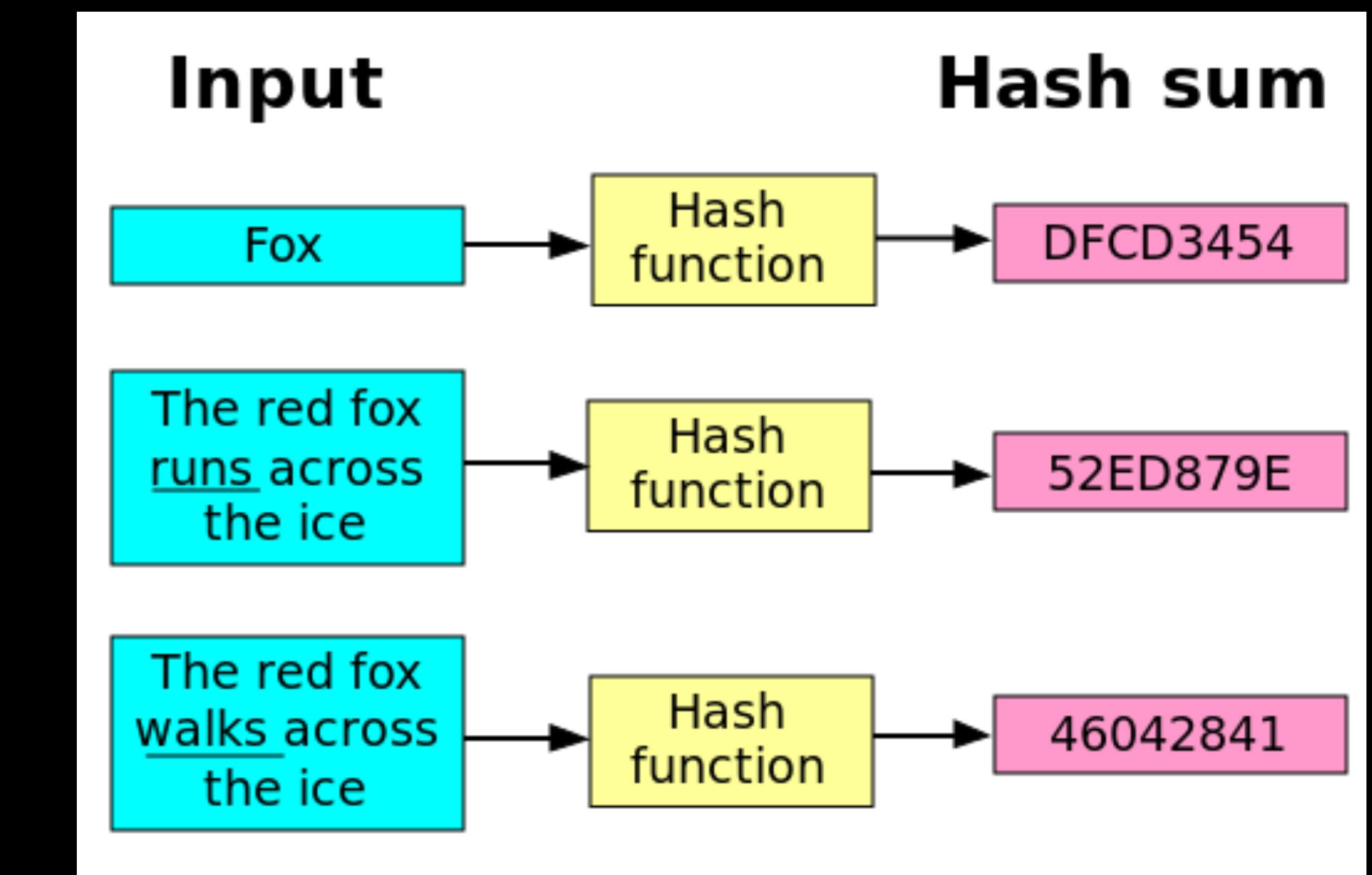
休息一下~

# 雜湊 Hash

# 雜湊函數

## Hash Function

- 將資料經過計算，生成固定長度數值的單向函數
- 此數值稱為該資料的雜湊值 (hash)
- 實際應用
  - 資料庫索引
  - 快取
  - 錯誤檢測



# 密碼學雜湊函數

## Cryptographic Hash Function

- 專門用於密碼學領域的雜湊函數
- 特性
  - 固定長度：輸入任意長度的資料，輸出皆為固定長度
  - 不可逆性：難以從輸出回推輸入
  - 防碰撞性：難以找到兩個不同的輸入，具有相同的輸出
  - 雪崩效應：改變輸入的任一 bit，都會使輸出有巨大的改變

# 密碼學雜湊函數

## Cryptographic Hash Function

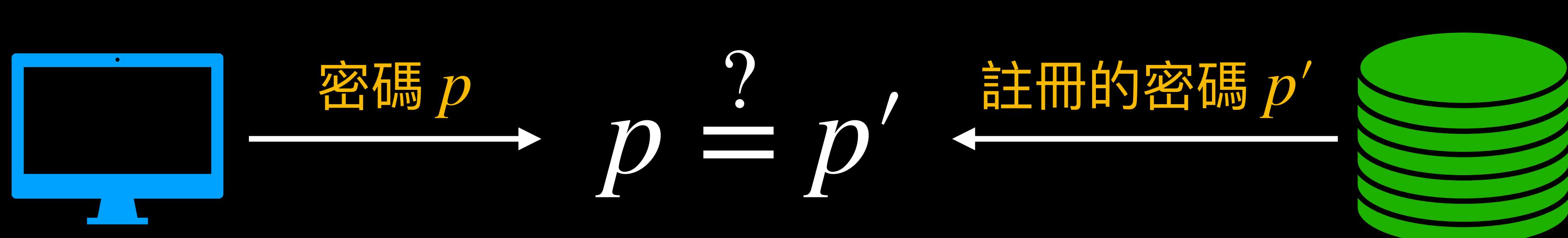
- 非常多領域的重要元件
  - 密碼學安全偽隨機數生成器
  - 儲存密碼
  - 身份驗證
  - 區塊鏈
  - ...

# 討論：如何安全地儲存密碼？

- 同服務器怎麼儲存你的密碼？

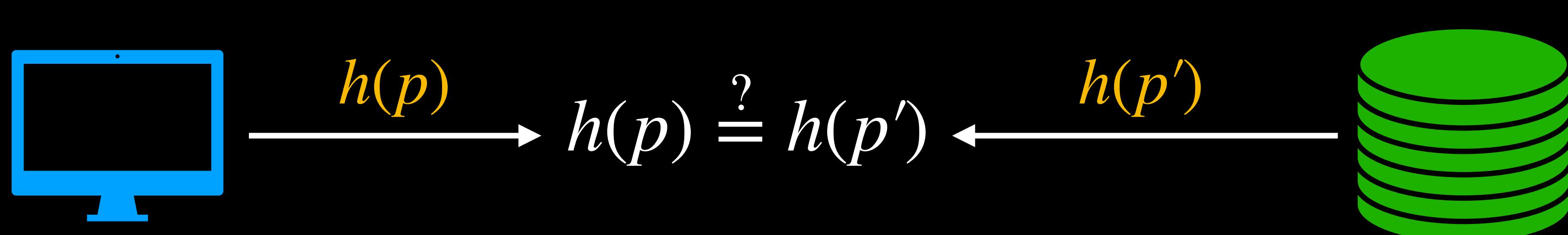
# 討論：如何安全地儲存密碼？

- 同服務器怎麼儲存你的密碼？
  - 直接儲存註冊時輸入的密碼
    - 該公司管理員知道你的密碼，好像不太安全 😱



# 討論：如何安全地儲存密碼？

- 同服務器怎麼儲存你的密碼？
  - ~~直接儲存註冊時輸入的密碼~~
  - 儲存密碼的 hash
    - 不必傳送、儲存密碼，洩漏時風險也相對較小



# 常見的密碼學雜湊演算法

- md5 (已被破解)
- sha1 (已被破解)
- sha256
- sha512

新聞

## 史上第一例！Google破解SHA1實現碰撞攻擊

Google與荷蘭數學暨電腦科學研究機構CWI Amsterdam共同宣佈，費時兩年時間研究終於破解SHA1，讓兩個不同的文件使用相同的訊息摘要，使實現碰撞攻擊的可能性提昇。

文/ 陳曉莉 | 2017-02-24 發表

13 譴 13 分享

Expected behavior: different hashes

Collision attack: same hashes

Doc 1 Sha-1 42C1..21

Doc 2 Sha-1 3E2A..AE

Good doc Sha-1 3713..42

Bad doc Sha-1 3713..42

圖片來源: Google

左側通過SHA-1後不同文件產生不同的Hash，破解後相同的Hash可用在惡意文件上。

<https://www.ithome.com.tw/news/112347>

# 雜湊破解方法

- 破解雜湊演算法本身非常困難
- 依靠建表、查表的方式來回推輸入，又稱作彩虹表 (rainbow table)



# Lab: Crack The Hash

## Crack The Hash

100

找到雜湊值「7c222fb2927d828af22f592134e8932480637c0d」的輸入為何  
Hint：網路上有許多提供雜湊彩虹表的英文網站

Author: Ice1187

# Lab: Crack The Hash

## 1. 根據長度推測雜湊演算法

- 20 bytes => 160 bits => sha1

## 2. 查詢線上彩虹表，若找不到可以多試幾個網站

The screenshot shows a web-based hash cracking tool. At the top, a text input field contains the hex digest "7c222fb2927d828af22f592134e8932480637c0d". To the right of the input is a button labeled "Crack Hashes". Below the input field, a section titled "Supports:" lists various hashing algorithms: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+(sha1(sha1\_bin)), QubesV3.1BackupDefaults. A table below displays the cracked result:

Hash	Type	Result
7c222fb2927d828af22f592134e8932480637c0d	sha1	12345678

<https://crackstation.net/>

# 編碼、加密、雜湊的差異

# 編碼

## Encoding

- 將資訊以特定格式重新進行表達
- 知道編碼方式即可解碼，完全沒有保密功能
- 常見：ASCII、Base64

- $65 = 0x41 = A$

二進位	十進位	十六進位	圖形	二進位	十進位	十六進位	圖形	二進位	十進位	十六進位	圖形
0010 0000	32	20	(space)	0100 0000	64	40	@	0110 0000	96	60	'
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	\$	0100 0100	68	44	D	0110 0100	100	64	d

ASCII 表，<https://zh.wikipedia.org/zh-tw/ASCII>

# 編碼、加密、雜湊的差異

- 三者是完全不同的東西，功能、用途都不一樣

	編碼	加密	雜湊
目的	轉換資訊的表達形式	使不知道密鑰的人無法得知資訊內容	得到資訊在雜湊函數下的獨特輸出
機密性	X	O	O
可逆性	O	O	X
逆推 所需資訊	編碼方式	加密演算法、密鑰	//

# 編碼 ≠ 加密

## 檢舉人代號疑似使用 Base64 編碼

有網友在 PTT 論壇上發文指出，臺中市警察局公布的檢舉人代號，看起來很像是 Base64 編碼。Base64 編碼是一種將二進位資料編碼成可列印字元字串的標準方法，Base64 編碼可以用來傳輸二進位資料，例如圖片、音訊或影片。

該網友將公佈的檢舉人代號 Base64 編碼進行解碼，得到的結果與身分證號碼的格式相符，因此該網友認為臺中市警察局可能使用 Base64 編碼來自以為「加密」檢舉人代號。

## Base64 編碼並非加密 個人資料遭洩露風險

如果檢舉人代號確實是使用 Base64 編碼，那麼這可能會對檢舉人隱私造成一定的風險。因為 Base64 編碼並非加密，而是一種將二進位資料編碼成可列印字元字串的方法。因此，如果有人掌握了 Base64 解碼的方法，就可以輕鬆解碼檢舉人代號，並獲取檢舉人的個人資料。

而因為這次檢舉人資料是被公開在新聞上，也有可能造成檢舉人身份曝光後遭被檢舉人尋仇報復的風險。

# 參考資料

## Reference

- Understanding Cryptography: A Textbook for Students and Practitioners
- HITCON CMT 2018: 應用密碼學入門