記得簽到！

https://forms.gle/xFz8YY5Q2DEqu5es7

應用密碼學與 CTF 解題實務

AIS3 好厲駭 2025/05/03

Speaker :李安傑

WhoAmI

AnJ

- NTU CSIE
- NSLab
- Balsn CTF
- 計算機安全、密碼學與資訊安全 助教

Outline

1. Intro
2. 對稱式密碼學 (AES)
3. 非對稱式密碼學 (RSA, Dlog)
4. 其他密碼學構造 (Hash, Digital Signature, ZKP)
5. 進階(?)內容 (Elliptic Curve, Lattice)

Intro

	加密（Encrypt）	編碼（Encode）	雜湊 （Hash）
還原的操作	解密（Decrypt）	解碼（Decode）	✗
還原需要什麼	知道規則 並且知道密鑰	知道規則	理論上 不能還原
例子	AES, RSA, ElGamal	Base64	MD5, sha256

Kerckhoffs's Principle

- A cryptosystem should be secure even if the attacker knows all details about the system, except for the **secret key**.

Symmetric vs Asymmetric Encryption

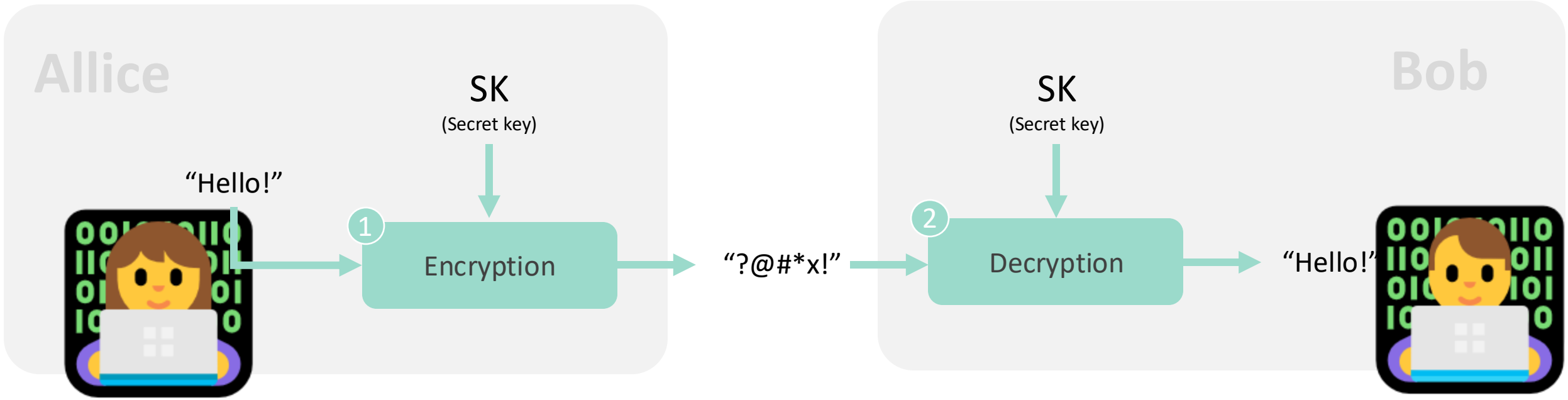
	Symmetric	Asymmetric (Public key)
Feature	速度快、搭配各種 operation mode 可以加密較長的內容	不需要預先的 shared secret、 Public key 有不可否認信
例子	AES, DES, ChaCha20-Poly1305	RSA, ElGamal

Useful tools

- PyCryptodome
  - pip install pycryptodome
- Sage
  - apt-get install sagemath
  - Online version: https://sagecell.sagemath.org/ or https://cocalc.com/
  - 基本上跟 python 語法一樣，只是要注意「^^ : xor, ^ : power」

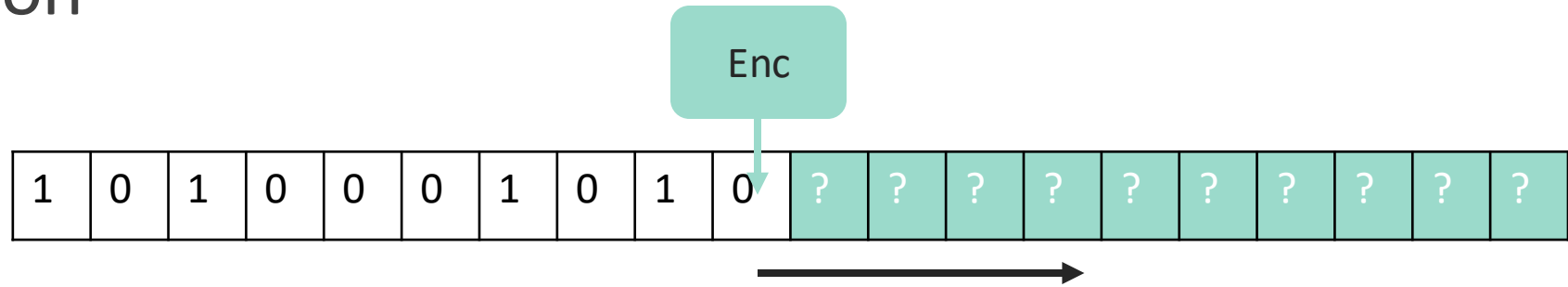
Symmetric

Symmetric Encryption

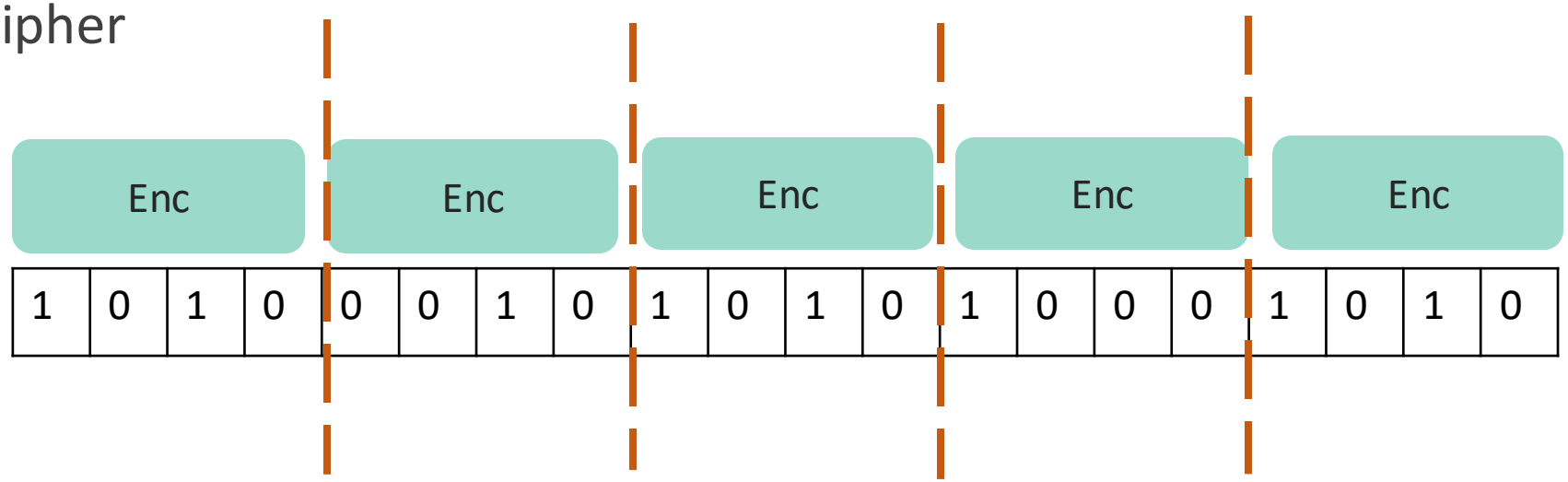


Symmetric Encryption

- Stream Cipher



- Block Cipher



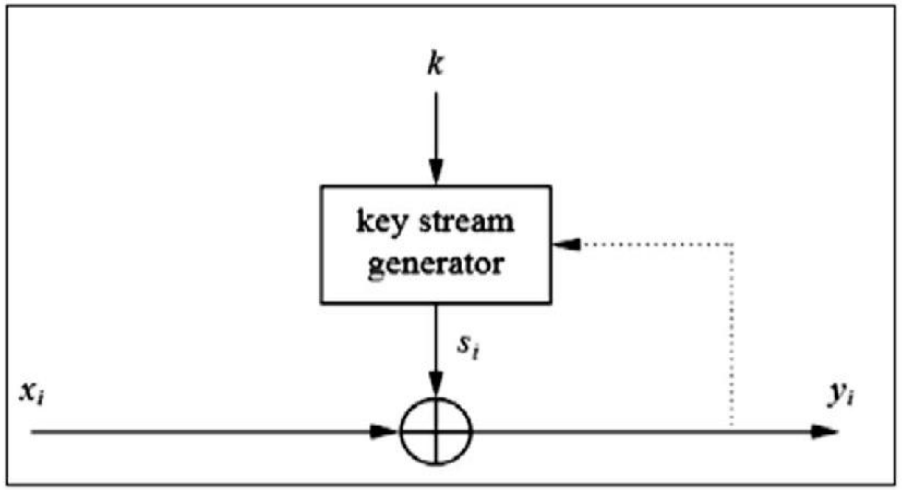
Symmetric

Stream Cipher

- Stream Cipher
- PRNG (Pseudo Random Number Generator)

Stream Cipher

- Encryption: $y_i = e_{si}(x_i) = x_i \oplus s_i$
- Decryption: $x_i = d_{si}(y_i) = y_i \oplus s_i$
- Random & reproducible keystream
  - Security depends entirely on the keystream
  - Reused Key Attack



Symmetric

PRNG (Pseudo Random Number Generator)

- Given an initial seed, generate a sequence of random numbers.
- **通常會有一些**內部的狀態跟週期（**週期長短通常跟狀態的位元數有關**）
- **常見的 PRNG 作法：**Linear Congruential, MT19937, LFSR

PRNG

1. PRNG can be predicted with enough output bits
2. Combine PRNGs with a nonlinear operation?

   => Correlation Attack

   e.g. 有三個 PRNG 的 output a,b,c; output = a if b else c

   -> output 有75%的機率等於 a

   -> output 有75%的機率等於 c

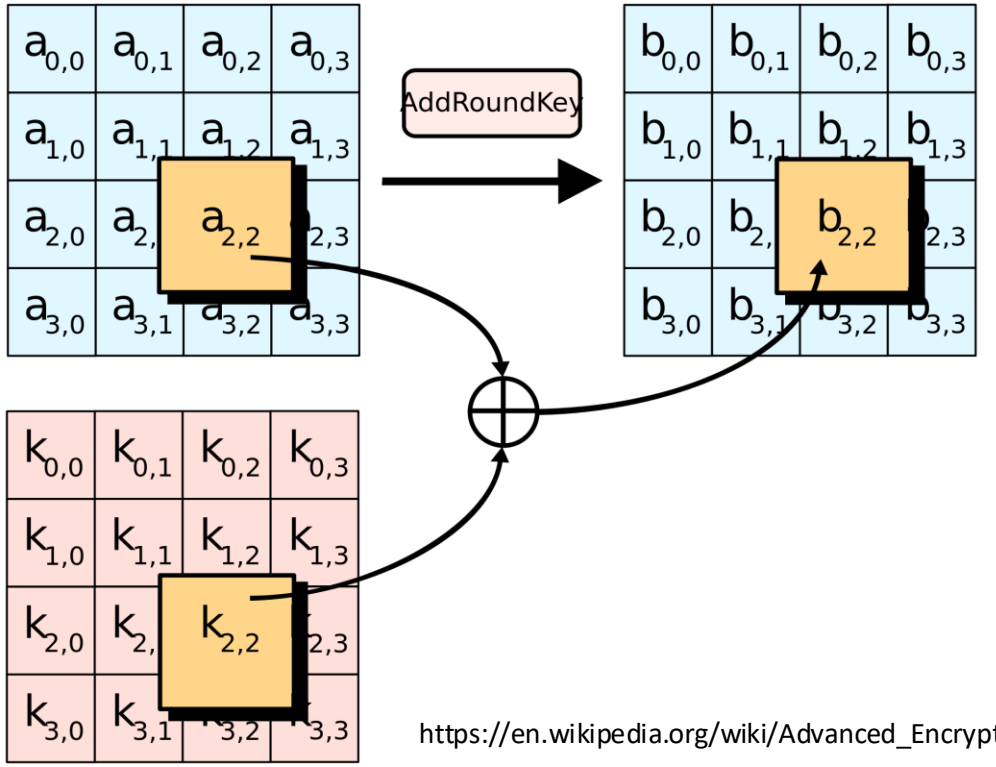
Symmetric

Block Cipher

- AES (Advanced Encryption Standard)
- Mode of operation
  - EBC
  - CBC
  - CTR
  - GCM

Block Cipher

- AES (Advanced Encryption Standard)
  - **大概是最常用的對稱式加密**
  - 內部有四個步驟：
    1. AddRoundKey



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

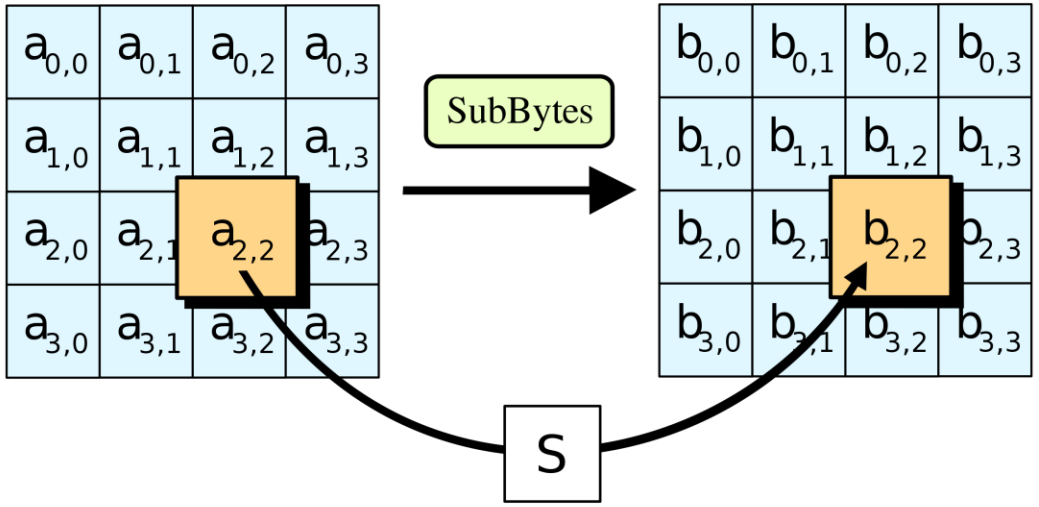
Block Cipher

- AES (Advanced Encryption Standard)
  - **大概是最常用的對稱式加密**
  - 內部有四個步驟：
    1. AddRoundKey
    2. SubBytes

AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

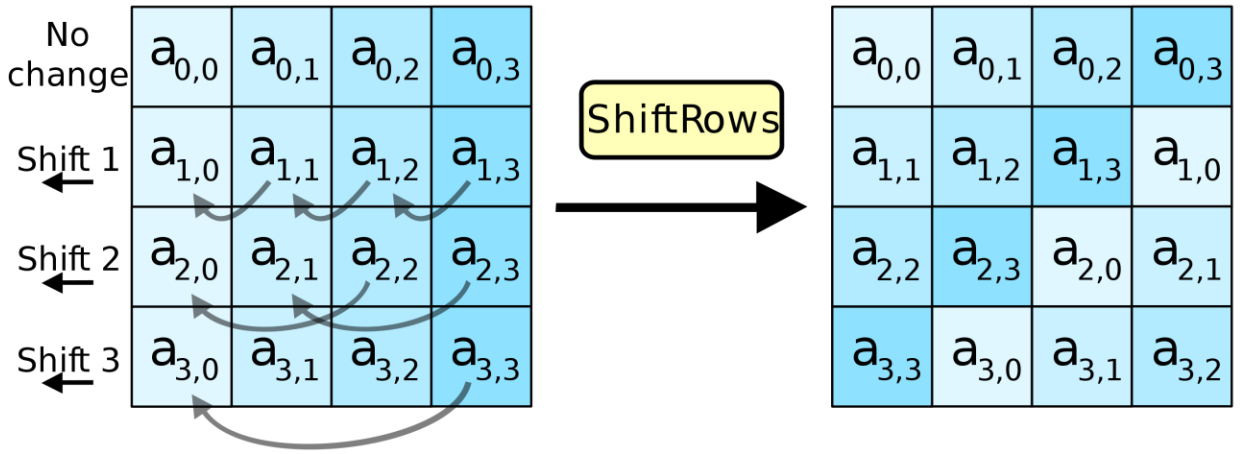
The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value $9a_{16}$ is converted into $b8_{16}$.



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

Block Cipher

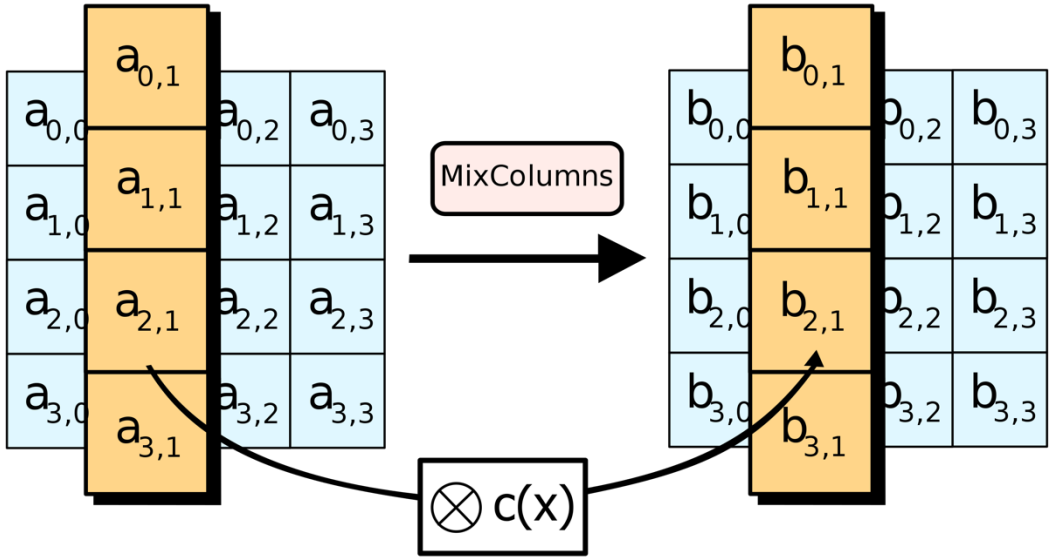
- AES (Advanced Encryption Standard)
  - **大概是最常用的對稱式加密**
  - 內部有四個步驟：
    1. AddRoundKey
    2. SubBytes
    3. ShiftRows



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

Block Cipher

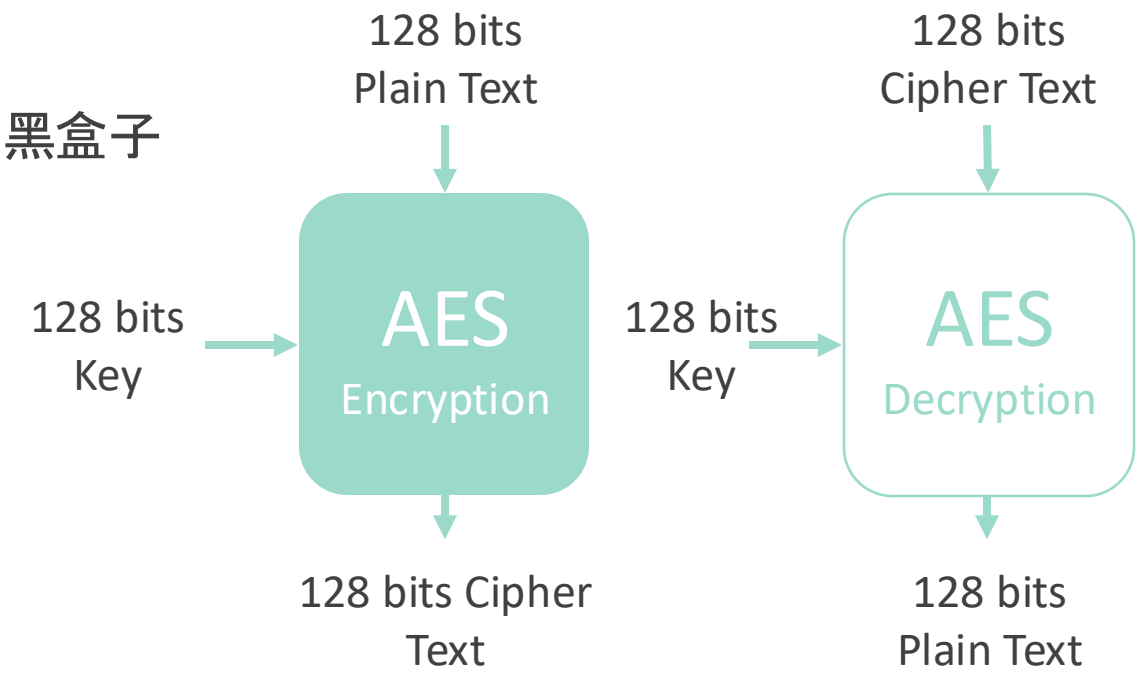
- AES (Advanced Encryption Standard)
  - **大概是最常用的對稱式加密**
  - 內部有四個步驟：
    1. AddRoundKey
    2. SubBytes
    3. ShiftRows
    4. MixColumns



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

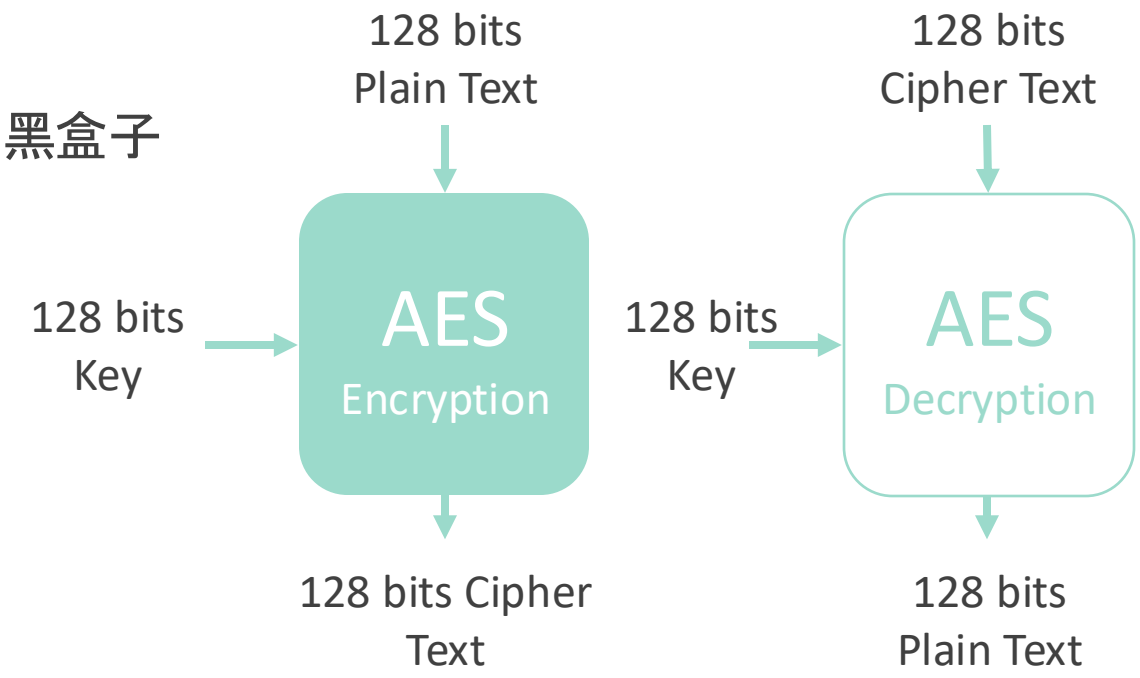
Block Cipher

- AES (Advanced Encryption Standard)
  - 大概是最常用的對稱式加密
  - 因為很安全所以通常在 CTF 裡面都把它當作黑盒子



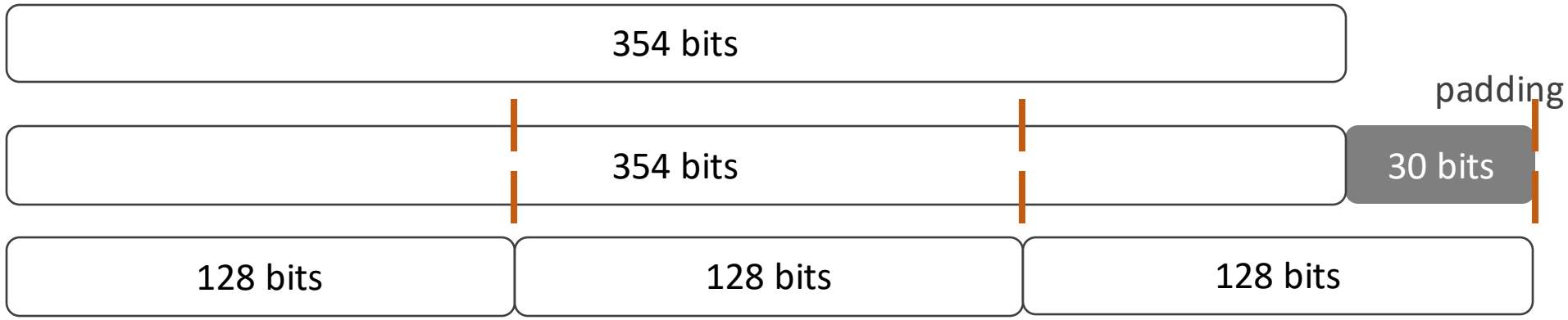
Block Cipher

- AES (Advanced Encryption Standard)
  - 大概是最常用的對稱式加密
  - 因為很安全所以通常在 CTF 裡面都把它當作黑盒子
  - Side-Channel Attack?



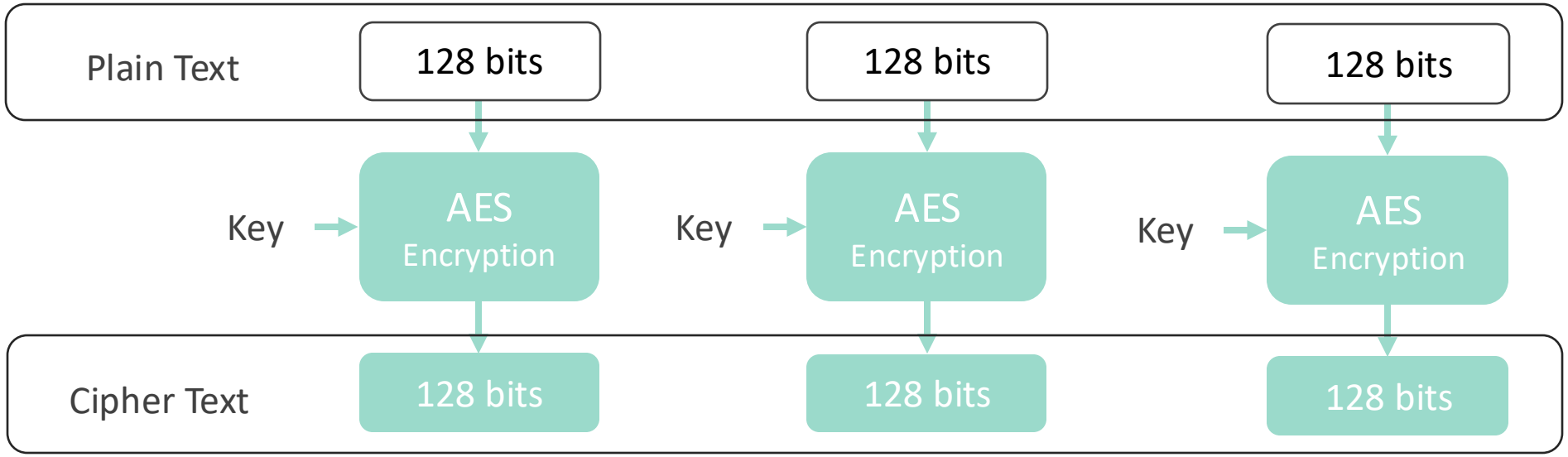
Block Cipher – mode of operation

- 怎麼加密任意長度呢？



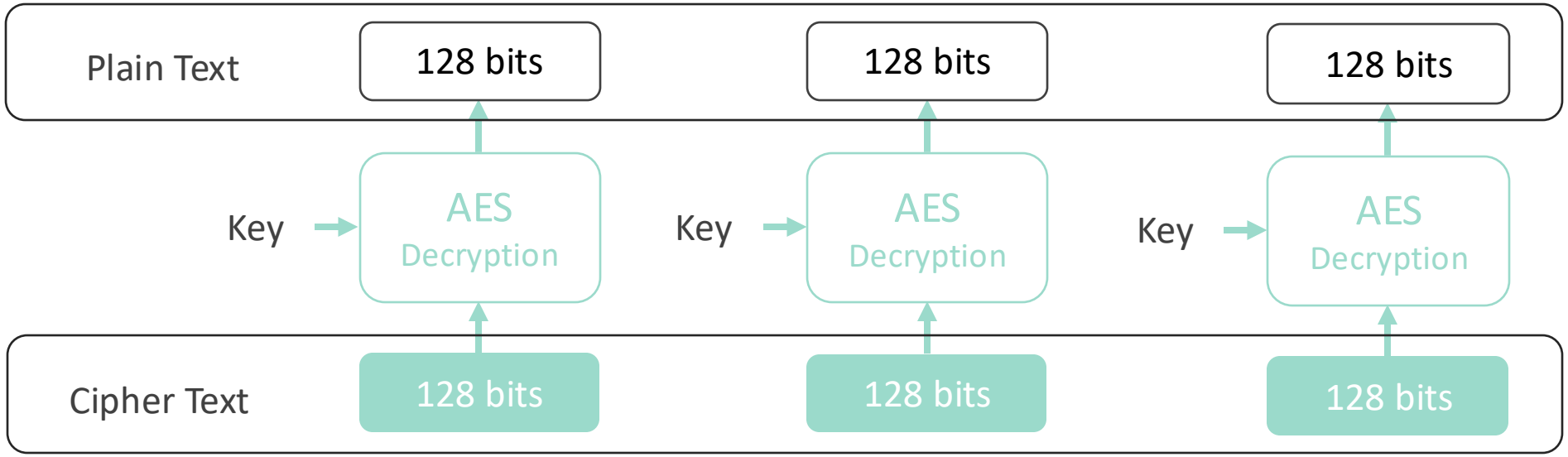
Block Cipher – mode of operation

- ECB mode



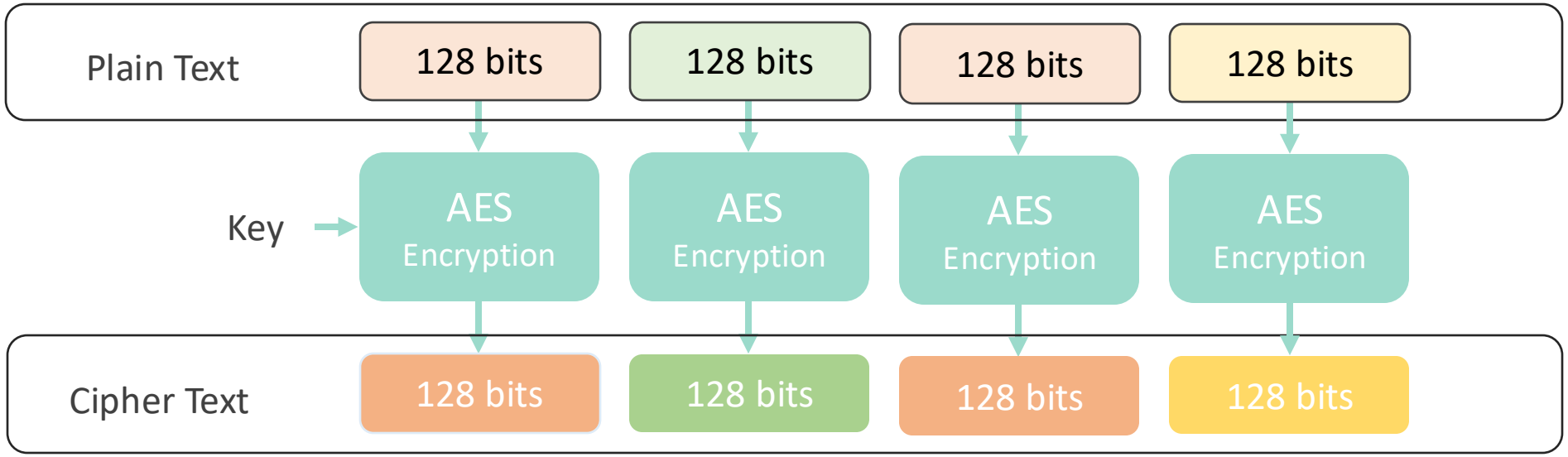
Block Cipher – mode of operation

- ECB mode



Block Cipher – mode of operation

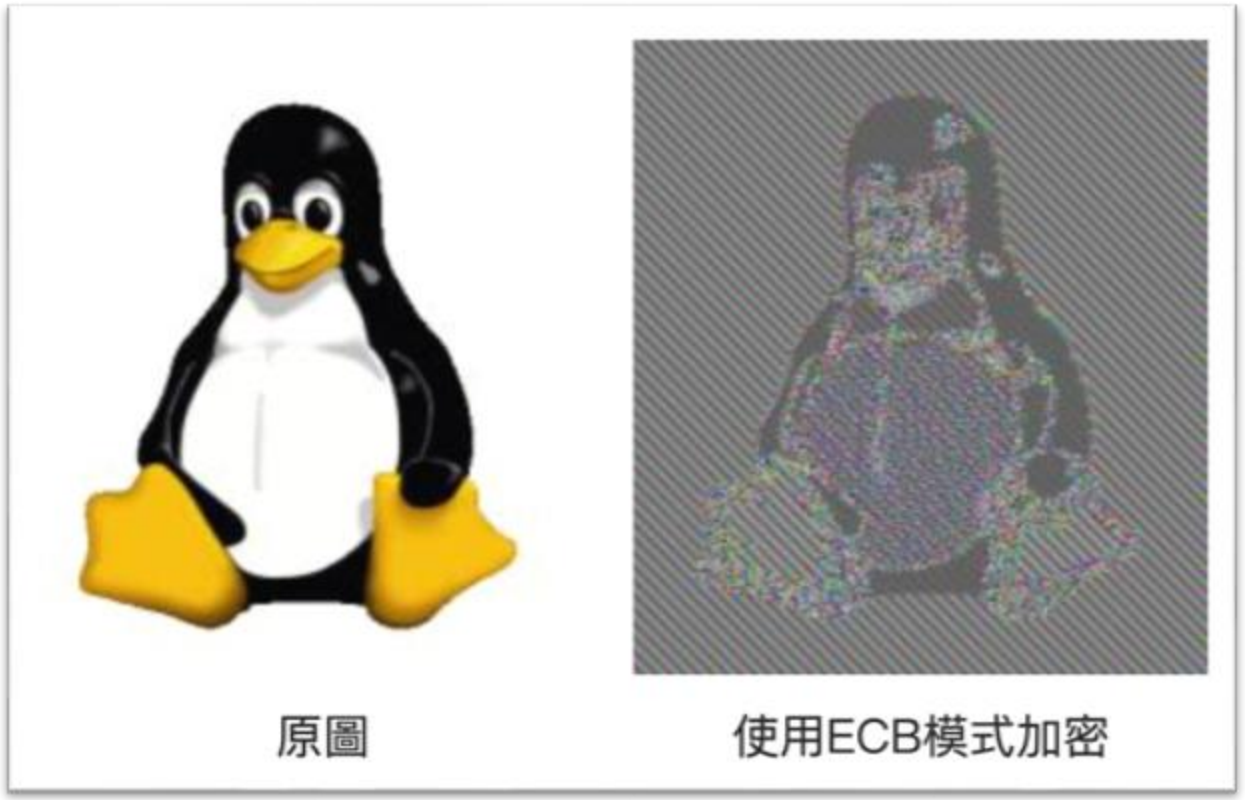
- ECB mode



❌ 一樣的 Block 會加密出一模一樣的密文

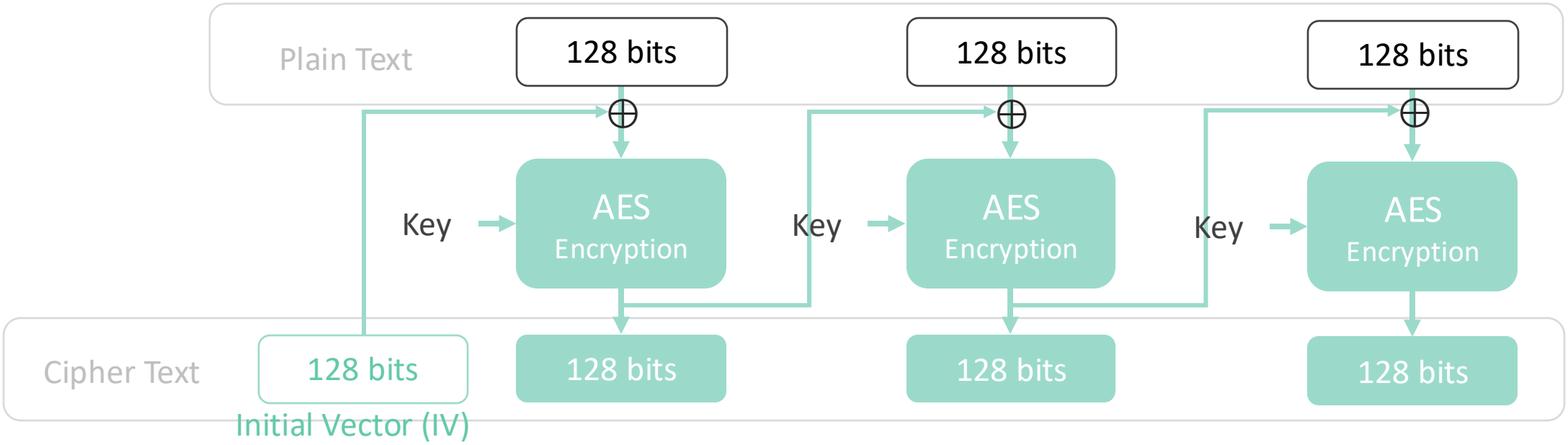
Block Cipher – mode of operation

- ECB mode



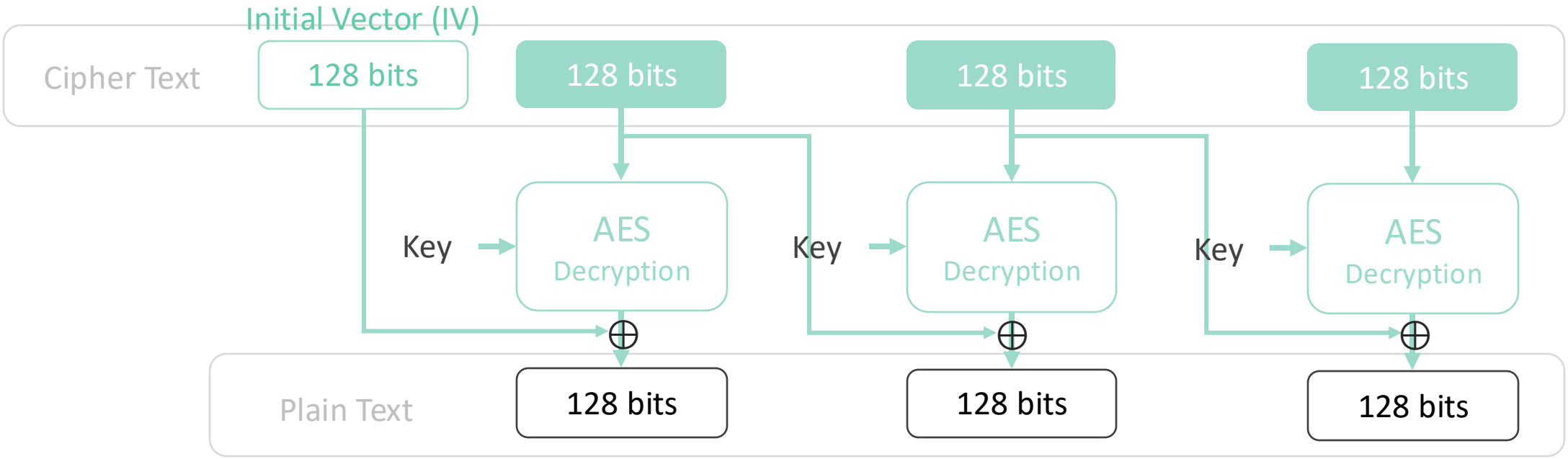
Block Cipher – mode of operation

- CBC mode



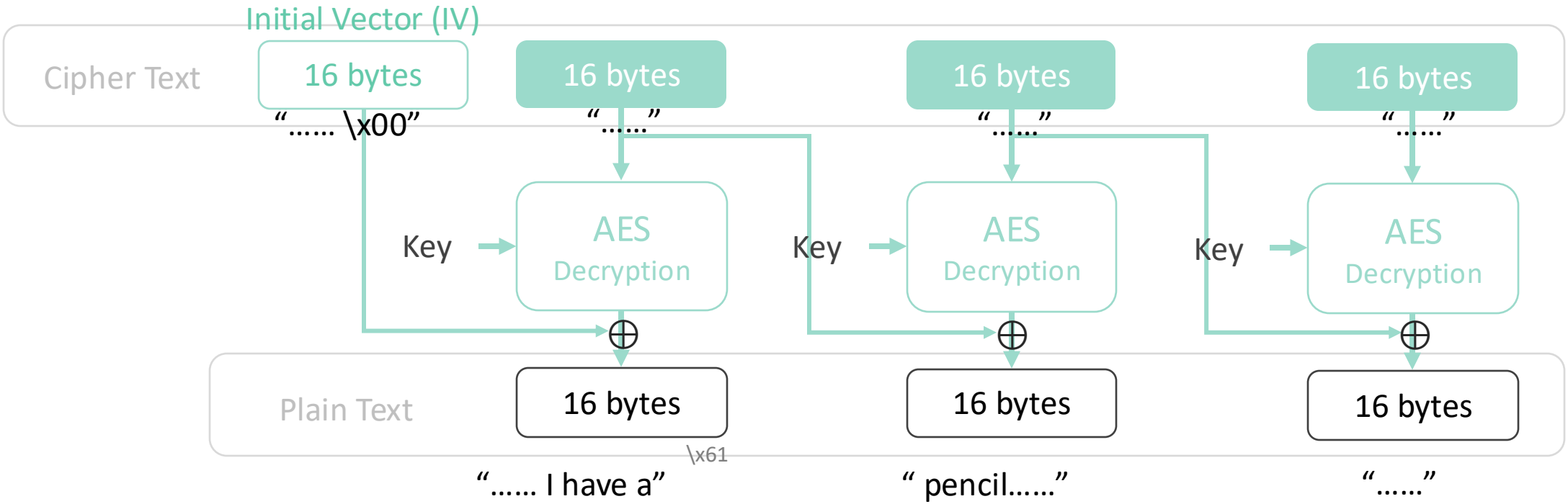
Block Cipher – mode of operation

- CBC mode



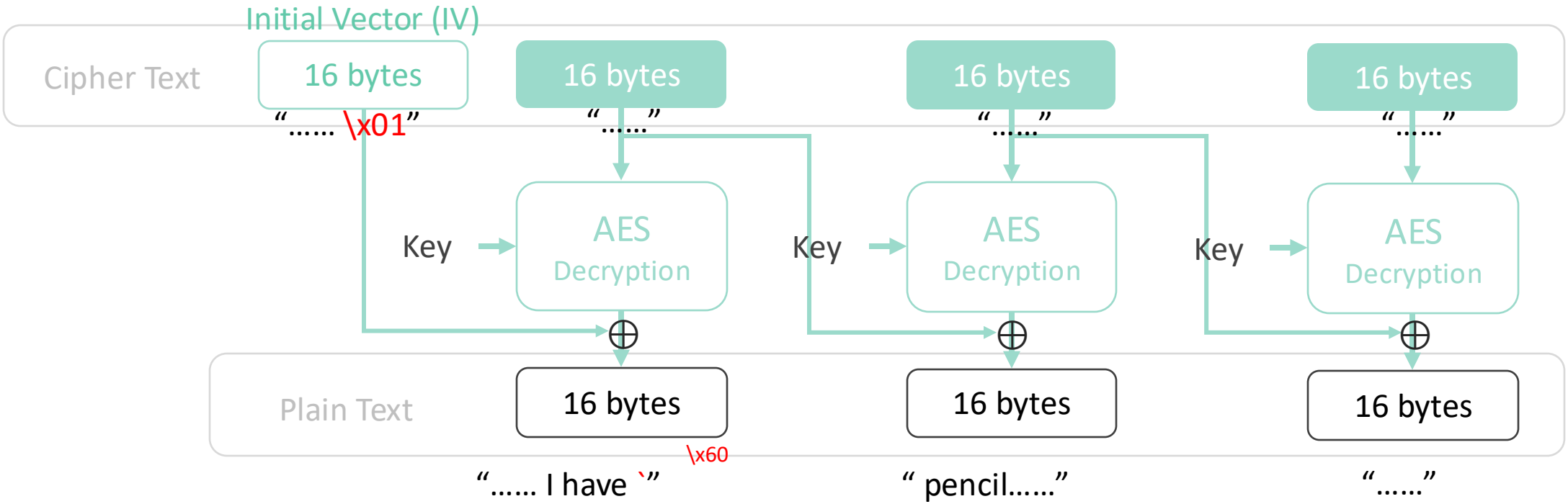
Block Cipher – mode of operation

- CBC mode



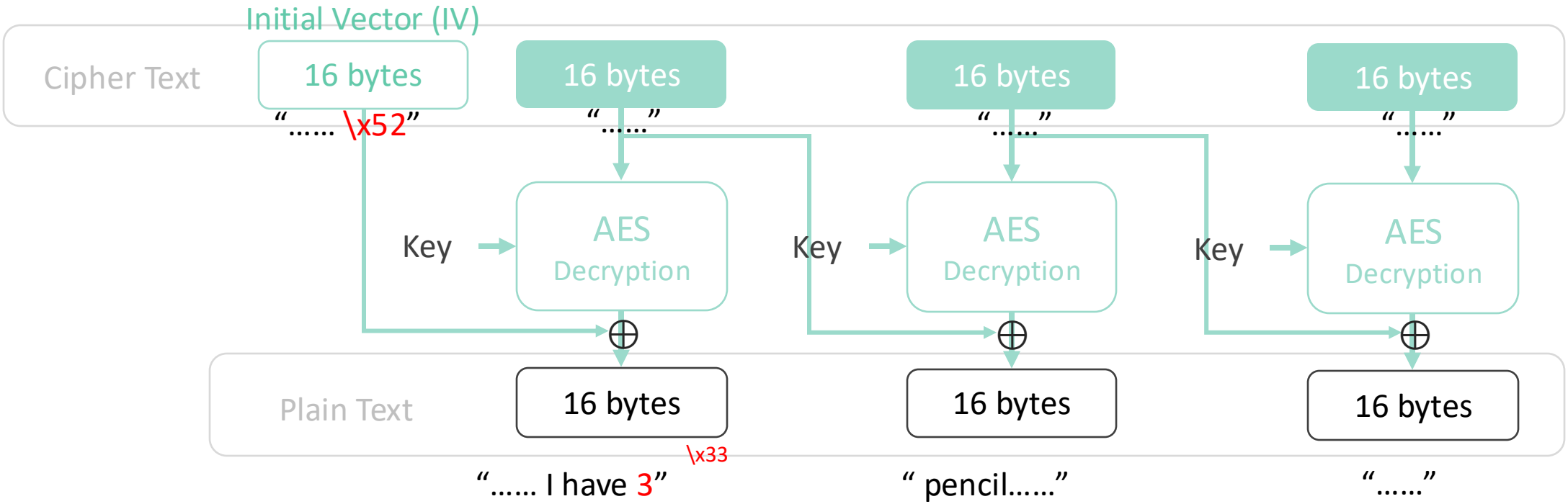
Block Cipher – mode of operation

- CBC mode



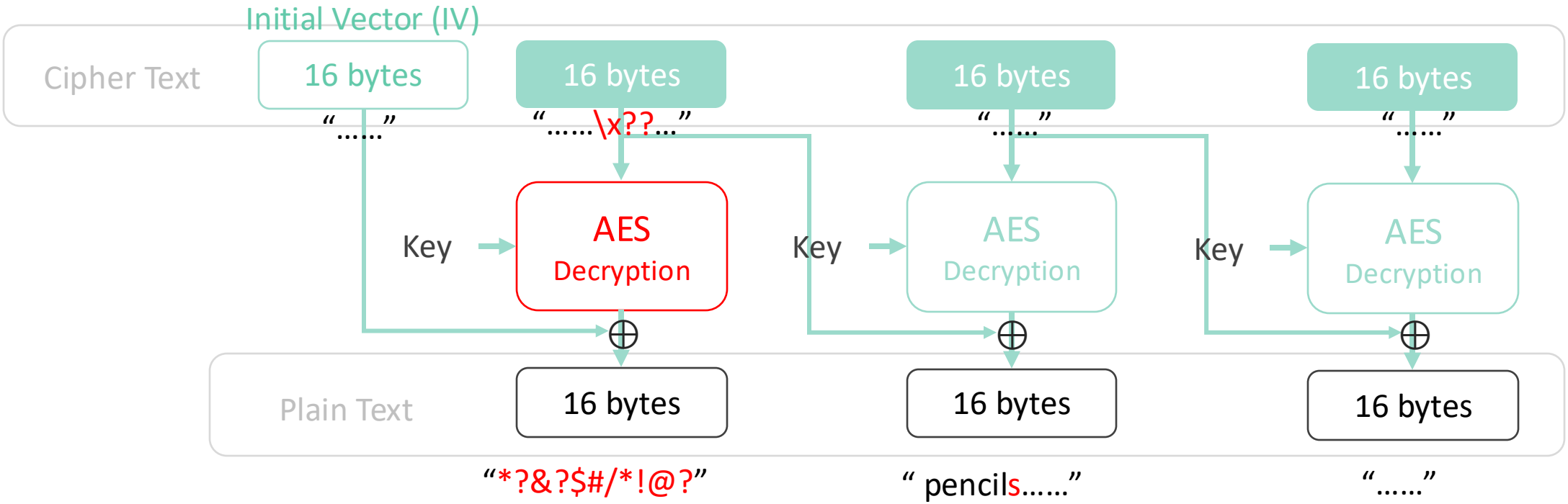
Block Cipher – mode of operation

- CBC mode



Block Cipher – mode of operation

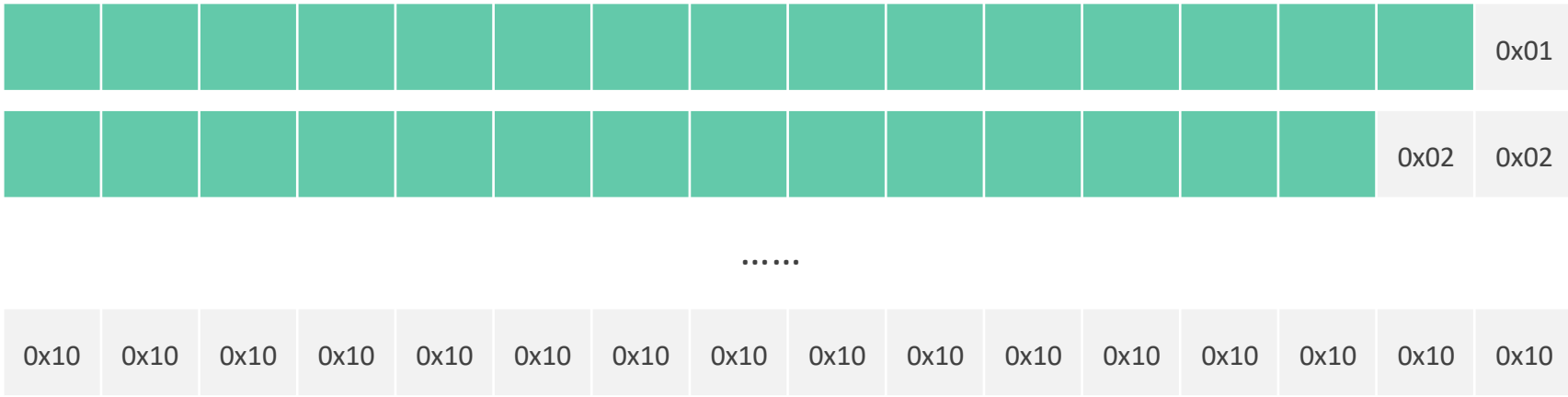
- CBC mode



Block Cipher – mode of operation

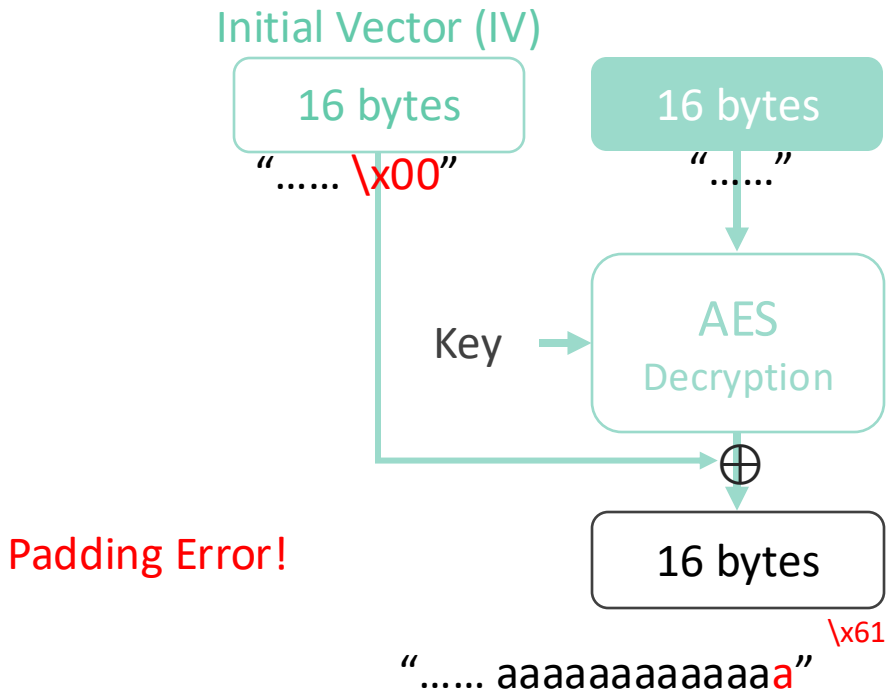
- CBC mode – Padding Oracle Attack
  - 由於 Padding 通常會有一定的規則 (e.g., PKCS#7 Padding)

PKCS#7 Padding (以 bytes 以為單位，少幾個就補多少)



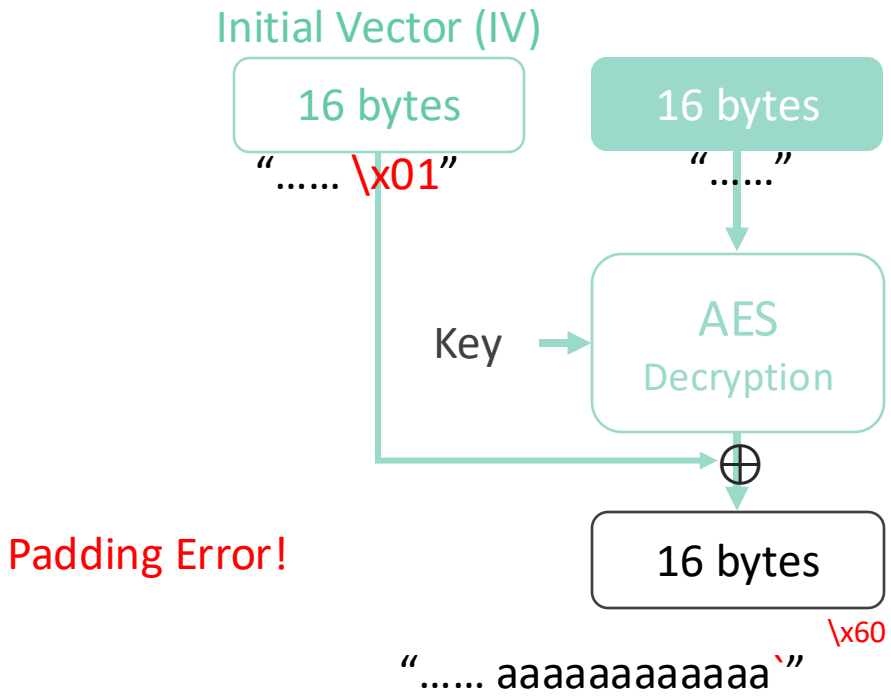
Block Cipher – mode of operation

- CBC mode – Padding Oracle Attack
  - 由於 Padding 通常會有一定的規則 (e.g., PKCS#7 Padding)
  - 可以透過 Server 有沒有觸發 Error 來找出最後一個 Byte



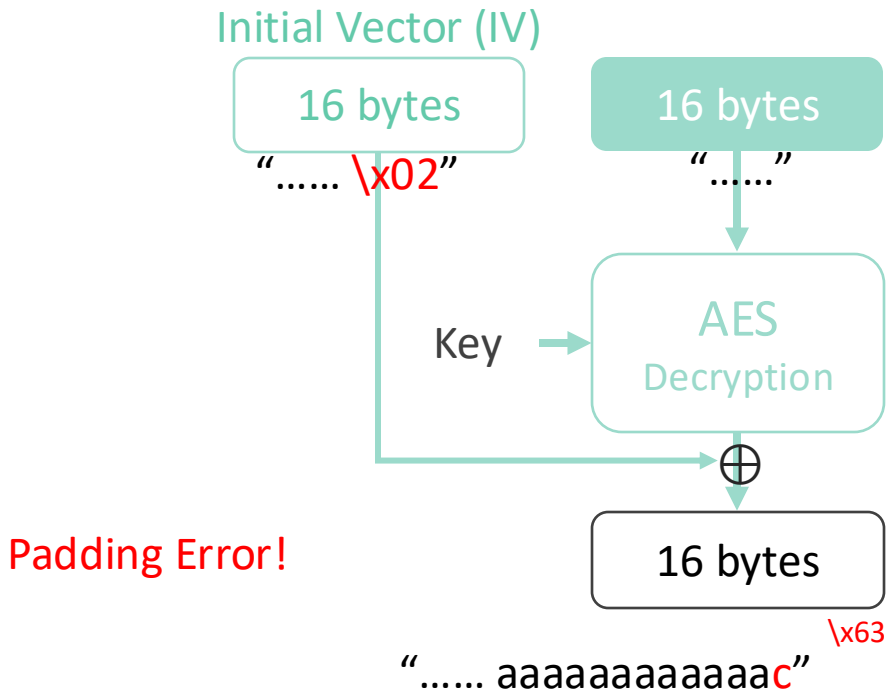
Block Cipher – mode of operation

- CBC mode – Padding Oracle Attack
  - 由於 Padding 通常會有一定的規則 (e.g., PKCS#7 Padding)
  - 可以透過 Server 有沒有觸發 Error 來找出最後一個 Byte



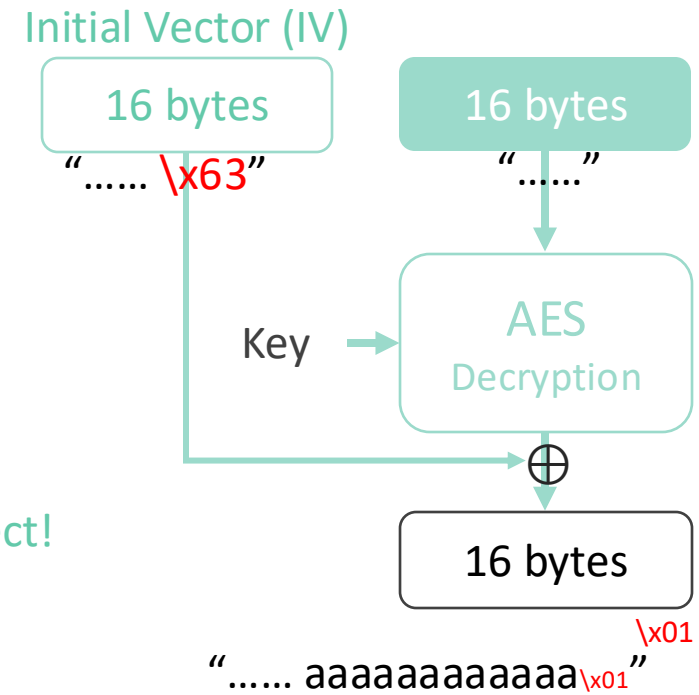
Block Cipher – mode of operation

- CBC mode – Padding Oracle Attack
  - 由於 Padding 通常會有一定的規則 (e.g., PKCS#7 Padding)
  - 可以透過 Server 有沒有觸發 Error 來找出最後一個 Byte



Block Cipher – mode of operation

- CBC mode – Padding Oracle Attack
  - 由於 Padding 通常會有一定的規則 (e.g., PKCS#7 Padding)
  - 可以透過 Server 有沒有觸發 Error 來找出最後一個 Byte

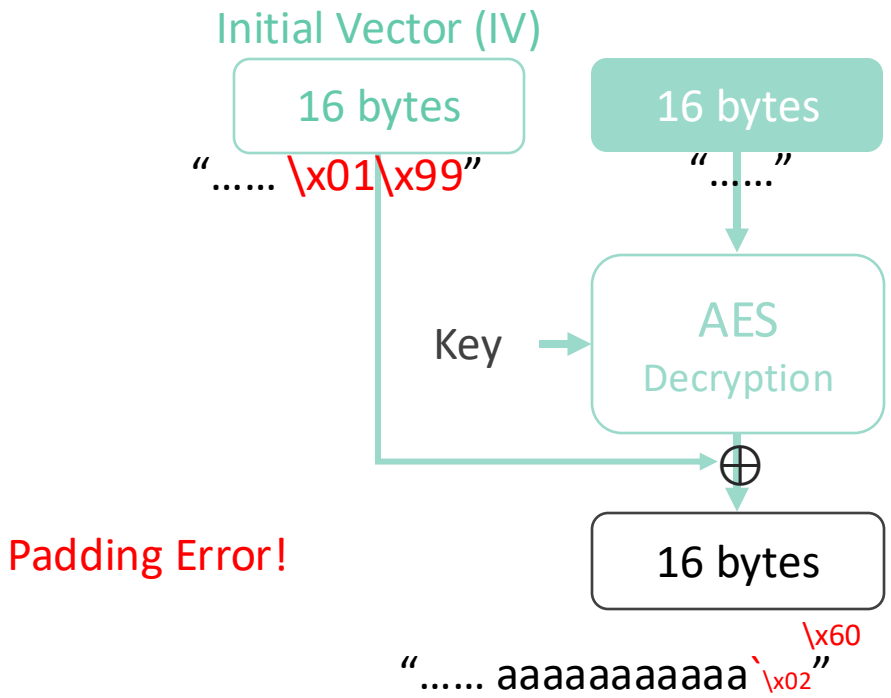


Padding Correct!

The last byte is 96 xor 1 = 97 -> last byte is 'a'

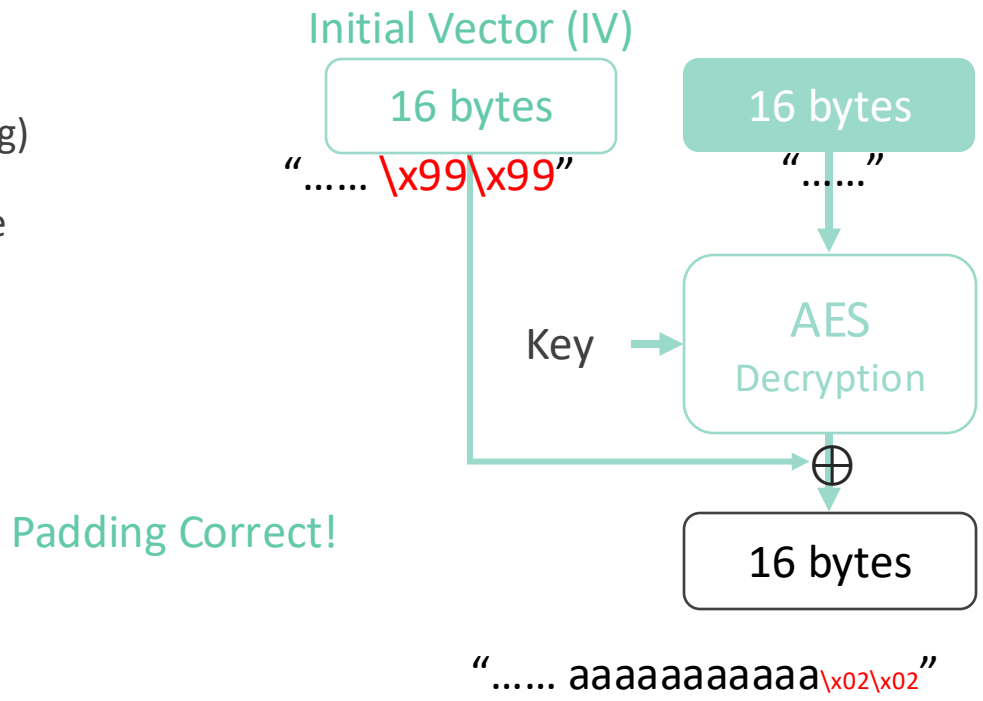
Block Cipher – mode of operation

- CBC mode – Padding Oracle Attack
  - 由於 Padding 通常會有一定的規則 (e.g., PKCS#7 Padding)
  - 可以透過 Server 有沒有觸發 Error 來找出最後一個 Byte
  - 確定最後一個 Byte 再找出倒數第二個 Byte



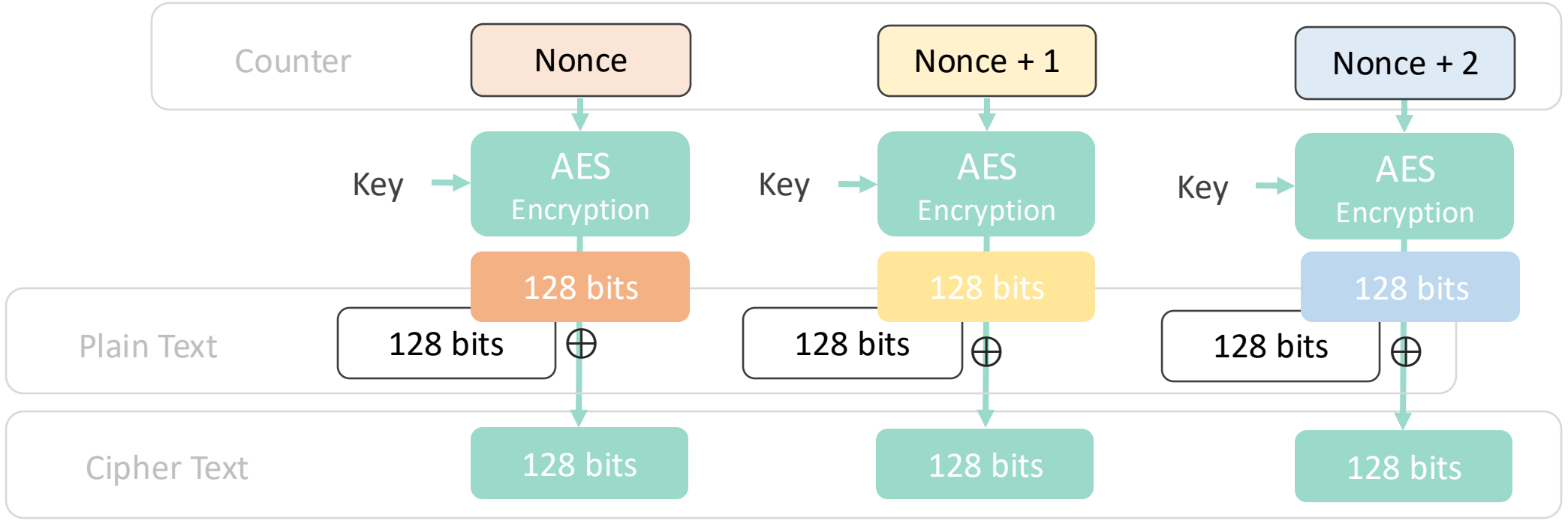
Block Cipher – mode of operation

- CBC mode – Padding Oracle Attack
  - 由於 Padding 通常會有一定的規則 (e.g., PKCS#7 Padding)
  - 可以透過 Server 有沒有觸發 Error 來找出最後一個 Byte
  - 確定最後一個 Byte 再找出倒數第二個 Byte



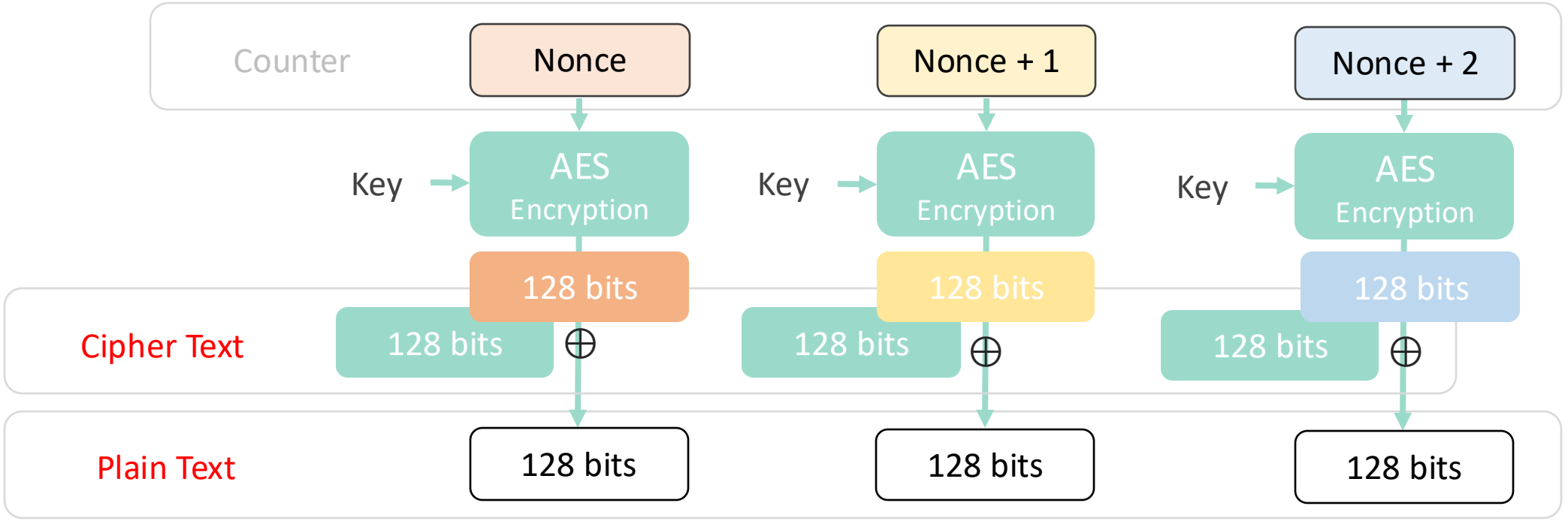
Block Cipher – mode of operation

- CTR mode (把 AES 當 PRNG)



Block Cipher – mode of operation

- CTR mode (把 AES 當 PRNG)

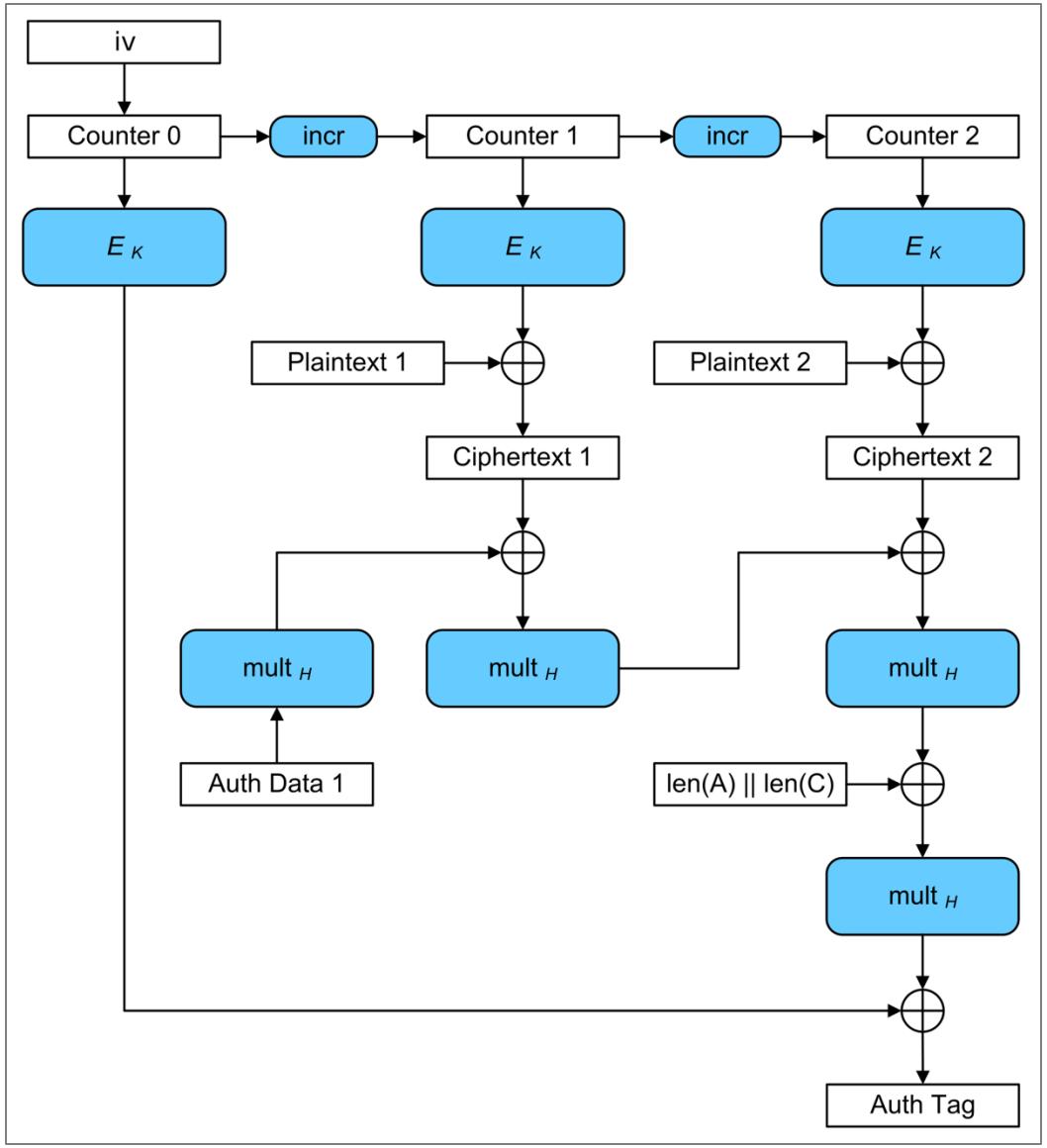


Block Cipher – mode of operation

- CTR mode (把 AES 當 PRNG)
  - 可以平行計算
  - 其實不需要 padding
  - 唯一的問題是 nonce 不能重複使用

Symmetric

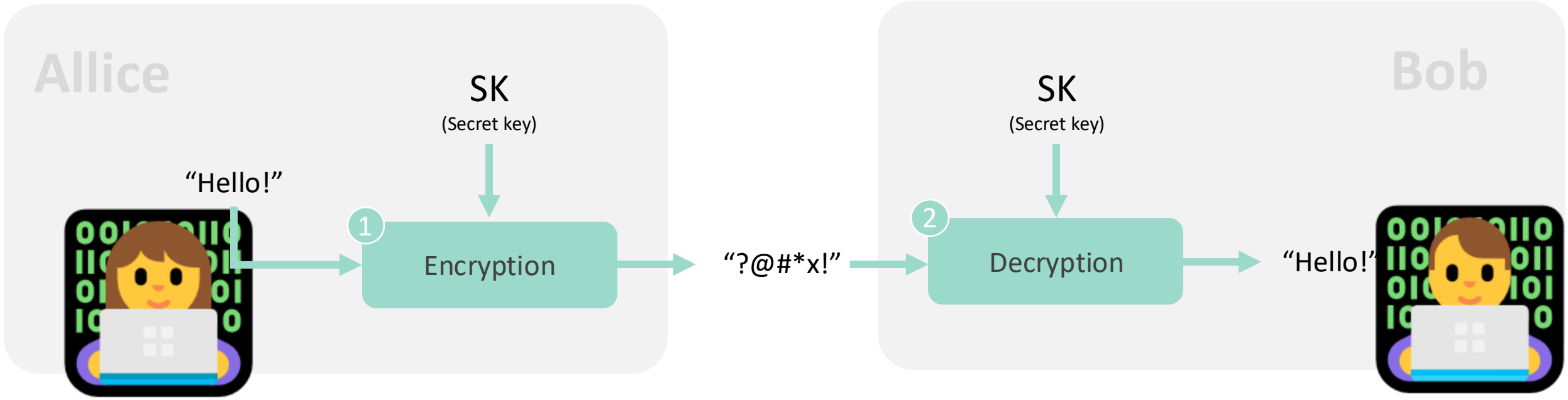
- CTR mode -> GCM mode



https://en.wikipedia.org/wiki/Galois/Counter_Mode#/media/File:GCM-Galois_Counter_Mode_with_IV.svg

Symmetric

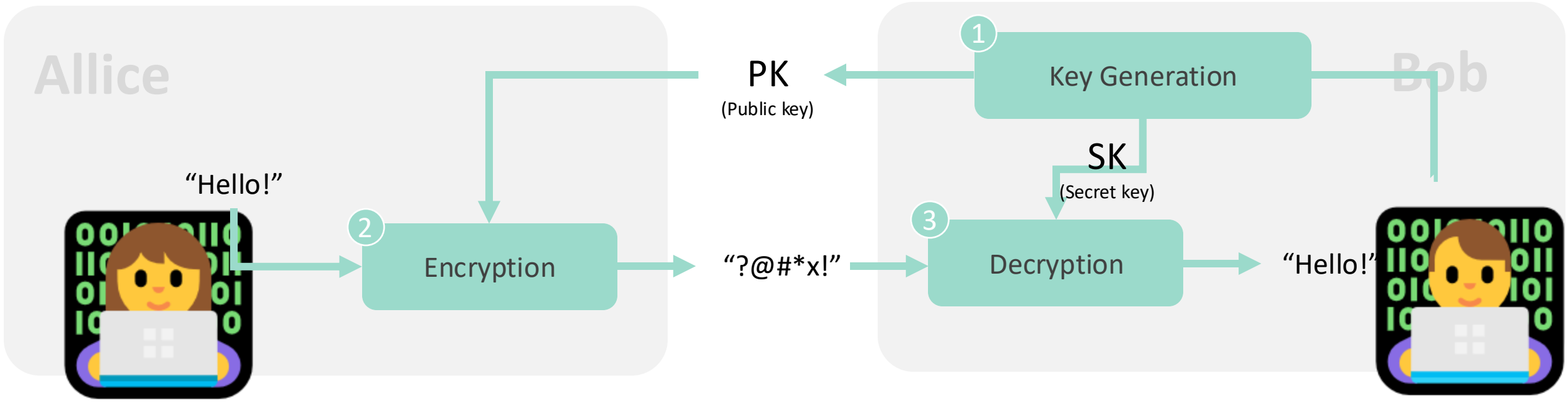
Symmetric Encryption



Questions?

Asymmetric

Asymmetric Encryption



- Cipher text 要看不出來 Plain Text
- PK 要沒有辦法推出 SK -> 透過一些數學難題

Asymmetric

RSA

- Introduction
- Attacks against RSA
  - Fermat's Factorization
  - Pollard's p-1
- Homomorphic Encryption

RSA

Large Numbers Factorization

With two large primes p and q, it is easy to compute p*q = n, but it is hard to factor n to find p and q.

RSA

Key Generation

1. Choose two large primes p and q, compute
   - $n = p*q$
   - $\phi = (p-1)(q-1)$

2. Choose e such that $GCD(e, \phi) = 1$, compute
   - $d = e^{-1} \bmod \phi$

3. Return $K_{pub} = (e, n)$, $K_{pr} = d$

RSA

Encryption

Raise the message m to the power e

- $c = m^e \pmod n$

Decryption

Raise the ciphertext c to the power d (private key)

- $m = c^d \pmod n$

RSA

Correctness

1. Fermat's little theorem ($a^{p-1} = 1 \pmod p$)

   $m^{\phi} = (m^{p-1})^{q-1} = 1^{q-1} = 1 \pmod p$

   $m^{\phi} = (m^{q-1})^{p-1} = 1^{p-1} = 1 \pmod q$

2. Chinese remainder theorem

   $m^{\phi} = 1 \pmod q$ && $m^{\phi} = 1 \pmod p \Rightarrow m^{\phi} = 1 \pmod n$

3. Since $d = e^{-1} \pmod \phi \Rightarrow ed = 1 \pmod \phi \Rightarrow ed = k\phi + 1$

   $c^d = m^{ed} = m^{k\phi+1} = 1^k * m = m \pmod n$

RSA

Key Generation

1. Choose two large primes p and q, compute
   - $n = p*q$
   - $\phi = (p-1)(q-1)$

2. Choose e such that $GCD(e, \phi) = 1$, compute
   - $d = e^{-1} \bmod \phi$
   - (commonly choose e = 2^16 + 1 = 65537)

3. Return $K_{pub} = (e, n)$, $K_{pr} = d$

Asymmetric

RSA ($n = pq$; $\phi = (p-1)(q-1)$; $d = e^{-1} \bmod \phi$)

1. 針對 n: Fermat's factorization、Pollard's p-1、Williams' p+1
2. 針對 e: e-th root、 broadcast attack
3. 針對 d: Wiener's Attack

Asymmetric

Fermat's factorization

適用於 p, q 很接近的時候，可以很快的分解 N

$N = ((N+1)/2)^2 - ((N-1)/2)^2$

1. 所有奇數都可以表達成兩個整數的平方差： $N = a^2 - b^2$
2. 所以當 p, q 很接近的時候：
    1. $N = p * q = (a + b)(a - b)$ -> (a+b) 跟 (a-b) 很接近
    2. 從 $a = sqrt(n)$ 開始暴搜 $b^2 = a^2 – N$ 成立的 b 就可以

Pollard’s p-1

適用於 p - 1 很 smooth (可以分解成很多個小質因數)的時候

Smooth:
1000000009 is prime, (1000000009 – 1) = 2^3 * 3^2 * 7 * 109^2 * 167

Not smooth:
1000000007 is also prime, but (1000000007 – 1) = 2 * 500000003

Pollard’s p-1

適用於 p - 1 很 smooth (可以分解成很多個小質因數)的時候

1. 假設 B >「p-1 最大的質因數」， 令 M = 1*2*3* … *B。
2. 則 gcd($a^M - 1 \pmod n$, n)= p。

Why：

1. 因為 B >「p-1 最大的因數」， 所以 M 會是 p – 1 倍數
2. 根據費馬小定理 $a^{(p-1)} = 1 \pmod p$， $a^M - 1 = a^{(p-1)k} - 1 = 0 \pmod p$
3. 所以 $a^M - 1 = kp \pmod n$

Asymmetric

Pollard’s p-1

適用於 p - 1 很 smooth (可以分解成很多個小質

1. 假設 B >「p-1 最大的質因數」， 令 M = 1*2*3* … *B。
2. 則 gcd($a^M - 1 \pmod n$, n)= p。

Why：

1. 因為 B >「p-1 最大的因數」， 所以 M 會是 p – 1 倍
2. 根據費馬小定理 $a^{(p-1)} = 1 \pmod p$， $a^M - 1 = a^{(p-1)k} - 1$
3. 所以 $a^M - 1 = kp \pmod n$

```python
def pollards_p_minus_1(N, B=10000):
    """
    Pollard's p-1 factorization method.
    Tries to find a nontrivial factor of N.

    Parameters:
        N: int - the number to factor
        B: int - smoothness bound (default 10000)

    Returns:
        A nontrivial factor of N, or None if failed
    """
    a = 2
    for j in range(2, B):
        a = pow(a, j, N)
        d = gcd(a - 1, N)
        if 1 < d < N:
            return d
    return None
```

Homomorphic Encryption

E.g., RSA, Paillier Encryption…

1. Homomorphism: $f(a) * f(b) = f(a \circ b)$
2. In RSA: $m_1^e * m_2^e = (m_1 m_2)^e$

=> LSB Oracle? 可以利用 $c * (2^{-k})^e = (m * 2^{-k})^e$!

Asymmetric

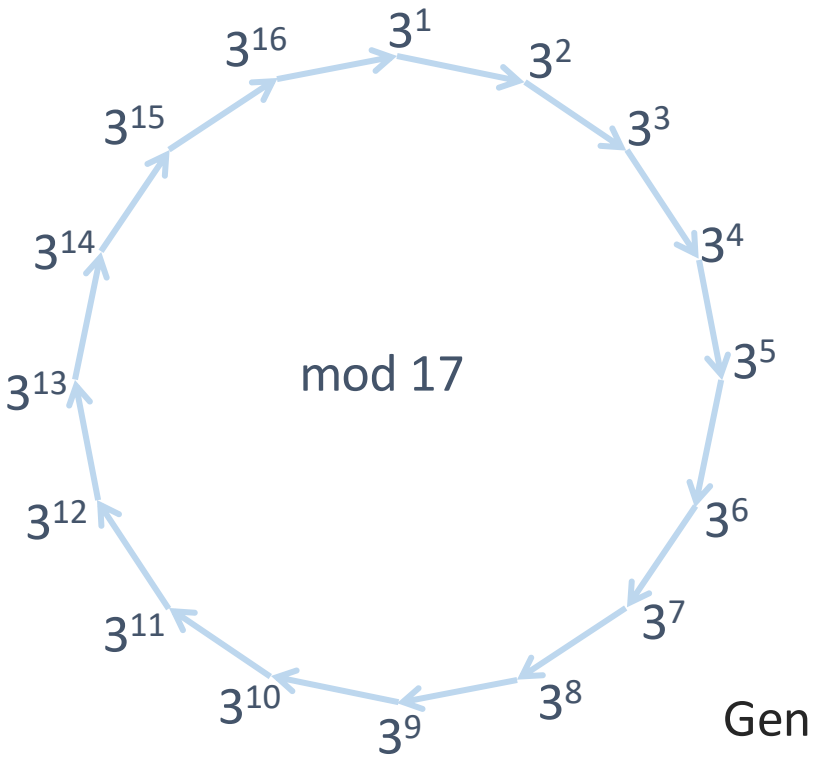
ElGamal Encryption

- DLP (Discrete Logarithm Problem)
- Introduction
- Diffie-Hellman Key Exchange
- Attacks on DLP
  - Brute force
  - Baby-step Giant-step
  - Pohlig Hellman

ElGamal Encryption (Discrete Logarithm)

Discrete Logarithm Problem

- Given a prime p, a generator g, we can have a cyclic group.



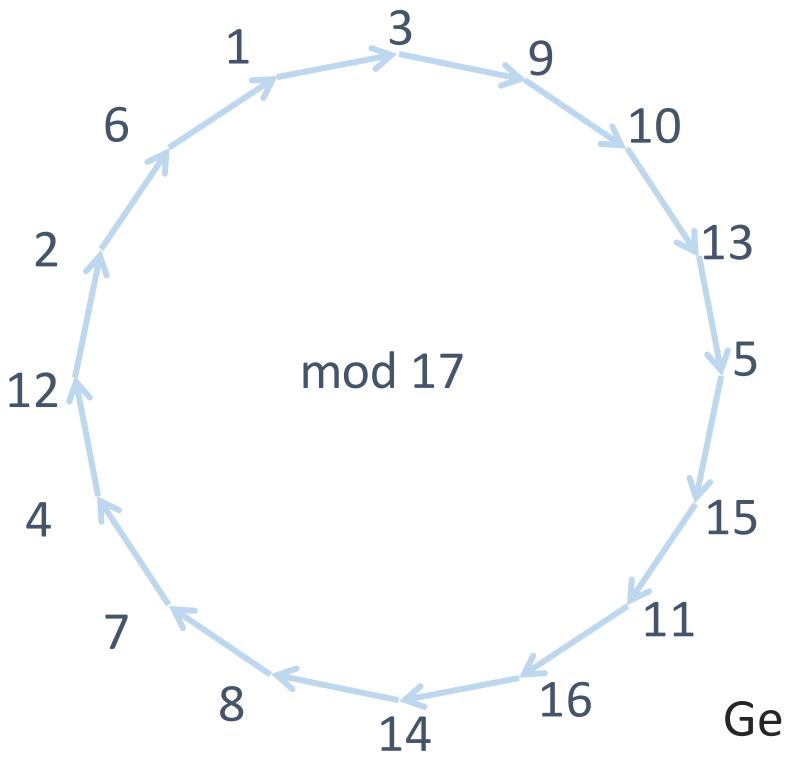
Generator : 3

Asymmetric

ElGamal Encryption (Discrete Logarithm)

Discrete Logarithm Problem

- Given a prime p, a generator g, we can have a cyclic group.
- The number of elements in the group is ”order”.



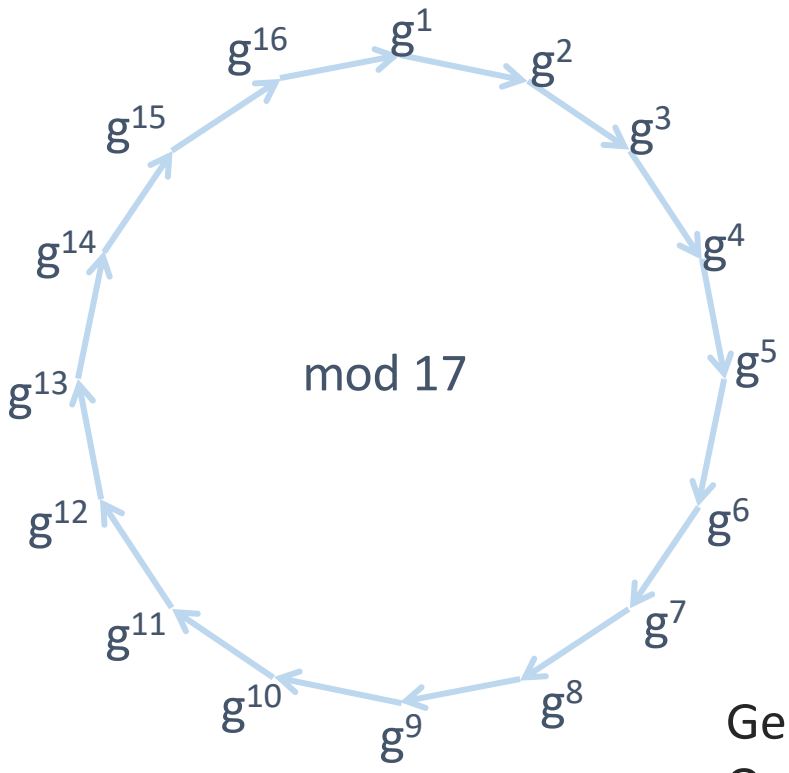
Generator : 3
Order: 16

Asymmetric

ElGamal Encryption (Discrete Logarithm)

Discrete Logarithm Problem

- Given a prime p, a generator g, we can have a cyclic group.
- The number of elements in the group is ”order”.



Generator :g
Order: 16

Asymmetric

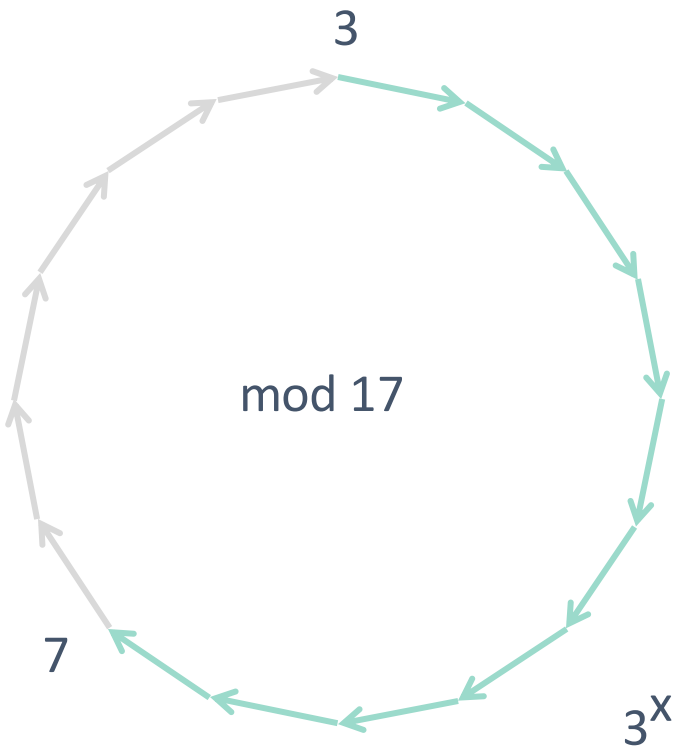
ElGamal Encryption (Discrete Logarithm)

Discrete Logarithm Problem

- Given a prime p, a generator g, we can have a cyclic group.
- given y in $g^x = y \pmod p$, x = ?

  計算 $3^{11} = 7 \pmod{17}$ 很簡單（可以做快速冪）

  計算 $7 = 3^{?} \pmod{17}$ 很難



$3^X = 7 \bmod 17$

X = ?

ElGamal Encryption (Discrete Logarithm)

Key Generation

1. Choose a large prime p, generator g, and private key b, compute
   - $B = g^b \pmod p$

2. Return $K_{pub} = (p, g, B)$, $K_{pr} = b$

ElGamal Encryption (Discrete Logarithm)

Encryption

1. Choose a random a, compute ephemeral key A and masking key M
   - $A = g^a \pmod p$
   - $M = B^a \pmod p$
   - CipherText = M * m (mod p)
2. Return (A, CipherText)

Decryption

1. Compute masking key M with A, b
   - $M = A^b \pmod p$
2. Recover message m
   - $m = \text{CipherText} * M^{-1} \pmod p$

ElGamal Encryption (Discrete Logarithm)

Correctness

$M = B^a = g^{ab} = A^b = M \pmod p$

Other people without key a or b cannot recover g^{ab} by observing the public $B = g^b$ and $A = g^a$.

Asymmetric

Diffie-Hellman Key Exchange

Allice

Generate secret key a

$A = g^a \pmod p$

$K = B^a \pmod p$

Have shared secret $K = g^{ab}$

A, B (Public)

Bob

Generate secret key b

$B = g^b \pmod p$

$K = A^b \pmod p$

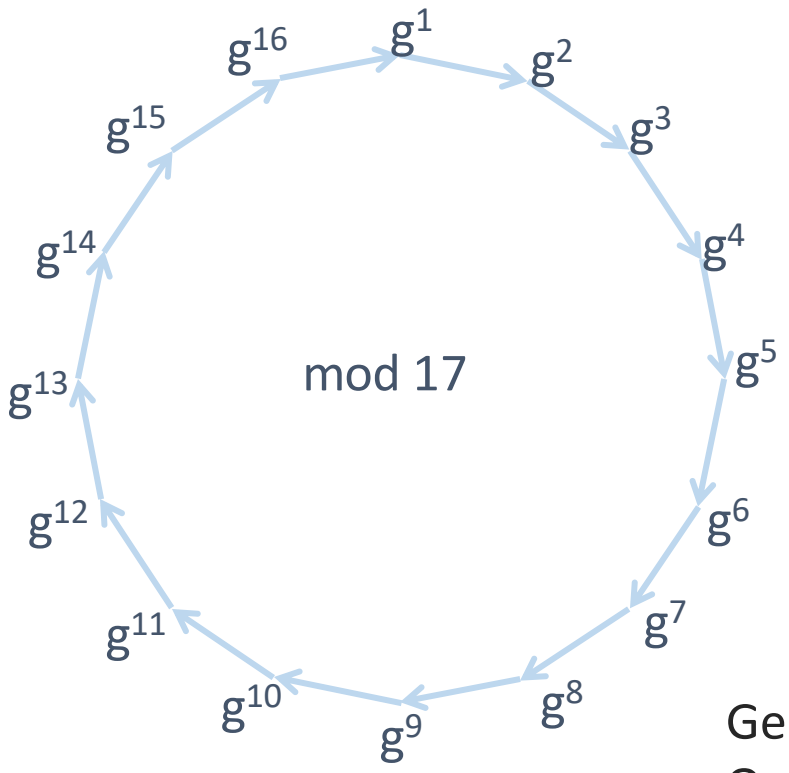
Have shared secret $K = g^{ab}$

Asymmetric

ElGamal Encryption (Discrete Logarithm)

Discrete Logarithm Problem

- Given a prime p, a generator g, we can have a cyclic group.
- The number of elements in the group is ”order”.

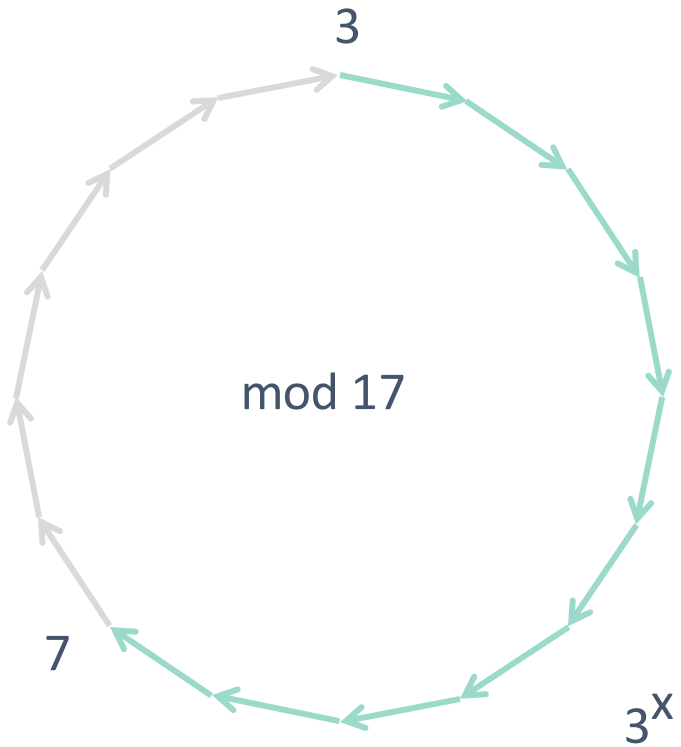


Generator :g
Order: 16

Attack on DLP – Brute force

- $3^1 = 3 \pmod{17}$
- $3^2 = 9 \pmod{17}$
- $3^3 = 10 \pmod{17}$
- $3^4 = 13 \pmod{17}$
- …

O(p)



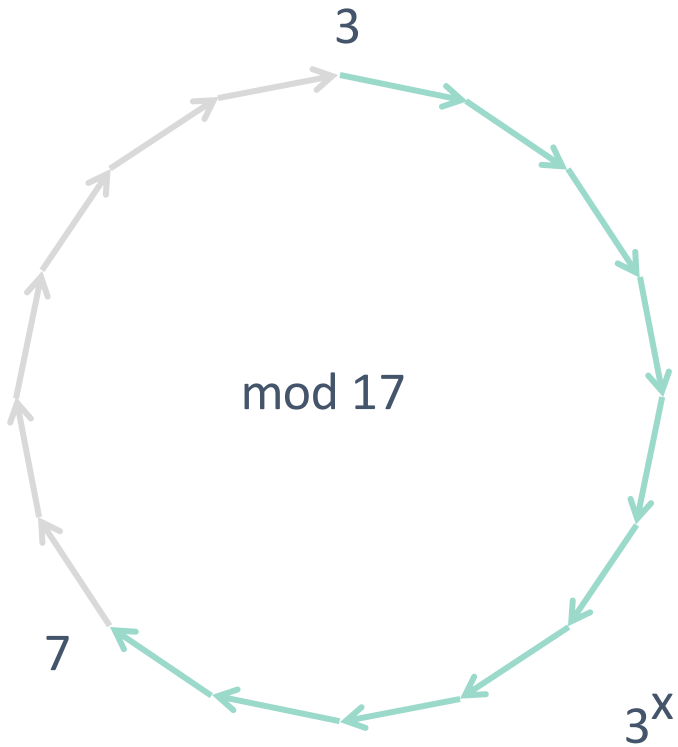
$3^X = 7 \bmod 17$

X = ?

Attack on DLP – Baby-Step Giant-Step

- Let m = sqrt(p), x 可以寫成 am + b
- b 的部分（for all b < sqrt(p)） :
  - Pre-calculate $g^b = B \pmod p$
  - 把結果預先建表
- a 的部分（ for all a < sqrt(p) ）
  - 檢查 y / g^{am} 有沒有對應到表中任何 B

O(sqrt(p) *查表)

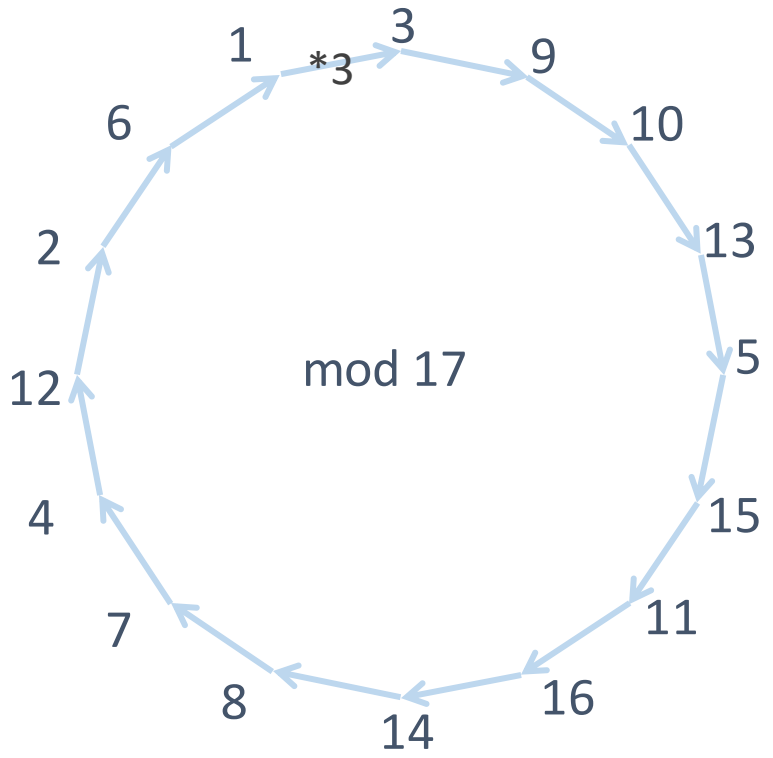


$3^X = 7 \pmod{17}$

X = ?

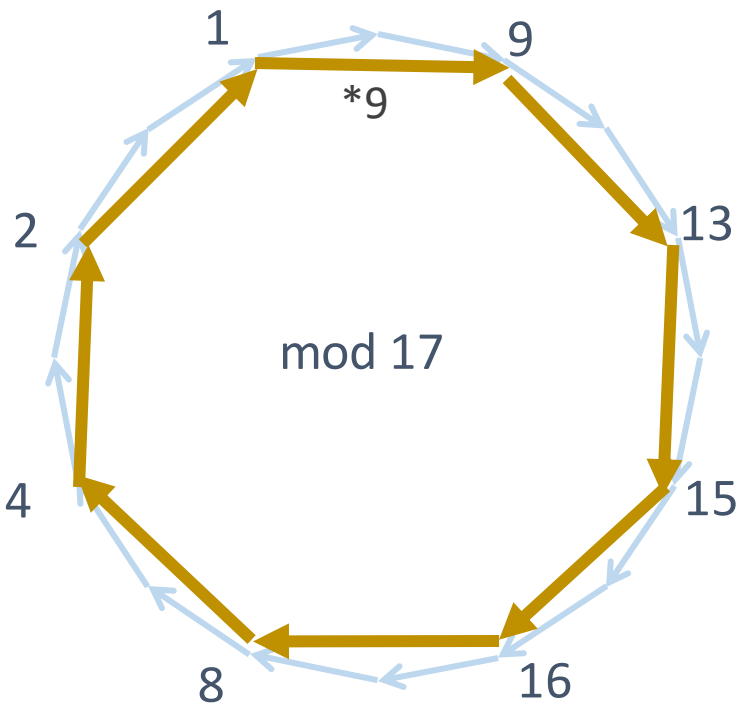
Attack on DLP – Pohlig Hellman

- If p - 1 is smooth.
- 我們可以把原本 order p – 1 的 group 拆成很多 order 很小的 subgroup!



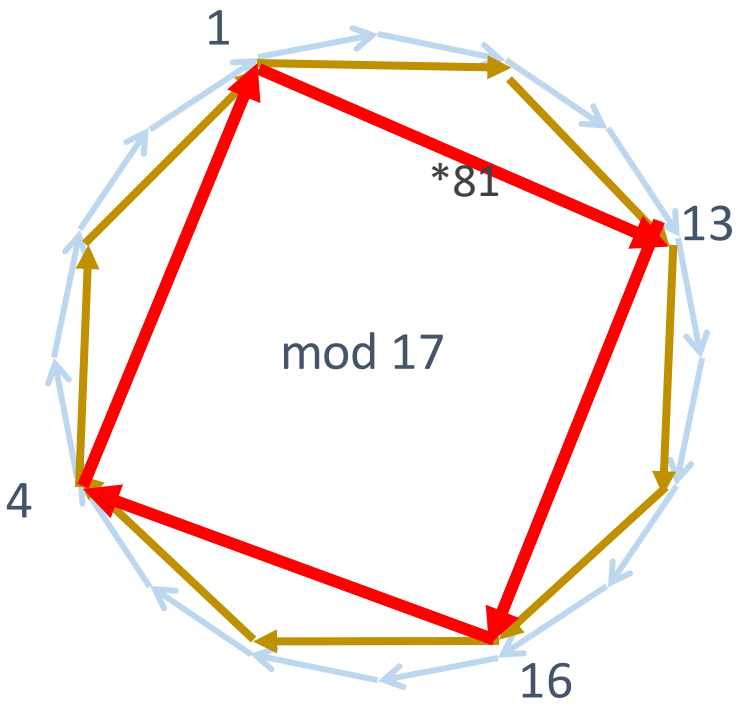
Attack on DLP – Pohlig Hellman

- If p - 1 is smooth.
- 我們可以把原本 order p – 1 的 group 拆成很多 order 很小的 subgroup!



Attack on DLP – Pohlig Hellman

- If p - 1 is smooth.
- 我們可以把原本 order p – 1 的 group 拆成很多 order 很小的 subgroup!
- 在 subgroup 裡面計算 dlp 問題變得很簡單!



Attack on DLP – Pohlig Hellman

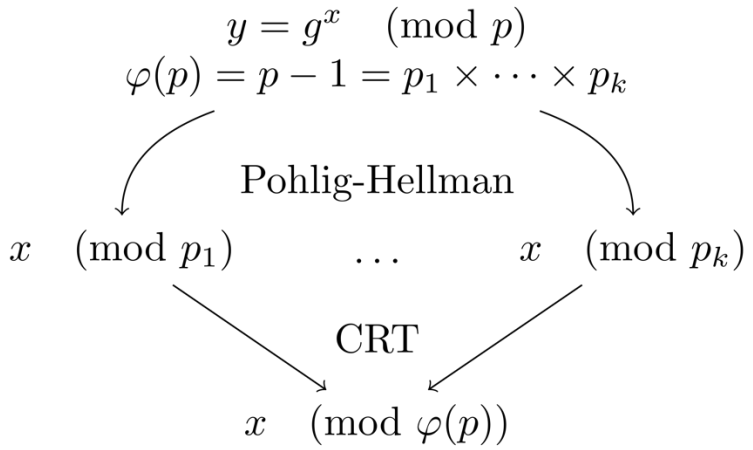
假設 $g^x = y \pmod p$ 求 x，且 $(p-1) = a * b * c$

- **底數為** g 的 group 有 order (p-1)
- **底數為** $g^{(p-1)/a}$ 的 subgroup 有 order a

⇒ $(g^x)^{(p-1)/a} = y^{(p-1)/a}$

⇒ $(g^{(p-1)/a})^x = y^{(p-1)/a}$

⇒ **我們可以得到** x mod a 的值



Attack on DLP – Pohlig Hellman

假設 $g^x = y \pmod p$ 求 x，且 $(p-1) = a * b * c$

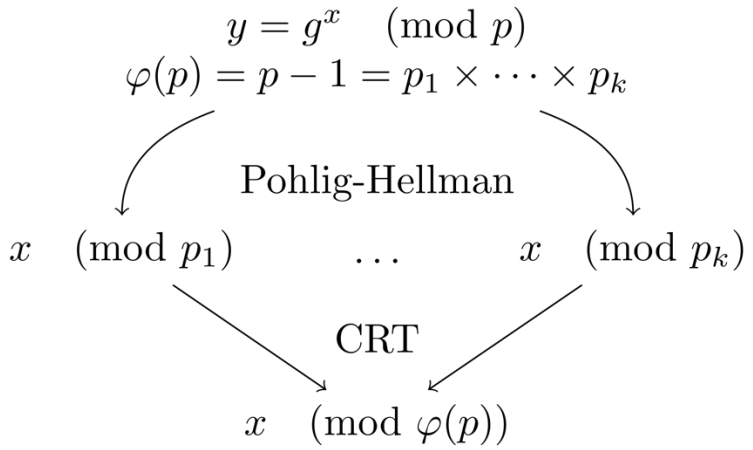
- 底數為 g 的 group 有 order (p-1)
- 底數為 $g^{(p-1)/b}$ 的 subgroup 有 order b

$\Rightarrow (g^x)^{(p-1)/b} = y^{(p-1)/b}$

$\Rightarrow (g^{(p-1)/b})^x = y^{(p-1)/b}$

⇒ 我們可以得到 x mod b 的值 …

有了 x mod a, x mod b, x mod c 就可以做 CRT 了!



Attack on DLP – Pohlig Hellman

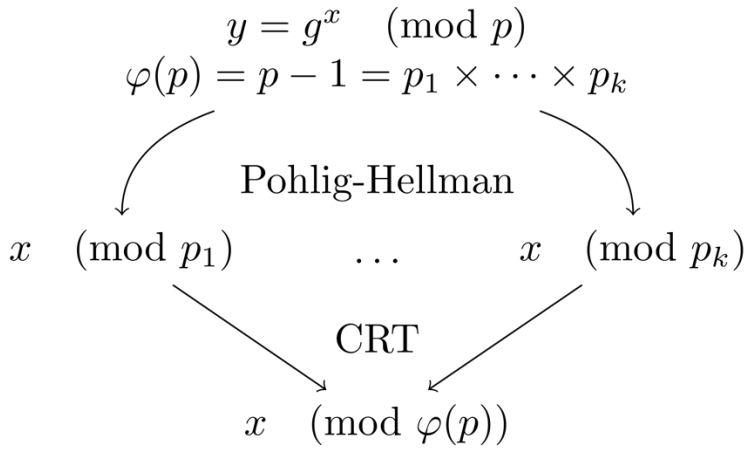
假設 $g^x = y \pmod p$ 求 x，且 $(p-1) = a^k$

- **底數為** g 的 group 有 order (p-1)
- **底數為** $g^{(p-1)/a}$ 的 subgroup 有 order a

$\Rightarrow (g^x)^{(p-1)/a} = y^{(p-1)/a}$

$\Rightarrow (g^{(p-1)/a})^x = y^{(p-1)/a}$

$\Rightarrow$ **我們可以得到** x mod a 的值，假設是 a_1



Attack on DLP – Pohlig Hellman

假設 $g^x = y \pmod p$ 求 x，且 $(p-1) = a^k$

- **底數為** g 的 group 有 order (p-1)
- **底數為** $g^{(p-1)/a}$ 的 subgroup 有 order a

$\Rightarrow (g^{x-a_1}) = (y^* g^{-a_1})$

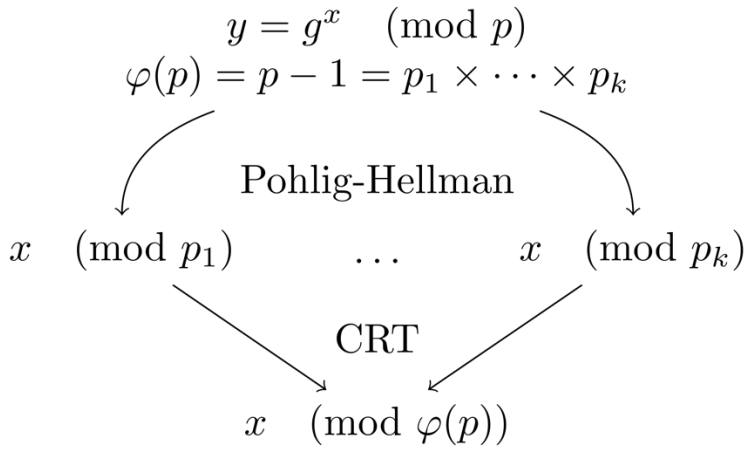
$\Rightarrow (g^{x-a_1})^{(p-1)/a^2} = (y^* g^{-a_1})^{(p-1)/a^2}$

$\Rightarrow (g^{(p-1)/a})^{(x-a_1)/a} = (y^* g^{-a_1})^{(p-1)/a^2}$

$\Rightarrow (g^{(p-1)/a})^{(x-a_1)/a} = (y^* g^{-a_1})^{(p-1)/a^2}$

$\Rightarrow$ **我們可以得到** $(x-a_1)/a \bmod a$ 的值 -> 得到 $x \bmod a^2$ 的值

$x = ? + ?a^1 + ?a^2 + ?a^3 + \ldots + ?a^k$



Attack on DLP – Pohlig Hellman

假設 $g^x = y \pmod p$ 求 x，且 $(p-1) = a^k$

- **底數為** g 的 group 有 order (p-1)
- **底數為** $g^{(p-1)/a}$ 的 subgroup 有 order a

$\Rightarrow (g^{x-a_1}) = (y^* g^{-a_1})$

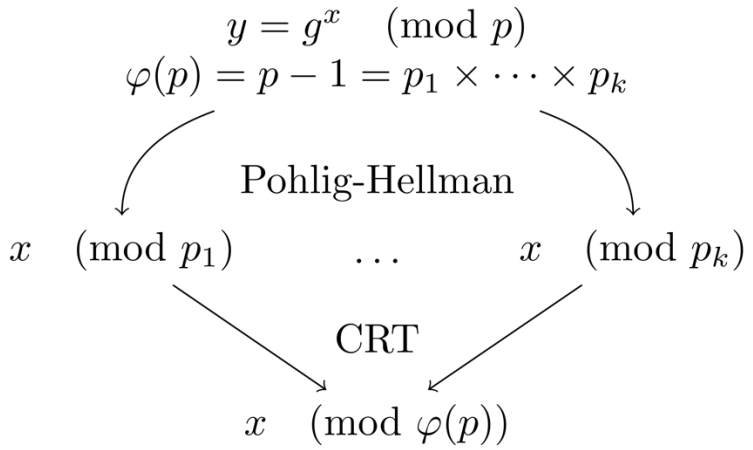
$\Rightarrow (g^{x-a_1})^{(p-1)/a^2} = (y^* g^{-a_1})^{(p-1)/a^2}$

$\Rightarrow (g^{(p-1)/a})^{(x-a_1)/a} = (y^* g^{-a_1})^{(p-1)/a^2}$

$\Rightarrow (g^{(p-1)/a})^{(x-a_1)/a} = (y^* g^{-a_1})^{(p-1)/a^2}$

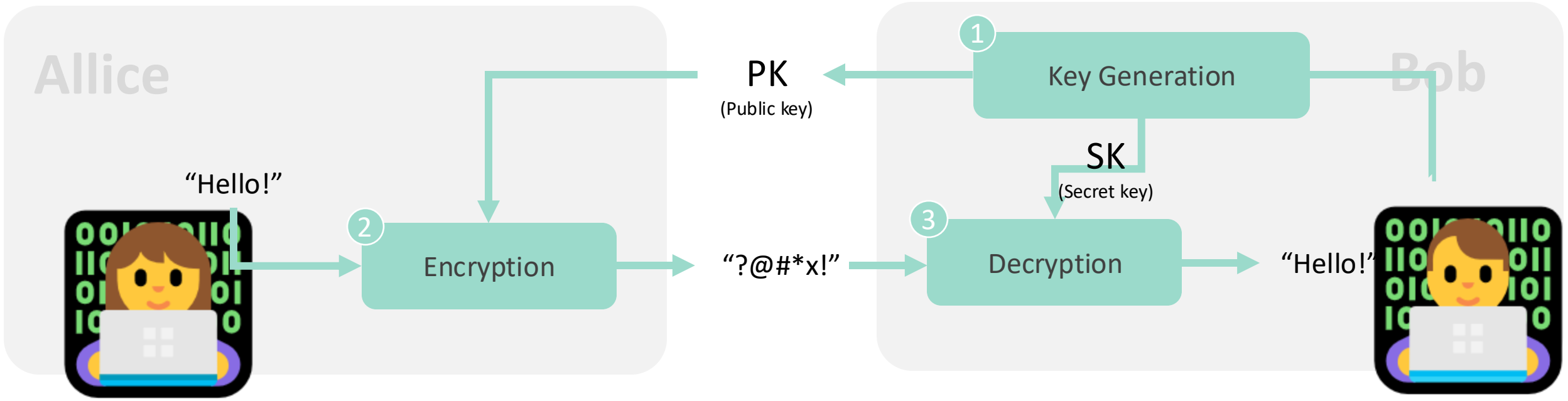
$\Rightarrow$ **我們可以得到** $(x-a_1)/a \bmod a$ 的值 -> 得到 $x \bmod a^2$ 的值

$x = ? + ?a^1 + ?a^2 + ?a^3 + \ldots + ?a^k$



Asymmetric

Asymmetric Encryption



- Cipher text 要看不出來 Plain Text
- PK 要沒有辦法推出 SK -> 透過一些數學難題

Questions?

Primitives

Other primitives in Cryptography

1. Hash function
2. Digital Signature
3. Zero-Knowledge Proof

Primitives

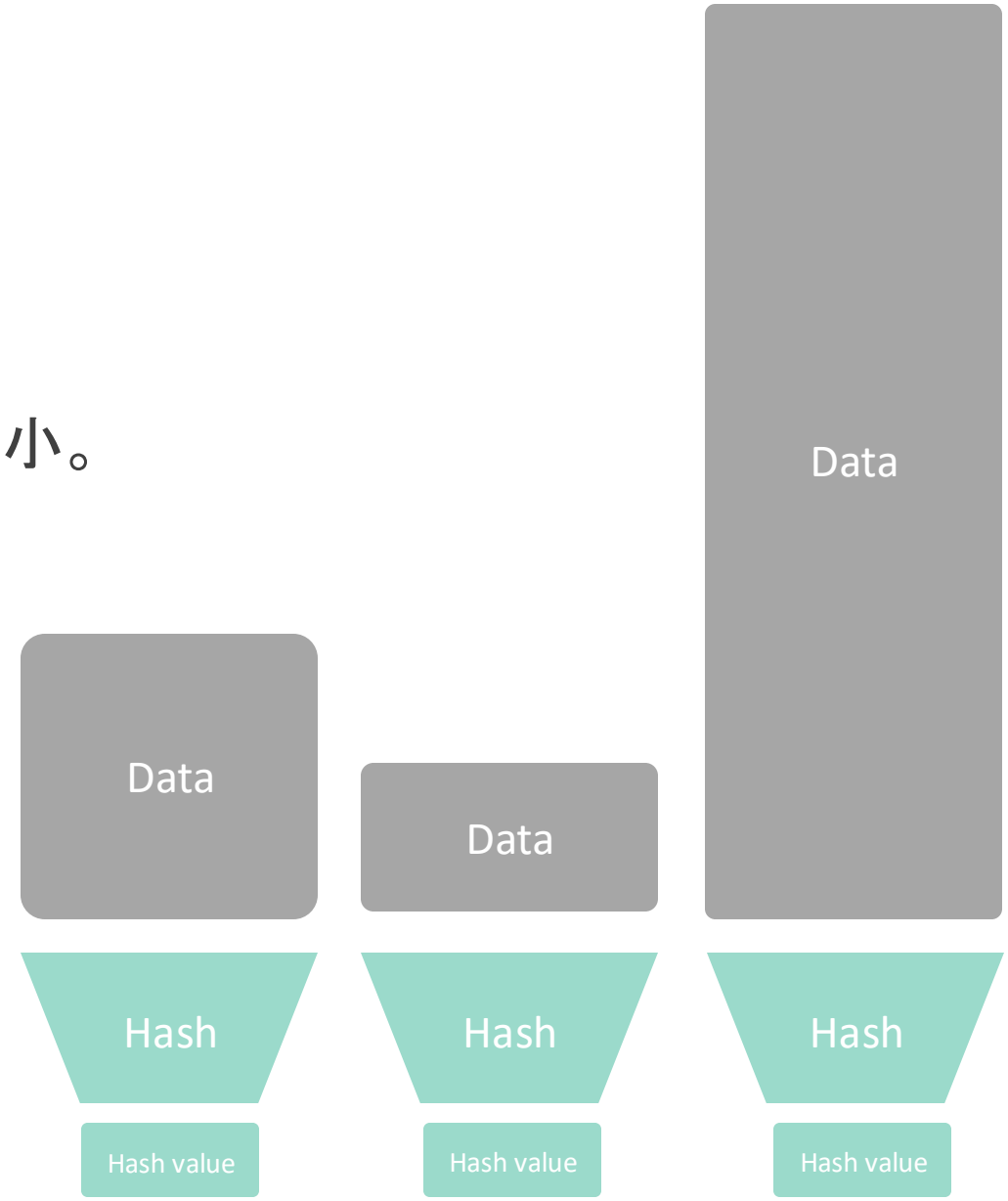
Hash Function

- Introduction
- Length Extension Attack

Primitives

Hash Function（雜湊函數）

把任意大小的資料，摘要成固定的大小。



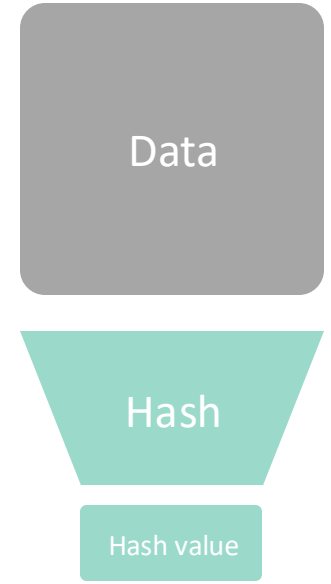
Primitives

Hash Function（雜湊函數）

性質：

1. 無論輸入多長的資料都會變成固定的大小。
2. 輸入一樣的資料會得到一樣的結果。
3. 輸入不一樣的資料會得到不一樣的結果。

   (collision: 不一樣的資料對應到一樣的 Hash value)
4. 不能從 Hash value 逆推回去 Data。



Primitives

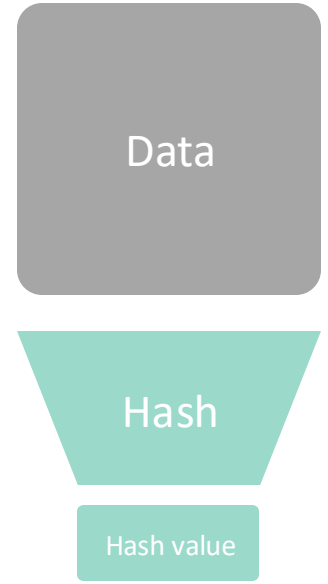
Hash Function（雜湊函數）

用途：

1. 下載檔案時的 Hash value 檢查
2. Database 裡面儲存使用者的密碼資料
3. HMAC (hash-based message authentication code) 雜湊訊息鑑別碼

（把自己的 key 放進訊息裡進行 Hash，可以事後再用同樣的 key 認證該訊息確實出自自己）

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



Hash Function（雜湊函數）

常見的 Hash function：

md5, sha1, sha256, sha512 ……

MD5 collision: https://www.mscs.dal.ca/~selinger/md5collision/
Sha1 collision: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

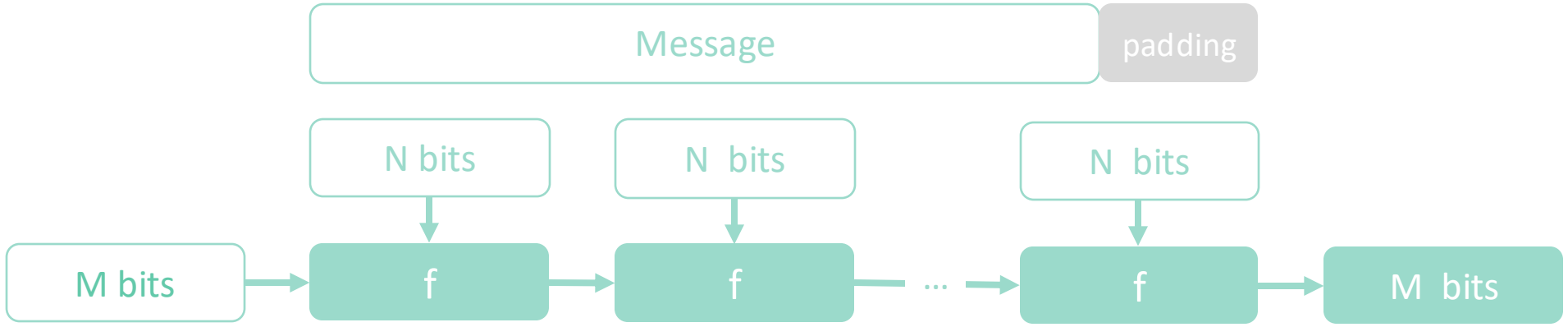
sha256 或甚至 sha512 的 collision 是很難找的，但 ...?

Hash Function（雜湊函數）

Length Extension attack

md5, sha1 sha256, sha512 基本上都是利用 Merkle-Damgård 構造

來達成 “無論輸入多長的資料都會變成固定的大小”。

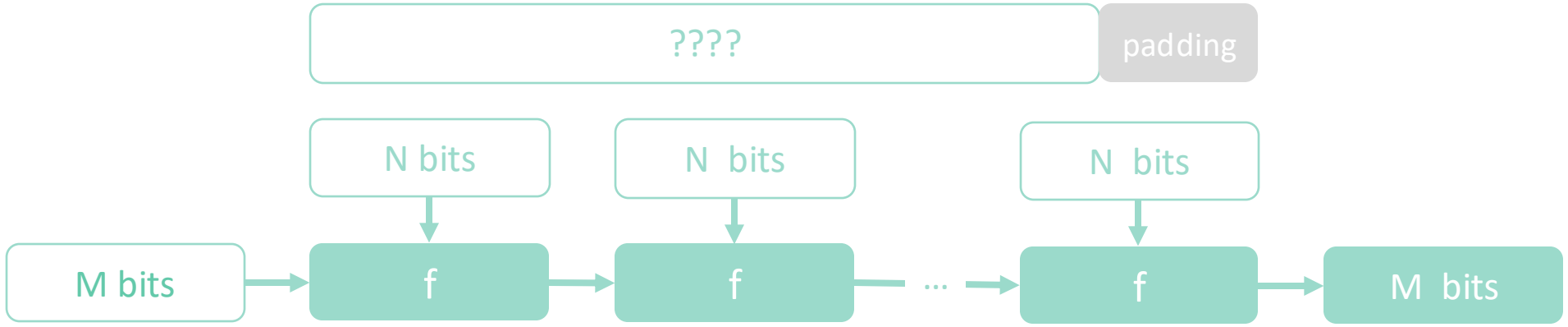


Hash Function（雜湊函數）

Length Extension attack

md5, sha1 sha256, sha512 基本上都是利用 Merkle-Damgård 構造

來達成 “無論輸入多長的資料都會變成固定的大小”。 => **可以直接加長！**

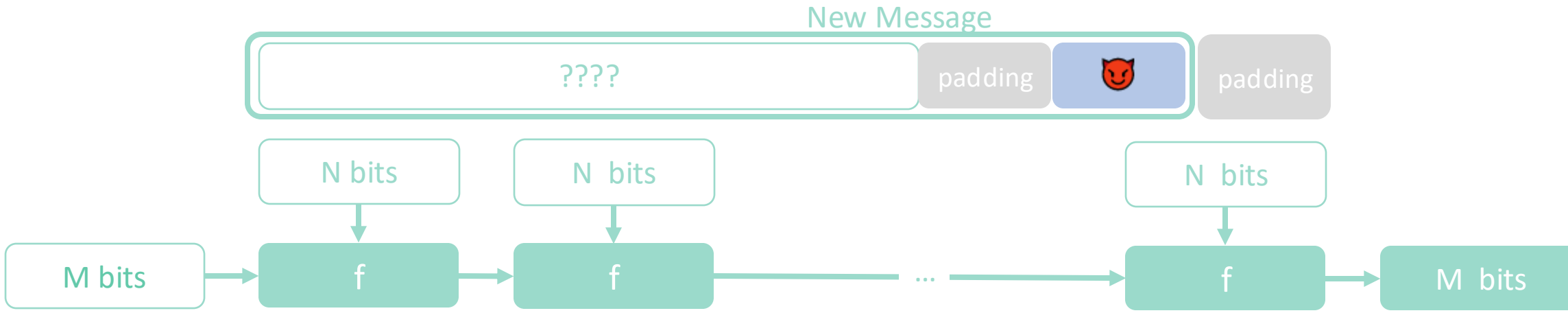


Hash Function（雜湊函數）

Length Extension attack

md5, sha1 sha256, sha512 基本上都是利用 Merkle-Damgård 構造

來達成 “無論輸入多長的資料都會變成固定的大小”。 => **可以直接加長！**



Hash Function（雜湊函數）

Length Extension attack

可以攻擊一些設計不良的 HMAC：無需知道 key，就能從一些 HMAC 值推測出另外一些 HMAC 值

Hash Function（雜湊函數）

In Length Extension Attack:

1. 無論輸入多長的資料都會變成固定的大小 -> 新的輸出還是固定長度
2. 輸入一樣的資料會得到一樣的結果 -> 假造的輸入輸出也還是符合該 Hash
3. 輸入不一樣的資料會得到不一樣的結果 -> 新的輸入跟舊的有不同 hash value
4. 不能從 Hash value 逆推回去 Data -> 不知道的 key 還是不知道

完全沒有問題…！？

Hash Function（雜湊函數）

目前常用的 SHA256, SHA512 都屬於 SHA-2 （第二代）， 演算法的設計都跟 SHA-1, SHA-0 比較類似。

在新一代的 SHA-3 就是很不一樣的另一個世界了（至少沒辦法 Length Extension）。

但由於目前沒有什麼針對 SHA-2 有效的攻擊、 SHA-2 又比較快、比較廣泛被使用、兼容性高，所以目前還是比較常用 SHA-2。

Primitives

Digital Signature

- Introduction
- RSA signature
- DSA (Digital Signature Algorithm)

Digital Signature – Introduction

在數位化的世界中，有沒有什麼密碼學方法可以向所有人證明訊息來自於特定的人呢？

HMAC?

-> 驗證跟計算是同一個 secret key (任何人都可以簽就沒有意義了)

Public Key Cryptography?

-> 用私鑰計算、用公鑰驗證！

Digital Signature - RSA Signature

Key Generation

1. $n = pq$
2. $\phi = (p-1)(q-1)$
3. $d = e^{-1} \bmod \phi$

Encryption

- $c = m^e \pmod n$

Decryption

- $m = c^d \pmod n$

Digital Signature - RSA Signature

Key Generation

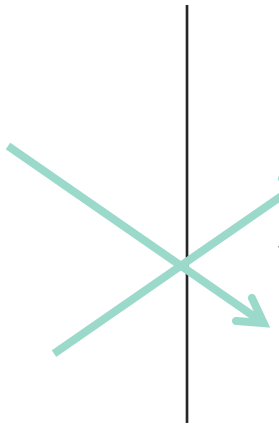
1. $n = pq$
2. $\phi = (p-1)(q-1)$
3. $d = e^{-1} \bmod \phi$

Encryption

- $c = m^e \pmod n$

Decryption

- $m = c^d \pmod n$



Signing

- $s = m^d \pmod n$

Verifying

- $m = s^e \pmod n$

只有 Alice 知道私鑰 d

大家都可以用公鑰 e 檢查 s

Digital Signature - RSA Signature

Key Generation

1. $n = pq$
2. $\phi = (p-1)(q-1)$
3. $d = e^{-1} \bmod \phi$

Signing　　只有 Alice 知道私鑰 d

- $s = m^d \pmod n$

Verifying　　大家都可以用公鑰 e 檢查 s

- $m = s^e \pmod n$

Primitives

Digital Signature

在數位化的訊息中，有沒有什麼密碼學方法可以向全世界證明訊息來自於特定的人呢？

性質：

1. Authentication: 能夠證明是 Alice 簽的
2. Integrity: Alice 簽過的東西不能再更改
3. Non-repudiation: Alice 不能否認自己說過

Digital Signature - RSA Signature

Key Generation

1. $n = pq$
2. $\phi = (p-1)(q-1)$
3. $d = e^{-1} \bmod \phi$

Signing　　只有 Alice 知道私鑰 d

- $s = m^d \pmod n$ -> $s = (m + kn)^d$

Verifying　　大家都可以用公鑰 e 檢查 s

- $m = s^e \pmod n$

❌ 違反 Integrity: Alice 簽過的東西不能再更改！

Digital Signature - RSA Signature

Key Generation

1. $n = pq$
2. $\phi = (p-1)(q-1)$
3. $d = e^{-1} \bmod \phi$

Signing　　只有 Alice 知道私鑰 d

- $s = Hash(m)^d \pmod n$

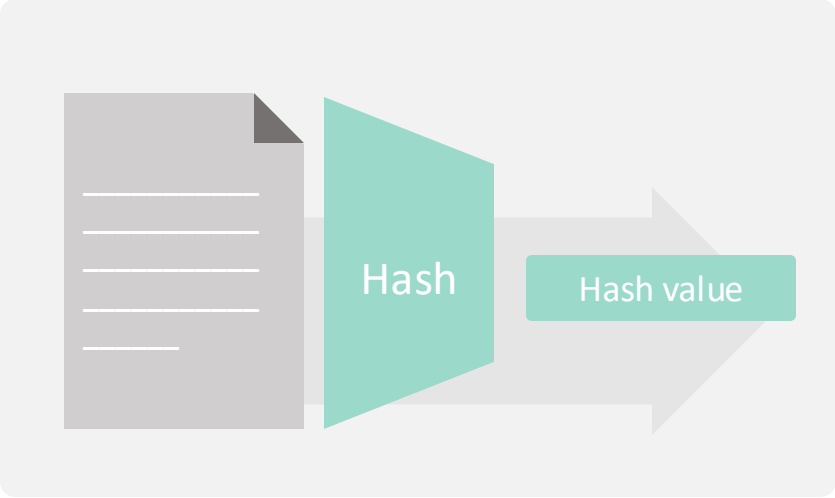
Verifying　　大家都可以用公鑰 e 檢查 s

- $m = s^e \pmod n$

✅ 只要 Hash 不 collision 就不會有問題

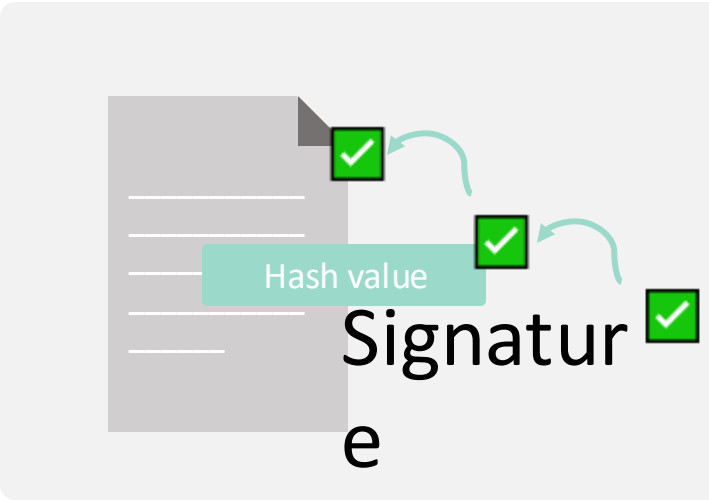
Digital Signature

1) Generate the Hash value of the file



2) Sign the Hash value

3) The signature guarantees the file



Digital Signature - DSA (Digital Signature Algorithm)

Key Generation

DSA 早期的建議位元數，後來有變多

1. Choose prime p (1024 bits), q (160 bits), 且 (p-1) % q = 0
2. Choose random h from [2, p-2],let g = $h^{(p-1)/q}$ (mod p) (直接 h = 2 比較方便)
3. Choose random x from [1, q-1], let y = g^x (mod p) (Public: y; Private: x)

Signing

1. Choose random k from [1, q-1]
2. Compute r = (g^k mod p) mod q, s = (k^{-1}(*message* + x * r)) mod q

訊息的 Hash 值

3. Signature = (r, s)

Digital Signature - DSA (Digital Signature Algorithm)

Verifying

1. Check $0 < r < q$, $0 < s < q$
2. Let $u1 = message * s^{-1} \pmod q$, $u2 = r * s^{-1} \pmod q$
3. Check $(g^{u1}y^{u2} \bmod p) \bmod q \stackrel{?}{=} r$

Digital Signature - DSA (Digital Signature Algorithm)

Correctness of "Check $(g^{u1}y^{u2} \bmod p) \bmod q \stackrel{?}{=} r$" ?

$g = h^{(p-1)/q} \pmod p$

$y = g^x \pmod p$

$r = (g^k \bmod p) \bmod q$

$s = (k^{-1}(message + x*r)) \bmod q$

$u1 = message * s^{-1} \pmod q$

$u2 = r * s^{-1} \pmod q$

$u1 + x * u2 \pmod q$

$= message * s^{-1} + x * r * s^{-1} \pmod q$

$= (message + x*r) * s^{-1} \pmod q$

$= (message + x*r) / k^{-1}(message + x*r) \pmod q$

$= k \pmod q$

$(g^{u1}y^{u2} \bmod p) \bmod q$

$= (g^{(u1 + x * u2)} \bmod p) \bmod q$

$= (g^k \bmod p) \bmod q = r$

*recall: g has order q!!

Digital Signature - DSA (Digital Signature Algorithm)

基本上跟 ElGamal 一樣,

把安全性建立在 $g^x = y \pmod p$ 的難度上

需要注意的是 random k 必須要夠 random 、不能被預測

舉例來說, 兩個 signature 用一樣的 k 的話：

* Sony playstaion3 的 ECDSA 就有這個 Bug

$r1 = (g^k \bmod p) \bmod q;\ s1 = (k^{-1}(\mathit{message1} + x*r1)) \bmod q$

$r2 = (g^k \bmod p) \bmod q;\ s2 = (k^{-1}(\mathit{message2} + x*r2)) \bmod q$

$r1 = r2 = r$(r都一樣) $\Rightarrow k * s1 = message1 + x * r \Rightarrow k * s2 = message2 + x * r$

$message2 * s1 + x * r * s1 = message1 * s2 + x * r * s2$

$x = message2*s1 - message1*s2 / (r*s2 - r*s1)$

得到私鑰了！

Digital Signature - DSA (Digital Signature Algorithm)

Deterministic DSA's Signing

1. ~~Choose random k from [1, q-1]~~ k = Hash(g, x, p, q, *message*)
2. Compute r = $(g^k \bmod p) \bmod q$, s = $(k^{-1}(message + x * r)) \bmod q$
3. Signature = (r, s)

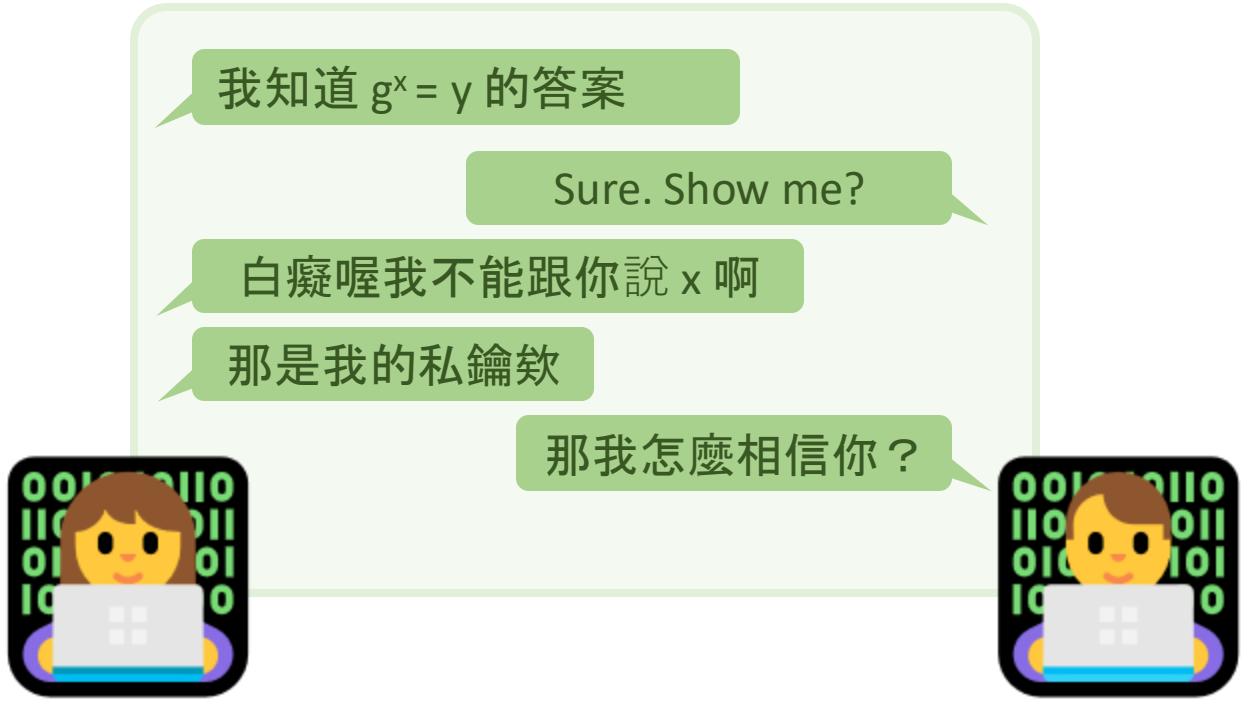
確保同一個私鑰、同一個訊息會有一個固定的夠亂的 k
且對於不知道 x 的他人來說無法預測**、也不能逆推出 x 的方法**

Primitives

Zero-Knowledge Proof

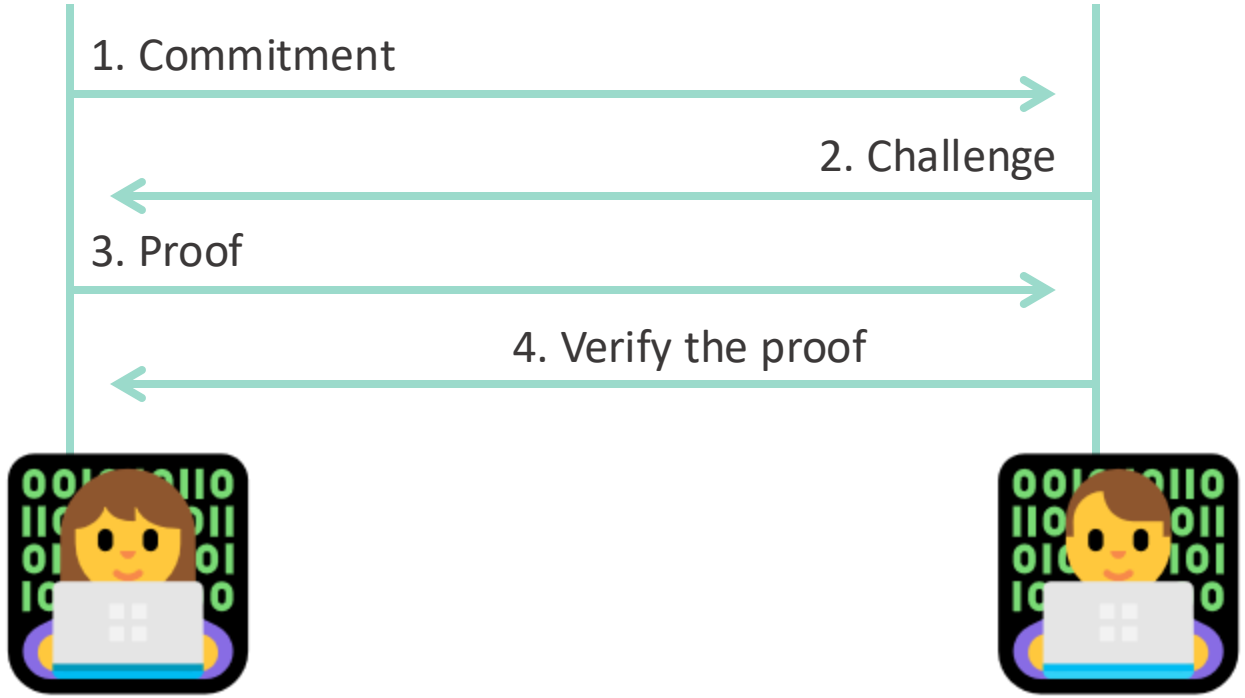
- Introduction
- Sigma Protocol
- Non-interactive proof

Zero-Knowledge Proof

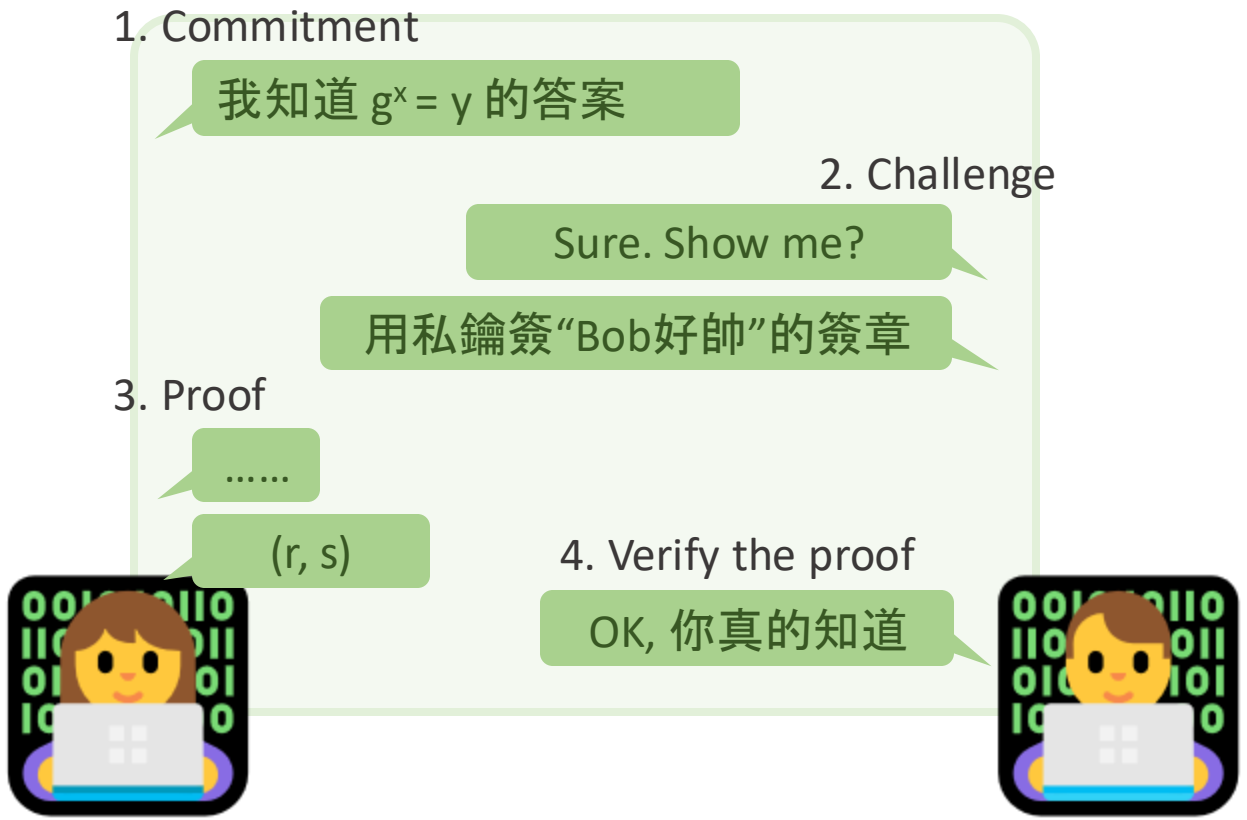


Zero-Knowledge Proof (Sigma protocol)

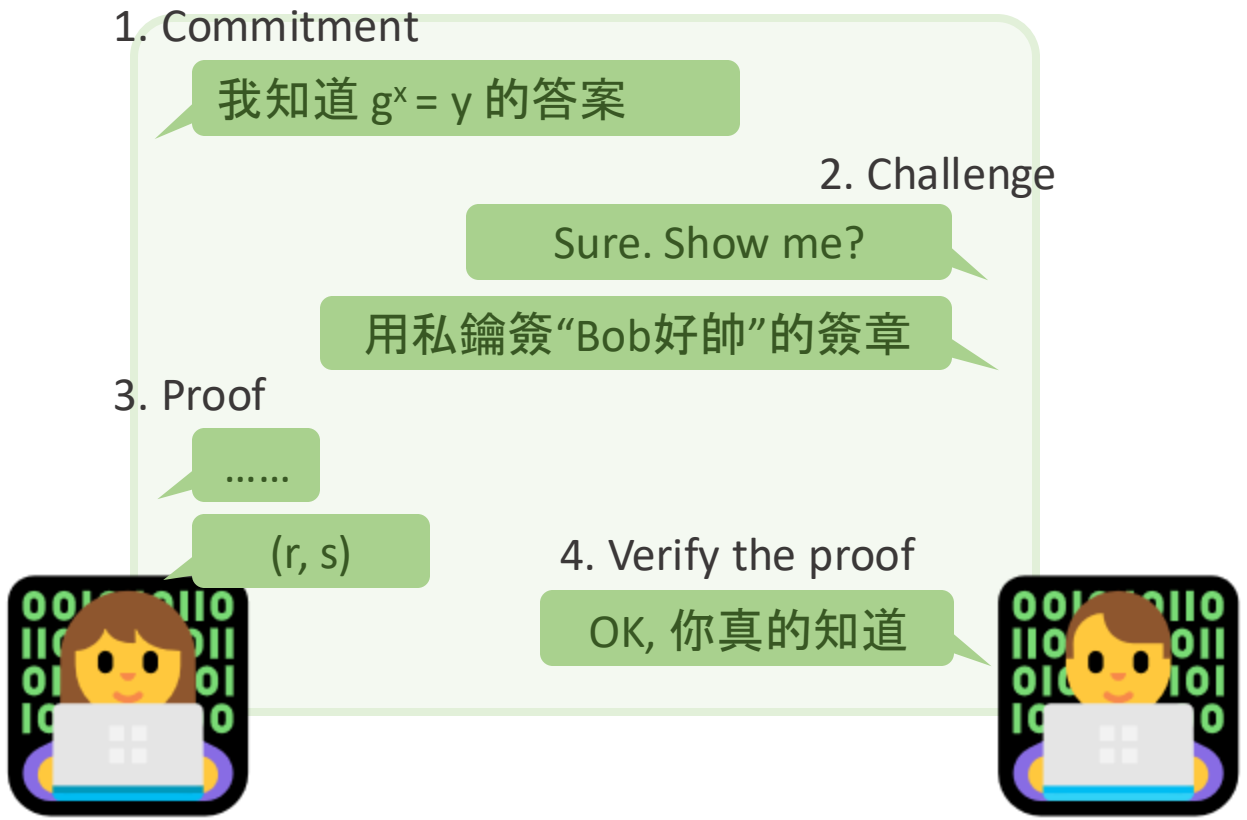
Proof of knowledge – Sigma protocol



Zero-Knowledge Proof (Sigma protocol)



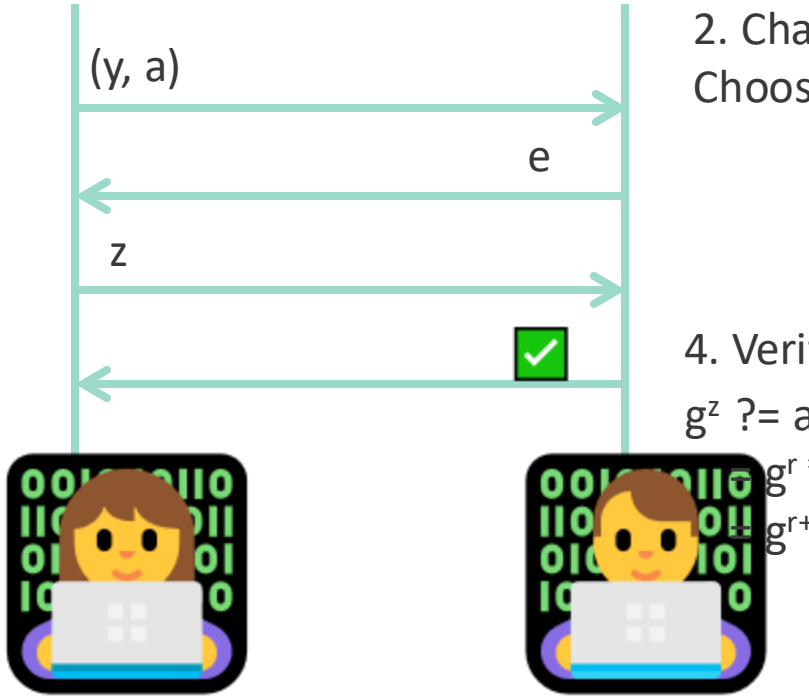
Zero-Knowledge Proof (Sigma protocol)



Zero-Knowledge Proof (Sigma protocol)

1. Commitment
$g^x=y$.
Choose a random r.
Compute $g^r=a$.

3. Proof
$z = r + e * x$



2. Challenge
Choose a random e

4. Verify the proof
g^z ?= $a * y^e$
$g^r * g^{x*e}$
g^{r+x*e}

Zero-Knowledge Proof (Non-interactive proof)

Fiat–Shamir transform

Turn an interactive protocol into a non-interactive one using a random oracle (in this case, the hash function).

1. Commitment
$g^x=y$.
Choose a random r.
Compute $g^r=a$.

2. Challenge
e = Hash(g, y, a)

3. Proof
z = r + e * x

4. Verify the proof
g, y, a
e = Hash(g, y, a)
g^z ?= a * y^e
$= g^r * g^{x*e}$
$= g^{r+x*e}$

Zero-Knowledge Proof (Non-interactive proof)

Fiat–Shamir transform

Turn an interactive protocol into a non-interactive one using a random oracle (in this case, the hash function).

1. Commitment
$g^x=y$.
Choose a random r.
Compute $g^r=a$.

2. Challenge
e = Hash(g, y, a)

3. Proof
z = r + e * x

因為 Hash 不能控制
需要先決定 random r 才能知道 e

Primitives

Other primitives in Cryptography

1. Hash function
2. Digital Signature
3. Zero-Knowledge Proof

Questions?

Advanced

Advanced Topic(?)

1. Elliptic Curve
2. Lattice

Advanced

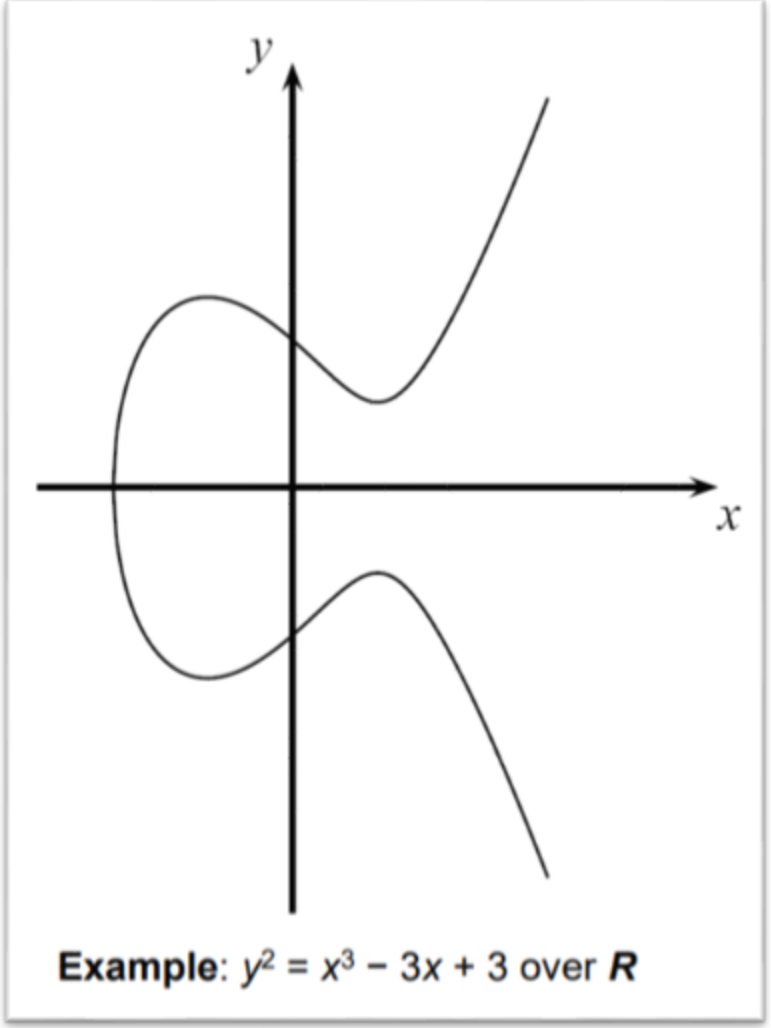
Elliptic Curve

- Introduction
- ECDLP
  - ECDH
  - ECDSA
- Attack on Elliptic Curve Cryptography

Elliptic Curve

xy 平面上的 Weierstrass‘s elliptic functions:

$y^2 = x^3 + ax + b$ $(4a^3 + 27b^2 \neq 0)$



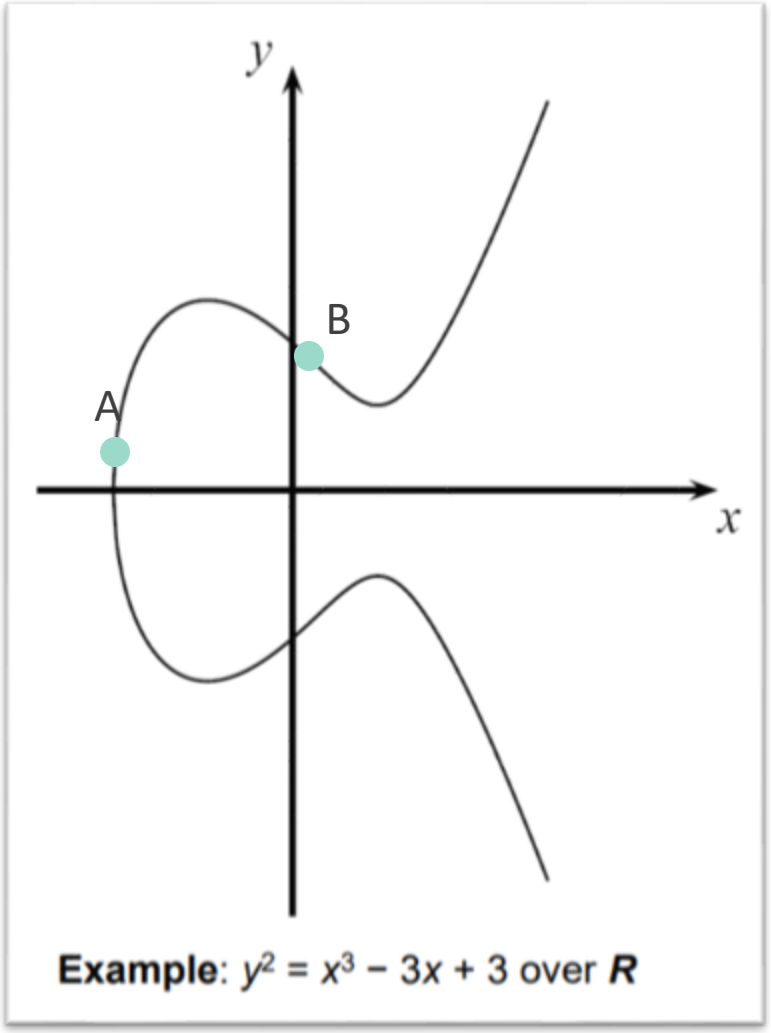
Elliptic Curve

xy 平面上的 Weierstrass‘s elliptic functions:

$y^2 = x^3 + ax + b$ $(4a^3 + 27b^2 \neq 0)$

Operation +:

A + B = C



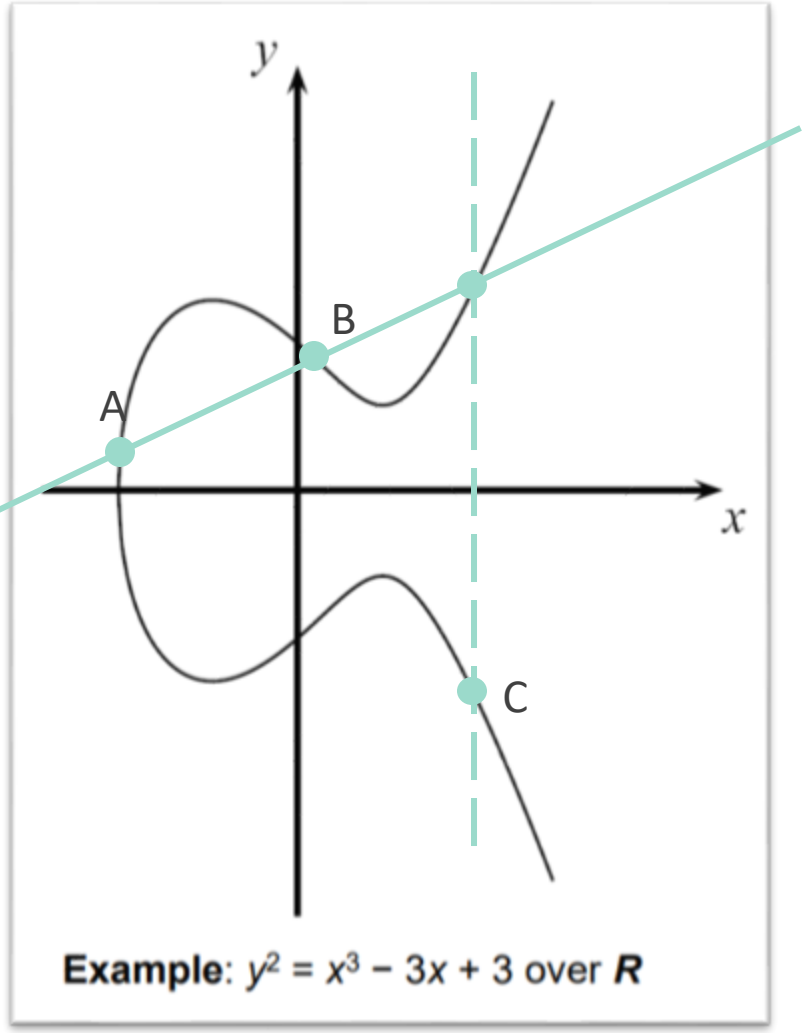
Elliptic Curve

xy 平面上的 Weierstrass‘s elliptic functions:

$y^2 = x^3 + ax + b$ $(4a^3 + 27b^2 \neq 0)$

Operation +:

A + B = C



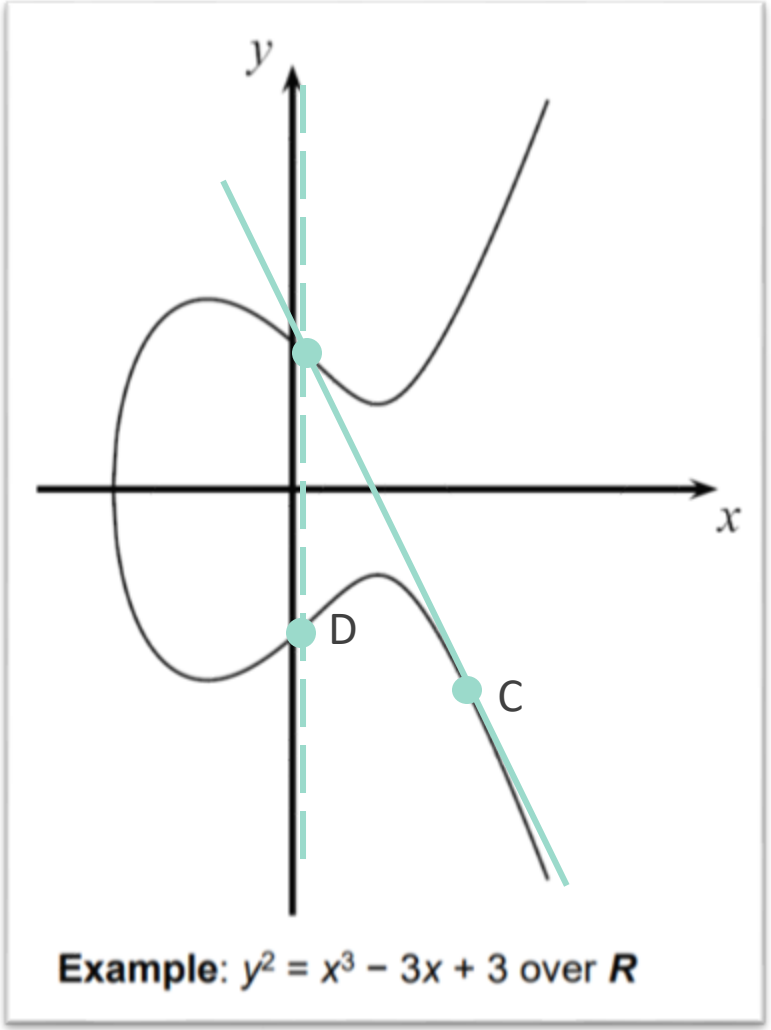
Elliptic Curve

xy 平面上的 Weierstrass‘s elliptic functions:

$y^2 = x^3 + ax + b$ $(4a^3 + 27b^2 \neq 0)$

Operation *:

2*C = C + C = D



Elliptic Curve

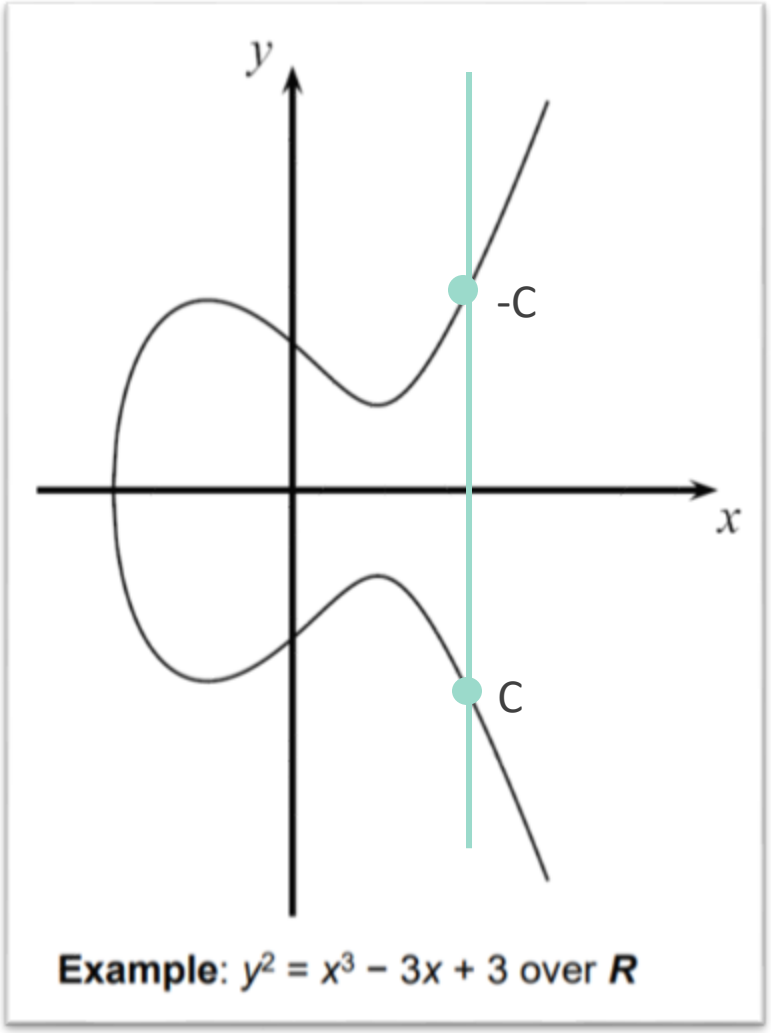
xy 平面上的 Weierstrass‘s elliptic functions:

$y^2 = x^3 + ax + b$ $(4a^3 + 27b^2 \neq 0)$

無窮遠點:

C + (-C) = 0

C + 0 = C



Elliptic Curve

xy 平面上的 Weierstrass‘s elliptic functions:

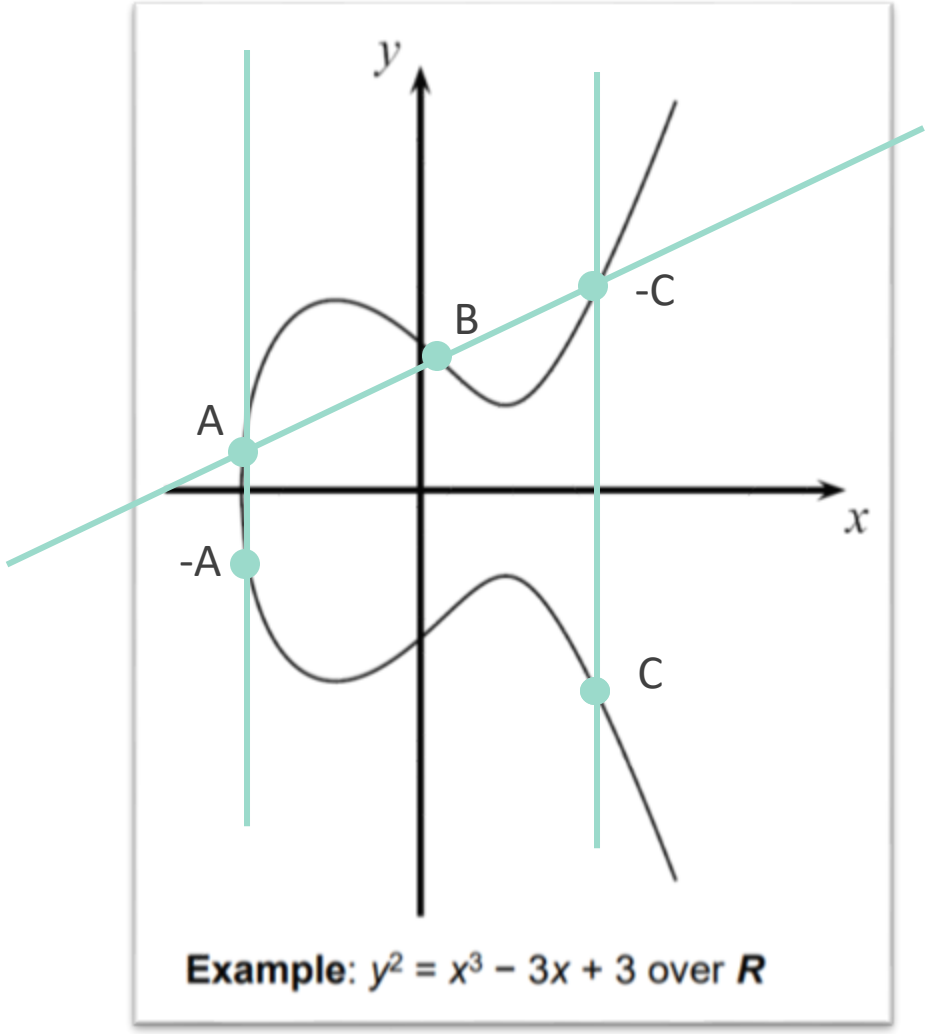
$y^2 = x^3 + ax + b$ $(4a^3 + 27b^2 \neq 0)$

為什麼這樣定義？-> 結合律

A + B = C

(A + B) + -C = 0

A + (B + -C) = 0



Elliptic Curve

xy 平面上的 Weierstrass's elliptic functions:

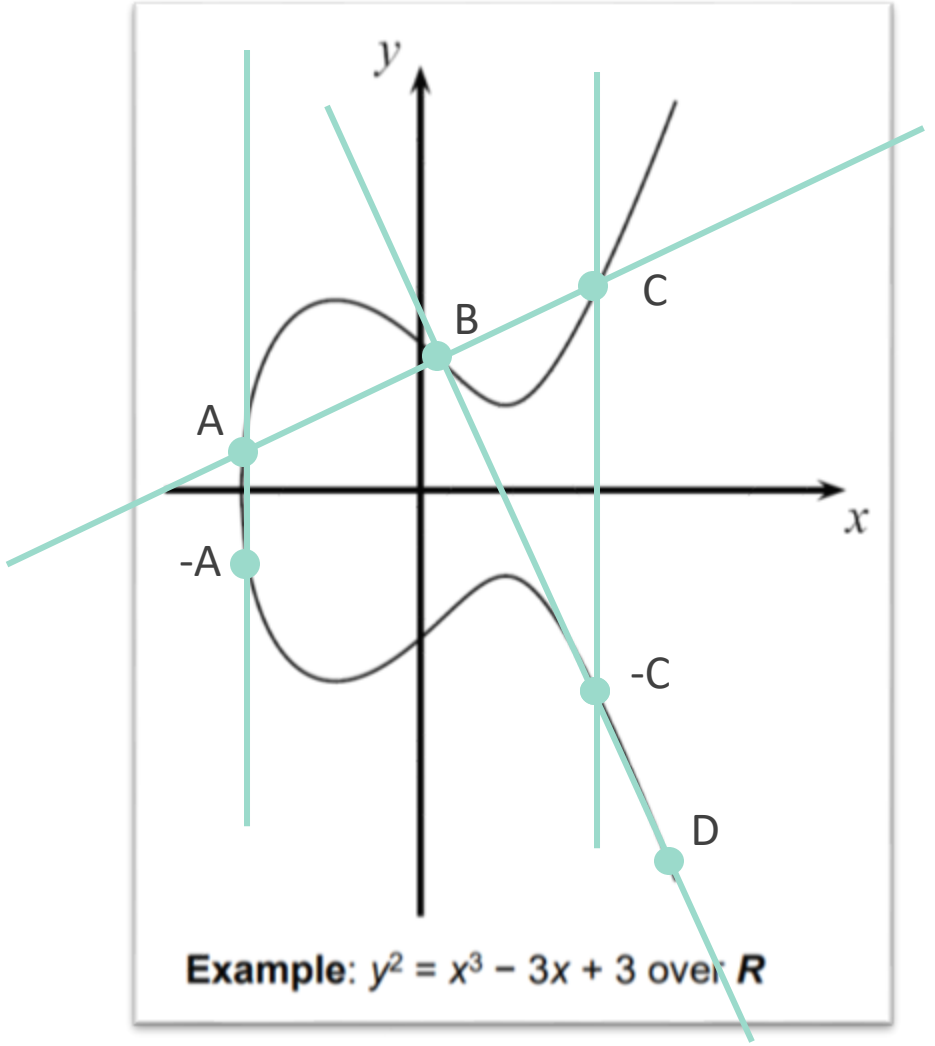
$y^2 = x^3 + ax + b$ $(4a^3 + 27b^2 \neq 0)$

為什麼這樣定義？-> 結合律

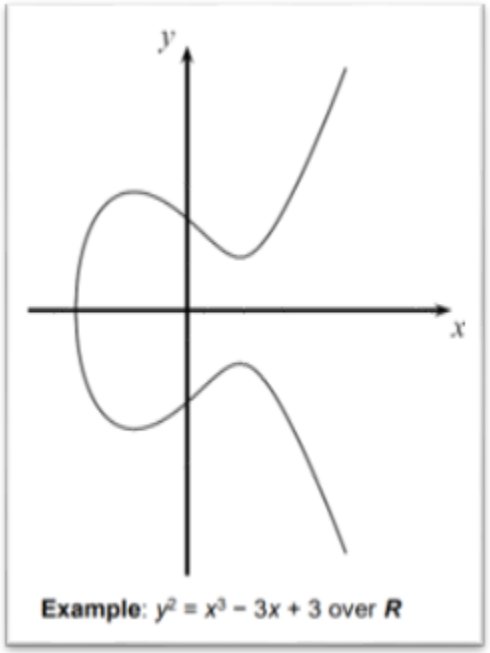
A + B = C

(A + B) + -C = 0 … OK

A + (B + -C) ≠ 0 … ?

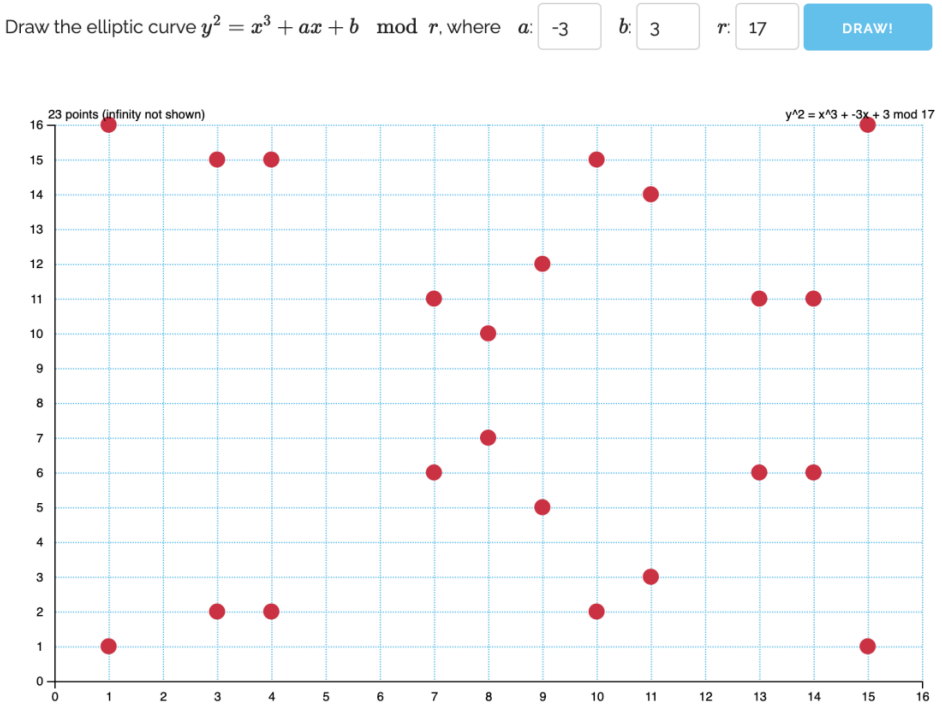


Elliptic Curve



xy 平面上的 Weierstrass's elliptic functions

$y^2 = x^3 + ax + b$ $(4a^3 + 27b^2 \neq 0)$



$y^2 = x^3 + ax + b \pmod{17}$

$4a^3 + 27b^2 \neq 0 \pmod{17}$

Elliptic Curve

$y^2 = x^3 + ax + b \ (4a^3 + 27b^2 \neq 0) \bmod p$

給定橢圓曲線 $E : y^2 = x^3 + ax + b$ 定義在質數有限域 $\mathbb{F}_p$ 上：

1. **斜率 s**：

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod p & (\text{當 } P_1 \neq P_2 \text{，點加法}) \\ \frac{3x_1^2 + a}{2y_1} \mod p & (\text{當 } P_1 = P_2 \text{，點倍加}) \end{cases}$$

2. **新點的座標 $P_3 = (x_3, y_3) = P_1 + P_2$**：

$$x_3 = s^2 - x_1 - x_2 \mod p$$

$$y_3 = s(x_1 - x_3) - y_1 \mod p$$

How to *x, for x > 2?

Elliptic Curve

$y^2 = x^3 + ax + b \ (4a^3 + 27b^2 \neq 0) \bmod p$

```python
def scalar_multiply(P, k, a, p):
    result = None  # 無窮遠點 O
    addend = P     # 初始加數為 P

    while k > 0:
        if k % 2 == 1:
            result = point_add(result, addend, a, p)  # 累加對應位元為 1 的 addend
        addend = point_double(addend, a, p)  # 每次都倍加
        k //= 2

    return result
```

Double-and-Add Algorithm

Elliptic Curve

$y^2 = x^3 + -3x + 3 \pmod{17}$ 就可以做成循環群

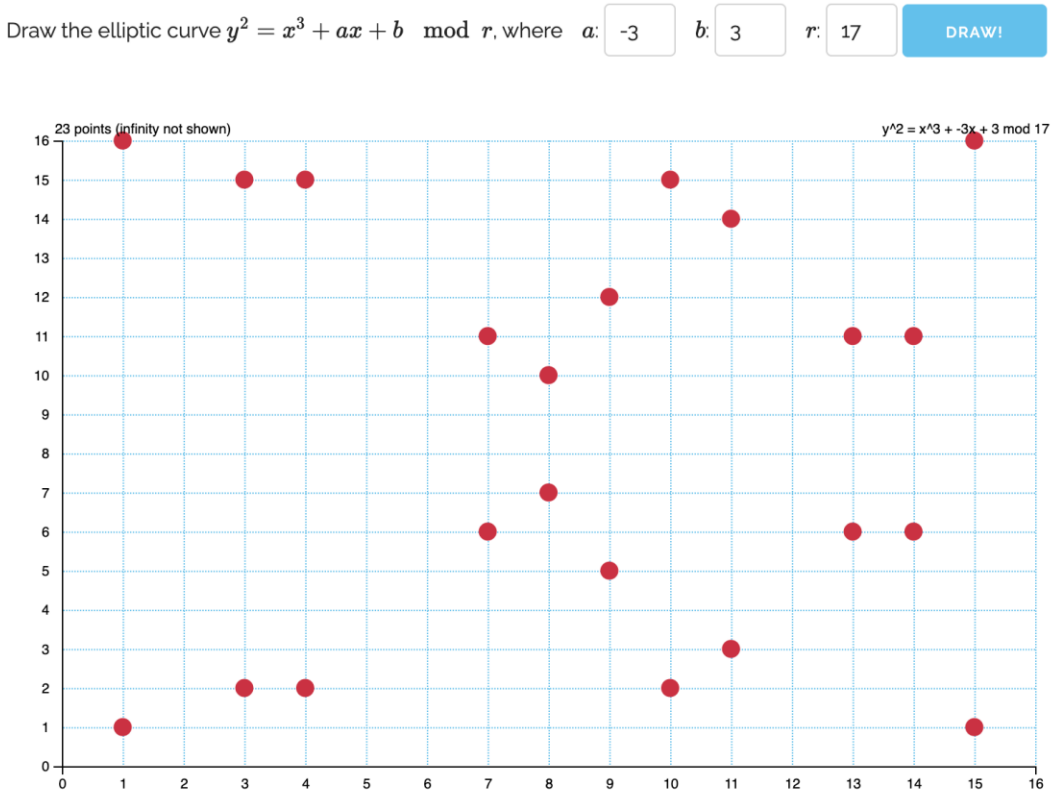
舉例來說：

選一點 P（8, 7），就可以遍歷上面所有點、得到一個 order 23 的 Group。

*order 會跟 (mod p) 的 p 很接近

Hasse's 定理: $|\#E(\mathbb{F}_p) - (p+1)| \leq 2\sqrt{p}$

P	2P	3P	…	22P	23P
(8 , 7)	(9, 5)	(4, 2)	…	(8, 10)	∞



Elliptic Curve

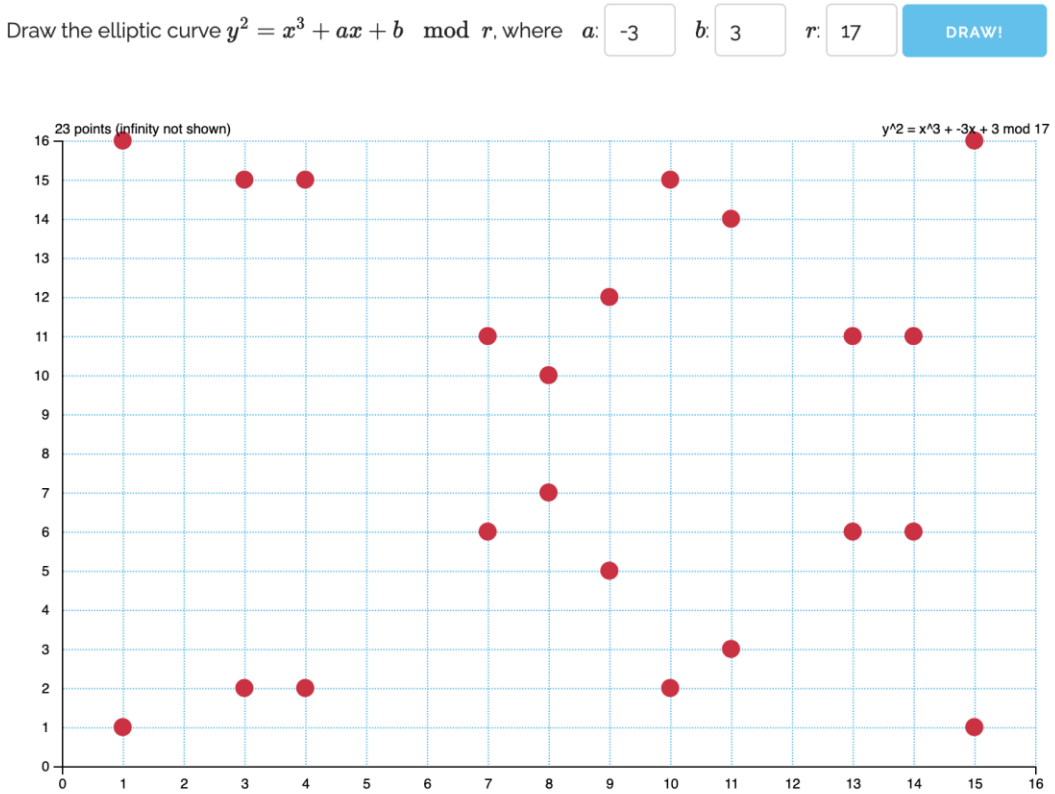
$y^2 = x^3 + -3x + 3 \pmod{17}$ 就可以做成循環群

舉例來說：

選一點 P（8, 7），就可以遍歷上面所有點、得到一個 order 23 的 Group。

```
sage: E = EllipticCurve(Integers(17), (-3, 3))
sage: E(8, 7).order()
23
```

P	2P	3P	…	22P	23P
(8 , 7)	(9, 5)	(4, 2)	…	(8, 10)	∞



Elliptic Curve – ECDLP

Elliptic Curve Discrete Logarithm Problem

Given a point P and a point Q on the curve, find the k of Q = kP.

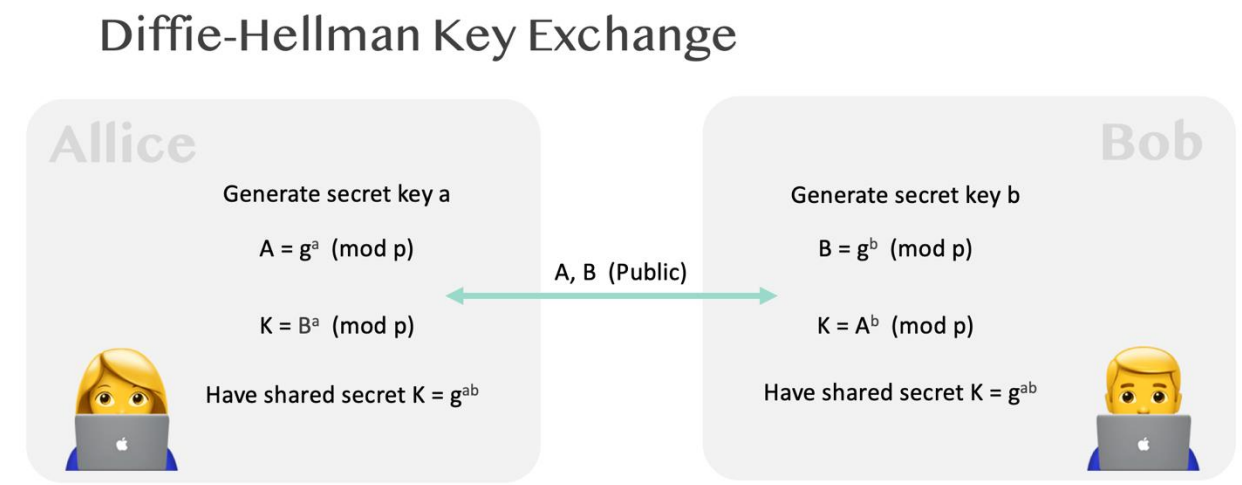
$g^x = y \pmod p \rightarrow kP = Q \pmod p$

Advanced

Elliptic Curve – ECDH

Elliptic Curve Diffie-Hellman Key Exchange

Asymmetric



Asymmetric

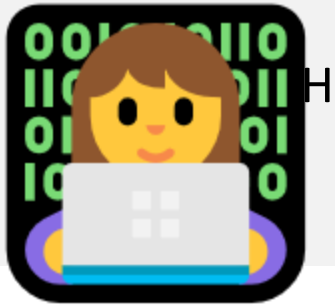
ECDH Key Exchange

Allice

Generate secret key a

A = aG (mod p)

K = aB (mod p)



Have shared secret K = ab*G

A, B (Public)

Bob

Generate secret key b

B = bG (mod p)

K = bA (mod p)

Have shared secret K = ab*G

Elliptic Curve – ECDSA

Elliptic Curve Digital Signature Algorithm

Primitives

Digital Signature - DSA (Digital Signature Algorithm)

Key Generation DSA 早期的建議位元數，後來有變多

1. Choose prime p (1024 bits), q (160 bits), 且 (p-1) % q = 0
2. Choose random h from [2, p-2],let g = $h^{(p-1)/q}$ (mod p) (直接 h = 2 比較方便)
3. Choose random x from [1, p-1], let y = g^x (mod p) (Public: y; Private: x)

Signing

1. Choose random k from [1, q-1]
2. Compute r = (g^k mod p) mod q, s = (k^{-1}(*message* + x * r)) mod q （訊息的 Hash 值）
3. Signature = (r, s)

91

Elliptic Curve – ECDSA

Key Generation

1. Choose curve E, a generator G on E, and prime p.
2. Choose random d from [1, p-1], let Q = dG (mod p) (Public: Q; Private: d)

Signing

1. Choose a random k from [1, p-1]
2. Compute (x, y) = kG (mod p), let r = x
3. Compute $s = k^{-1}(message + d * r) \pmod p$
4. Signature = (r, s)

Elliptic Curve – ECDSA

Verifying

1. Check $0 < r < p$, $0 < s < p$
2. Let $u1 = message * s^{-1} \pmod p$, $u2 = s^{-1} * r \pmod q$
3. Compute $(x, y) = u1 * G + u2 * Q \pmod p$
4. Check $x == r$?

Correctness

$u1 * G + u2 * Q$

$= (u1 + d*u2)\ G = s^{-1}(message + d * r)\ G = kG$

Elliptic Curve

跟 dlog 那邊介紹的一樣，安全性立基於 ECDLP 的難度，所以需要確保 Curve 跟 Generator 的選擇。

1. Smooth order：Pohlig Hellman
2. Singular curve：Node、Cusp，被轉成 DLP
3. Supersingular curve：MOV Attack，被轉成 DLP
4. Anomalous curve：Smart attack，被轉成 DLP

好的做法是用大家推薦的 Curve：https://neuromancer.sk/std/

Elliptic Curve -- Invalid Curve Attack

$y^2 = x^3 + ax + b \ (4a^3 + 27b^2 \neq 0) \bmod p$

給定橢圓曲線 $E : y^2 = x^3 + ax + b$ 定義在質數有限域 $\mathbb{F}_p$ 上：

1. 斜率 s：

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod p & (\text{當 } P_1 \neq P_2 \text{，點加法}) \\ \frac{3x_1^2 + a}{2y_1} \mod p & (\text{當 } P_1 = P_2 \text{，點倍加}) \end{cases}$$

2. 新點的座標 $P_3 = (x_3, y_3) = P_1 + P_2$：

$$x_3 = s^2 - x_1 - x_2 \mod p$$

$$y_3 = s(x_1 - x_3) - y_1 \mod p$$

式子中沒有用到 b！

Elliptic Curve – other applications

- Elliptic Curve Pairing
- Post-quantum Cryptography – isogeny-based Cryptography

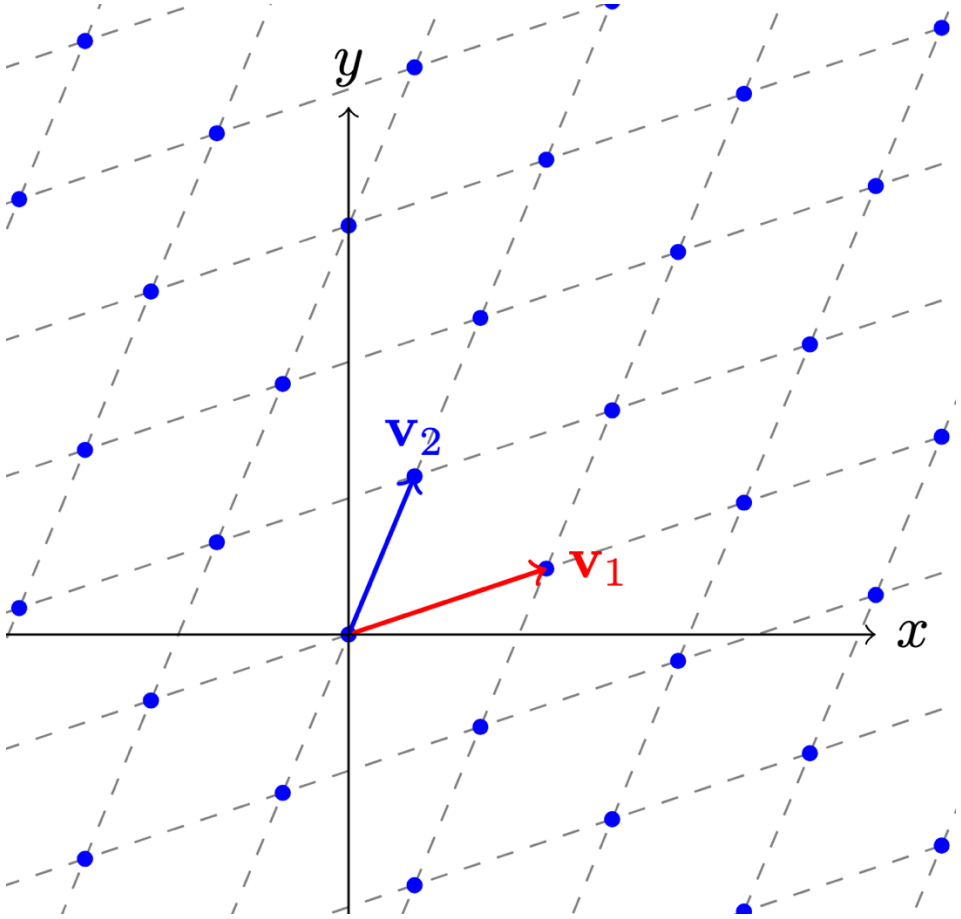
Advanced

Lattice

- Introduction
- Shortest Vector Problem (SVP)
  - LLL
  - Coppersmith method
- Closest Vector Problem (CVP)

Lattice

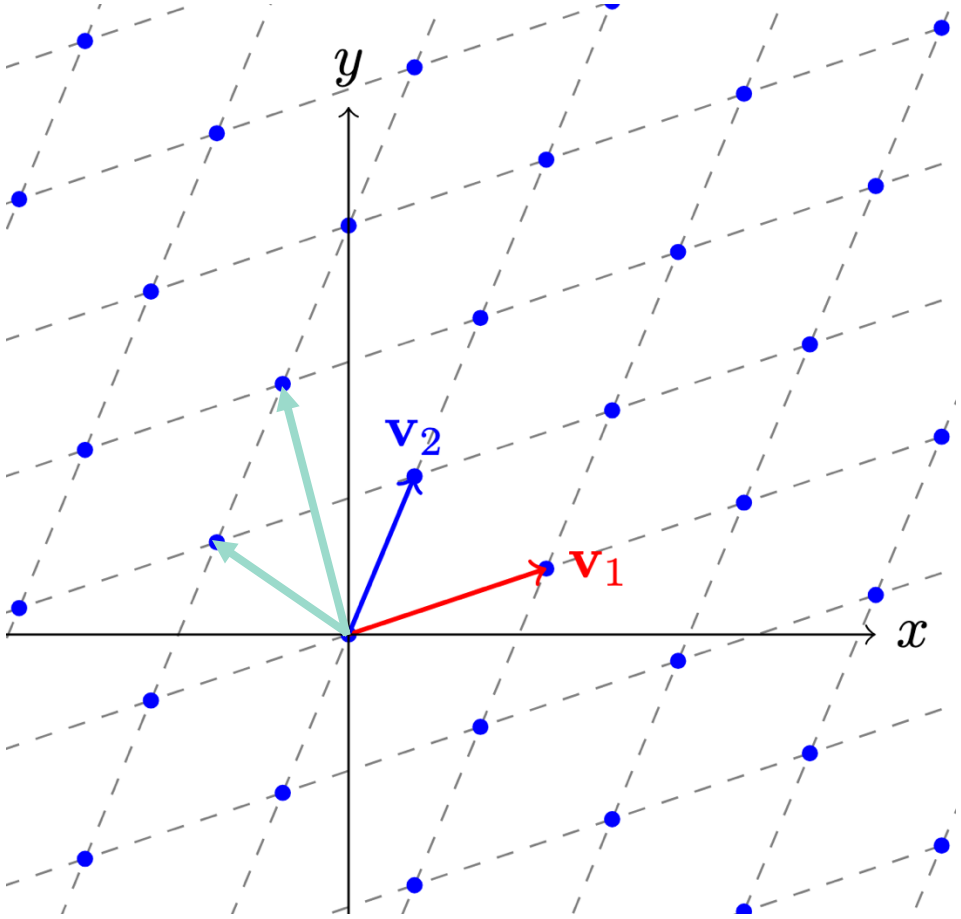
The set of all integer linear combinations of a set of linearly independent vectors.



Lattice

The set of all integer linear combinations of a set of linearly independent vectors.

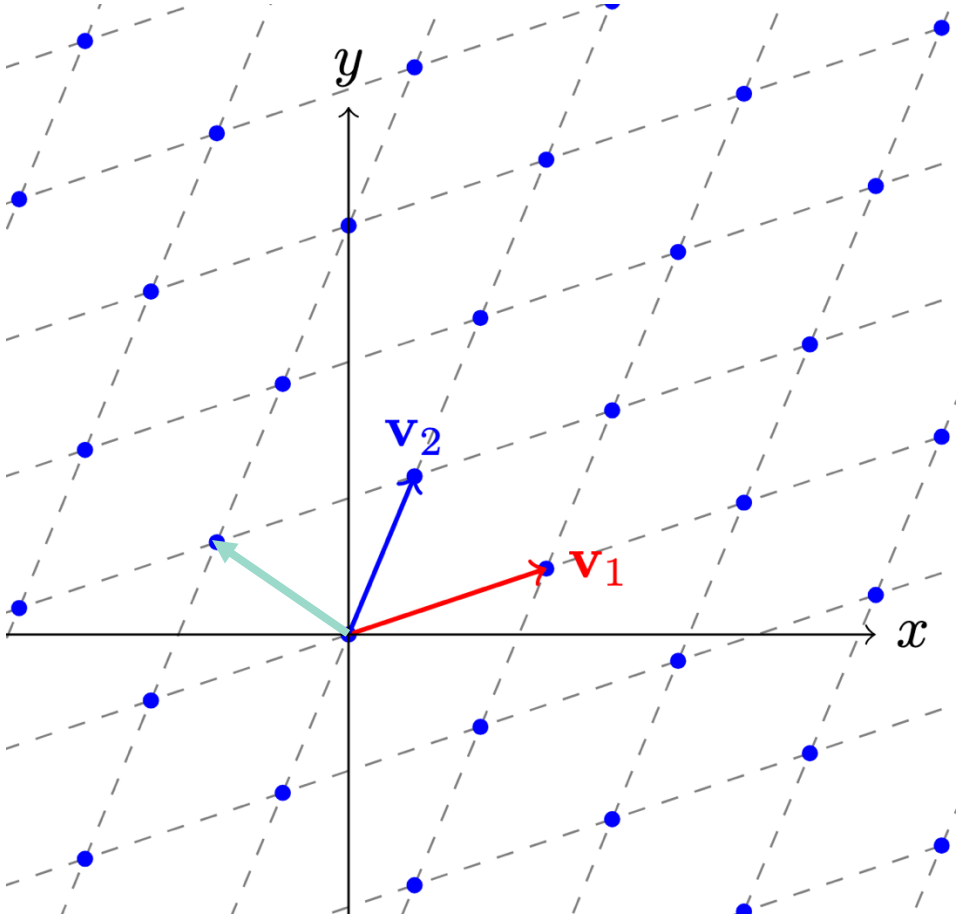
A Basis can only generate a lattice, while different bases may have the same lattice.



Lattice-- SVP (Shortest vector problem)

Given a basis, find the shortest vector

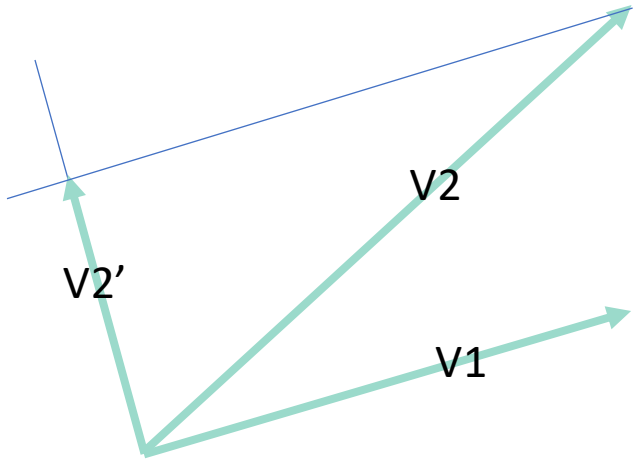
in the lattice of this basis.



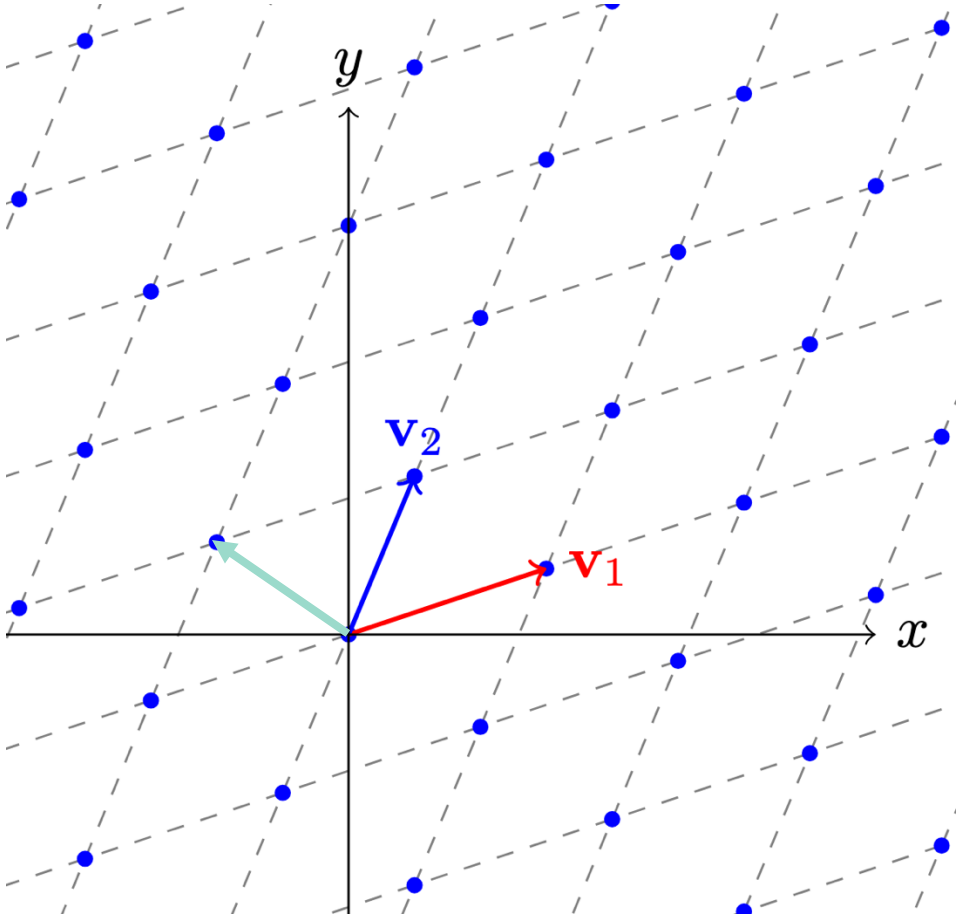
Lattice-- SVP (Shortest vector problem)

Given a basis, find the shortest vector

in the lattice of this basis.

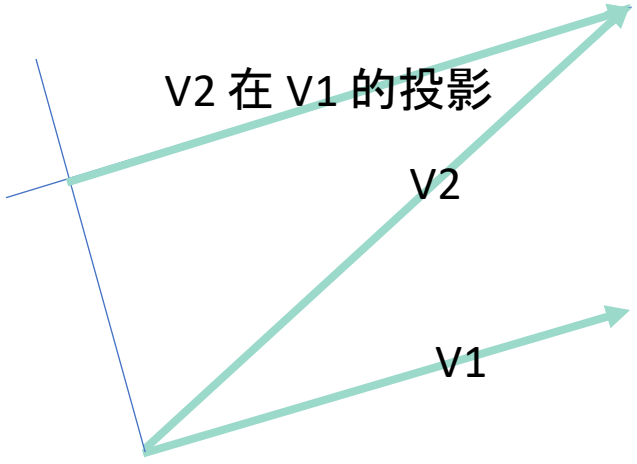


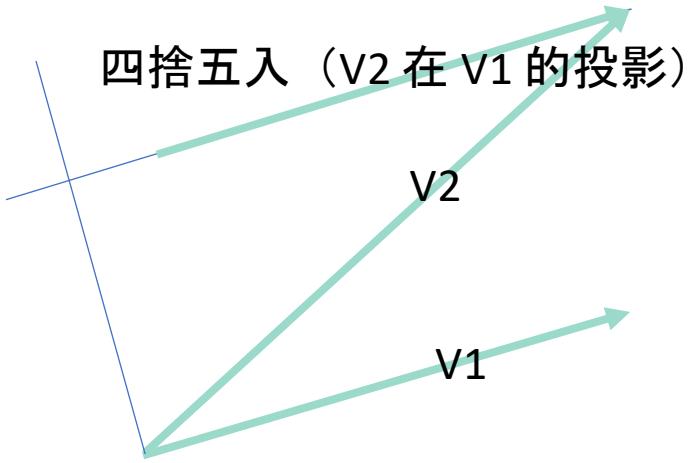
V2’ = V2 減掉 「V2 在 V1 上的投影」

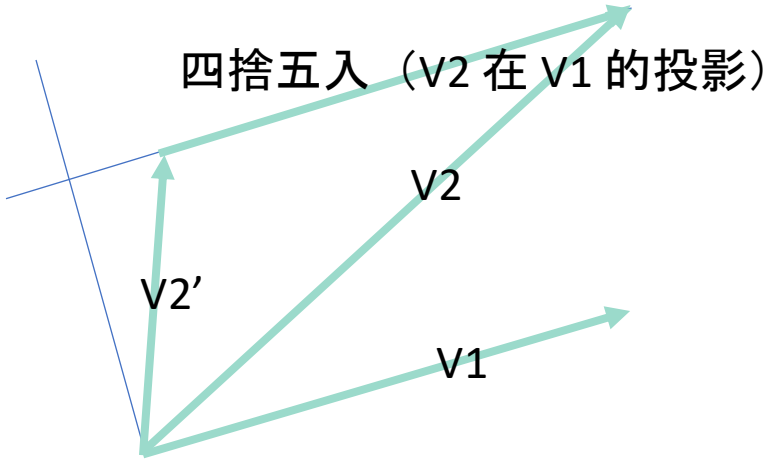


Lattice-- SVP (Shortest vector problem)

Given a basis, find the shortest vector in the lattice of this basis.







Lattice

LLL Algorithm

```
INPUT
    a lattice basis b_1, b_2, ..., b_n in Z^m
    a parameter δ with 1/4 < δ < 1, most commonly δ = 3/4
PROCEDURE
    B* <- GramSchmidt({b_1, ..., b_n}) = {b_1*, ..., b_n*};  and do not normalize
    μ_{i,j} <- InnerProduct(b_i, b_j*)/InnerProduct(b_j*, b_j*);   using the most current values of b_i and b_j*
    k <- 2;
    while k <= n do
        for j from k-1 to 1 do
            if |μ_{k,j}| > 1/2 then
                b_k <- b_k - ⌊μ_{k,j}⌉b_j;
               Update B* and the related μ_{i,j}'s as needed.
               (The naive method is to recompute B* whenever b_i changes:
                B* <- GramSchmidt({b_1, ..., b_n}) = {b_1*, ..., b_n*})
            end if
        end for
        if InnerProduct(b_k*, b_k*) > (δ - μ^2_{k,k-1}) InnerProduct(b_{k-1}*, b_{k-1}*) then
            k <- k + 1;
        else
            Swap b_k and  b_{k-1};
            Update B* and the related μ_{i,j}'s as needed.
            k <- max(k-1, 2);
        end if
    end while
    return B the LLL reduced basis of {b_1, ..., b_n}
OUTPUT
    the reduced basis b_1, b_2, ..., b_n in Z^m
```

https://en.wikipedia.org/wiki/Lenstra%E2%80%93Lenstra%E2%80%93Lov%C3%A1sz_lattice_basis_reduction_algorithm

Lattice -- LLL Algorithm

$$\begin{bmatrix} 1 & -1 & 3 \\ 1 & 0 & 5 \\ 1 & 2 & 6 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

1. A polynomial-time approximation algorithm.
2. Output an LLL-reduced basis
3. The length of the first (shortest) vector in the basis will be bounded by the determinant of the lattice.

$$\|\mathbf{b}_1\| \leq 2^{(n-1)/4} \cdot (\det(\mathcal{L}))^{1/n}$$

https://en.wikipedia.org/wiki/Lenstra%E2%80%93Lenstra%E2%80%93Lov%C3%A1sz_lattice_basis_reduction_algorithm

Lattice--Coppersmith Method

An algorithm for finding small roots of monic polynomials of degree d under modulo N.

$f(x) = a_0 + a_1x + a_2x^2 + \ldots + x^d \pmod N$

$f(r) = 0, r = ?$

1. 如果 N 是質數：一定有乘法反元素，這個問題就很好解
2. 如果 N 不是質數？

Lattice--Coppersmith Method

An algorithm for finding small roots of monic polynomials of degree d

under modulo N.

$f(x) = a_0 + a_1x + a_2x^2 + \ldots + x^d \pmod N$

$f(r) = 0$ && $r < R$, $r = ?$

1. 如果 N 是質數：一定有乘法反元素，這個問題就很好解
2. 如果 N 不是質數？

Lattice--Coppersmith Method

$f(x) = a_0 + a_1x + a_2x^2 + \ldots + x^d \pmod N$.

$f(r) = 0 \pmod N \;\&\&\; r < R,\ r = ?$

準備 d 個多項式，這些多項式在 mod N 底下都是 0。

$g_0(x) = N,\ g_1(x) = xN,\ g_2(x) = x^2N,\ \ldots\ldots,\ g_{d-1}(x) = x^{d-1}N$

把這些多項式做一個線性組合：

$$Q(x) = t_df(x) + t_0g_0(x) + t_1g_1(x) + t_2g_2(x) + \ldots\ldots + t_{d-1}g_{d-1}(x) \pmod N$$

$$= (t_da_0+t_0N) + (t_da_1+t_1N)x + (t_da_2+t_2N)x^2 + \ldots\ldots + (t_{d-1}a_{d-1}+t_{d-1}N)x^{d-1} + t_dx^d$$

$\Rightarrow\ Q(r) = 0 \pmod N$

Lattice--Coppersmith Method

$Q(x) = t_d f(x) + t_0 g_0(x) + t_1 g_1(x) + t_2 g_2(x) + \ldots\ldots + t_{d-1} g_{d-1}(x) \pmod N$

$= (t_d a_0 + t_0 N) + (t_d a_1 + t_1 N)x + (t_d a_2 + t_2 N)x^2 + \ldots\ldots + (t_{d-1} a_{d-1} + t_{d-1} N)x^{d-1} + t_d x^d$

$\Rightarrow Q(r) = 0 \pmod N$

如果我們可以讓 Q(x) 超小，小到對於所有 $x < R$ 能夠滿足 $|Q(x)| < N$

$|Q(x)| < |Q(R)| = |t_d a_0 + t_0 N| + |t_d a_1 + t_1 N|R + |t_d a_2 + t_2 N|R^2 + \ldots\ldots + |t_{d-1} a_{d-1} + t_{d-1} N|R^{d-1} + |t_d|R^d < N$

$\Rightarrow Q(r) = 0 \pmod N, |Q(R)| < N, r < R$

$\Rightarrow Q(r) = 0$ （可以直接在整數上面找，沒有 mod 就可以堪根之類的很簡單）

Lattice--Coppersmith Method

如果我們可以讓 Q(x) 超小，小到對於所有 $x < R$ 能夠滿足 $|Q(x)| < N$

$f(R) = a_0 + a_1R + a_2R^2 + \ldots + R^d \pmod N$.

$g_0(R) = N$, $g_1(R) = RN$, $g_2(R) = R^2N$, ……, $g_{d-1}(R) = R^{d-1}N$

$\Rightarrow (a_0, a_1R, a_2R^2, \ldots, R^d)$

$\Rightarrow (N, 0, 0, \ldots, 0), (0, NR, 0, \ldots, 0), \ldots, (0, 0, 0, \ldots, NR^{d-1}, 0)$

Q(R) 可以被寫成這些向量的線性組和

⇒Q(R) 他在這些向量構成的 Basis 裡面！

⇒「|Q(R)| 盡量的小」 是一個 Shortest Vector Problem

Lattice--Coppersmith Method

如果我們可以讓 Q(x) 超小，小到對於所有 x < R 能夠滿足 |Q(R)| < N

$f(R) = a_0 + a_1R + a_2R^2 + \ldots + R^d \pmod N$.

$g_0(R) = N,\ g_1(R) = RN,\ g_2(R) = R^2N,\ \ldots\ldots,\ g_{d-1}(R) = R^{d-1}N$

$\Rightarrow\ (a_0, a_1R, a_2R^2, \ldots, R^d)$

$\Rightarrow\ (N, 0, 0, \ldots, 0), (0, NR, 0, \ldots, 0), \ldots, (0, 0, 0, \ldots, NR^{d-1}, 0)$

$$\begin{bmatrix} N & 0 & 0 & 0 & 0 & 0 \\ 0 & NR & 0 & 0 & 0 & 0 \\ 0 & 0 & NR^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots\cdots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & NR^{d-1} & 0 \\ a_0 & a_1R & a_2R^2 & \ldots & a_{d-1}R^{d-1} & R^d \end{bmatrix}$$

Q(R) 可以被寫成這些向量的線性組和

⇒Q(R) 他在這些向量構成的 Basis 裡面！

⇒「|Q(R)| 盡量的小」 是一個 Shortest Vector Problem

Lattice--Coppersmith Method

An algorithm for finding small roots of monic polynomials of degree d under modulo N.

$f(x) = a_0 + a_1x + a_2x^2 + \ldots + x^d \pmod N = 0$

$f(r) = 0$ && $|r| < R$, $r = ?$

1. 如果 N 是質數：一定有乘法反元素，這個問題就很好解
2. 如果 N 不是質數？ => find small root r, for $|r| < N^{1/d}$

Lattice--Coppersmith Method (Stereotyped Message Attack on RSA)

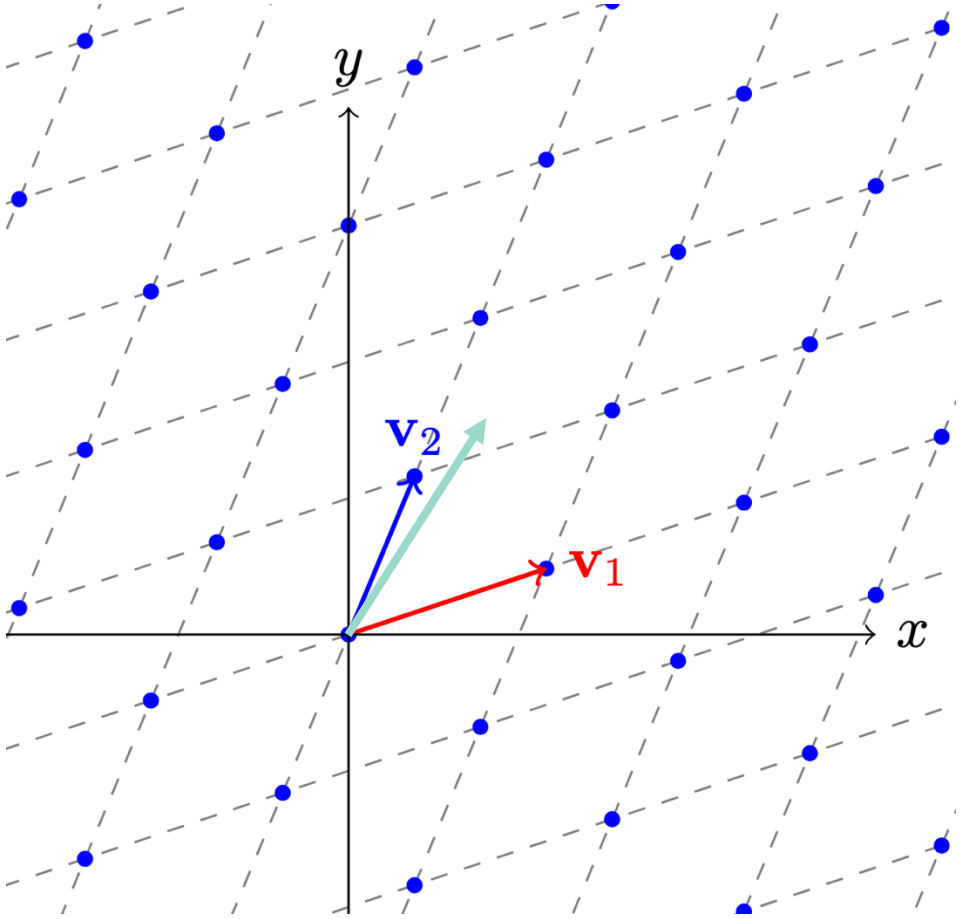
If the message has a known pattern: m = a + x.

Then, $f(x) = m^e - c = (a + x)^e - c \pmod n$ is a monic polynomial of degree e under modulo N, and we want to find a small root x.

```
sage: from Crypto.Util.number import long_to_bytes, bytes_to_long, getPrime
....: p, q = getPrime(1024), getPrime(1024)
....: e, N = 3, p * q
....: m =  2 ** 1000  + bytes_to_long(b"I Love LLL")
....: c = m ** e % N
....:
....: # --- RSA Stereotyped Message Attack ---
....: R.<x> = PolynomialRing(Integers(N))
....: f = (2 ** 1000 + x)^3 - c
....: f.small_roots()
[345328556810888809303116]
sage: print(long_to_bytes(345328556810888809303116))
b'I Love LLL'
sage:
```

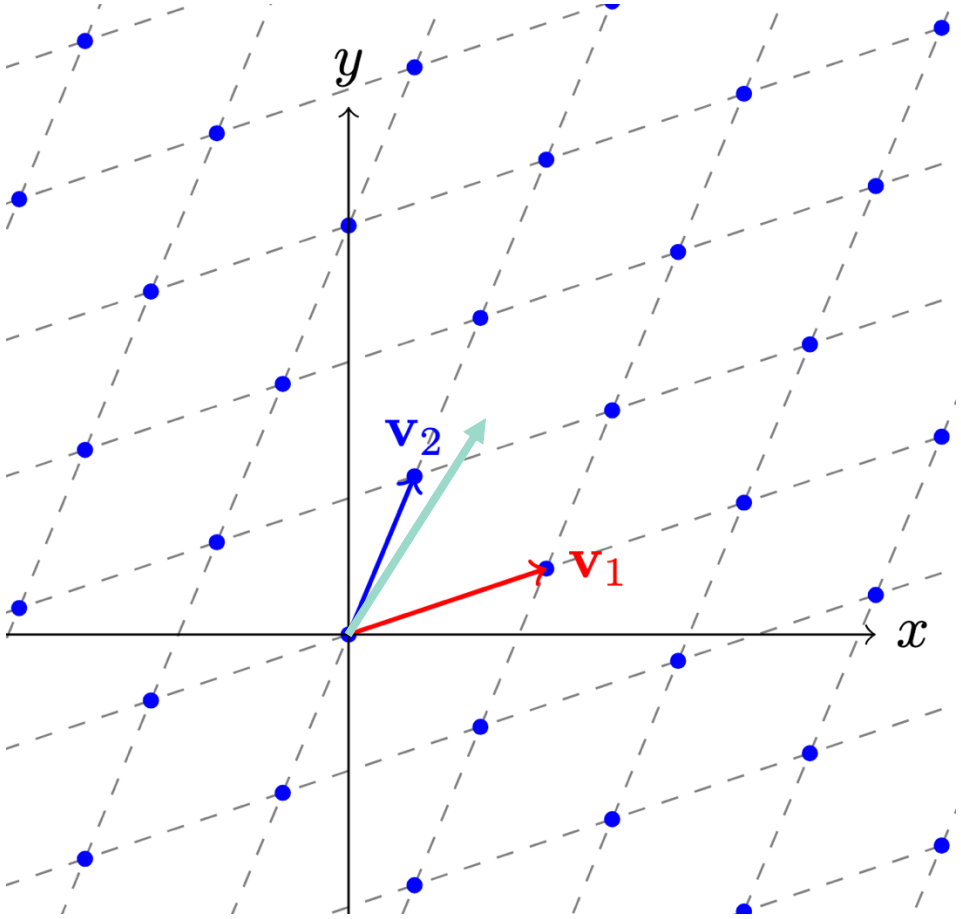
Lattice-- CVP (closest vector problem)

Given an arbitrary vector (no need to be in the basis), find the closest vector to this arbitrary vector in the lattice of this basis.



Lattice-- CVP (closest vector problem)

Given an arbitrary vector (no need to be in the basis), find the closest vector to this arbitrary vector in the lattice of this basis.



Babai's Nearest Plane Algorithm :

1. Reduce basis
2. For k in range(n):

   Fix the first k - 1 vectors in the basis.

   Greedily optimize the distance with the k-th vector.

Lattice

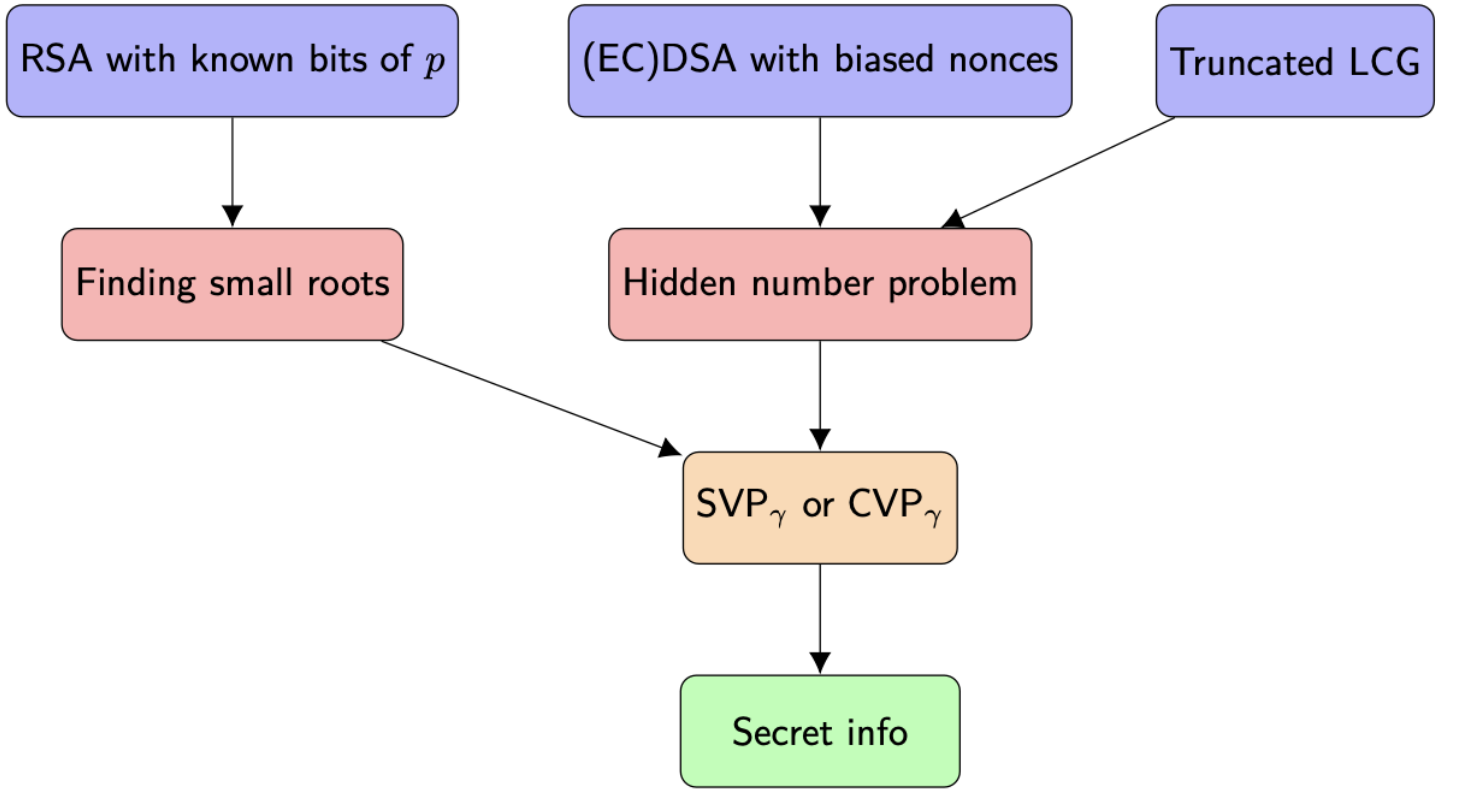


Figure 2: Examples of lattice-based attacks.

https://github.com/josephsurin/lattice-based-cryptanalysis/blob/main/tutorial.pdf

Advanced

Advanced Topic(?)

1. Elliptic Curve
2. Lattice

Questions?

回饋問卷表單

https://forms.gle/bStwQd5G7FUMK6kg6