Cryptography 3

Ice1187 2024/06/14

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

https://github.com/Ice1187

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！**

大綱

- 非對稱式密碼學的概念與應用
  - 金鑰交換
  - 數位簽章
  - 公鑰基礎建設 PKI
- RSA
- ElGamal

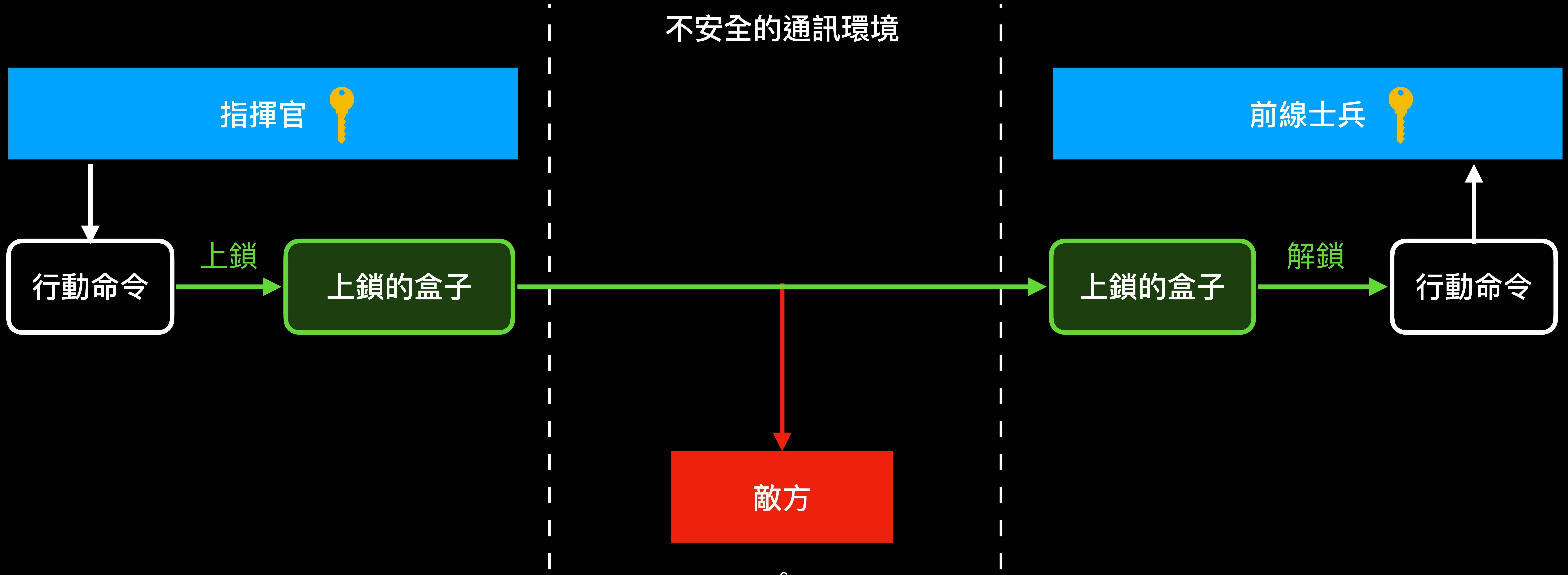
密碼學名詞與符號
Terminology

- 明文 plaintext / m：要傳遞的訊息
- 密文 ciphertext / c：經過加密後的明文
- 公鑰 public key / k_e：加解密時使用的鑰匙，公開供他人使用
- 私鑰 private key / k_d：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 encryption / E：將明文轉換成密文的程序，$E(m, k) = c$
- 解密 decryption / D：將密文轉換成明文的程序，$D(c, k) = m$

非對稱式加密的概念與應用

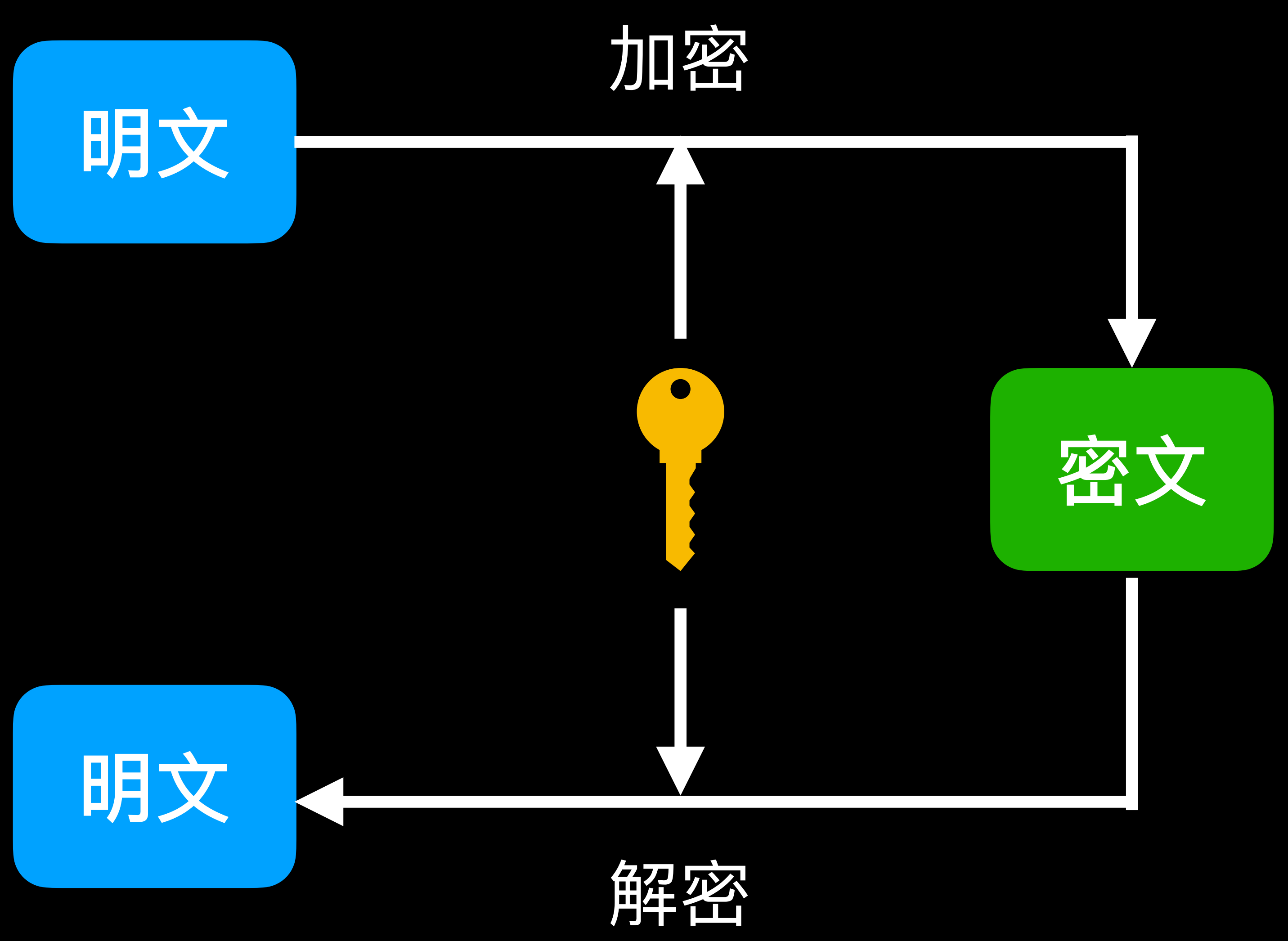
複習：對稱式加密

- 對稱：上鎖與解鎖使用相同的鑰匙



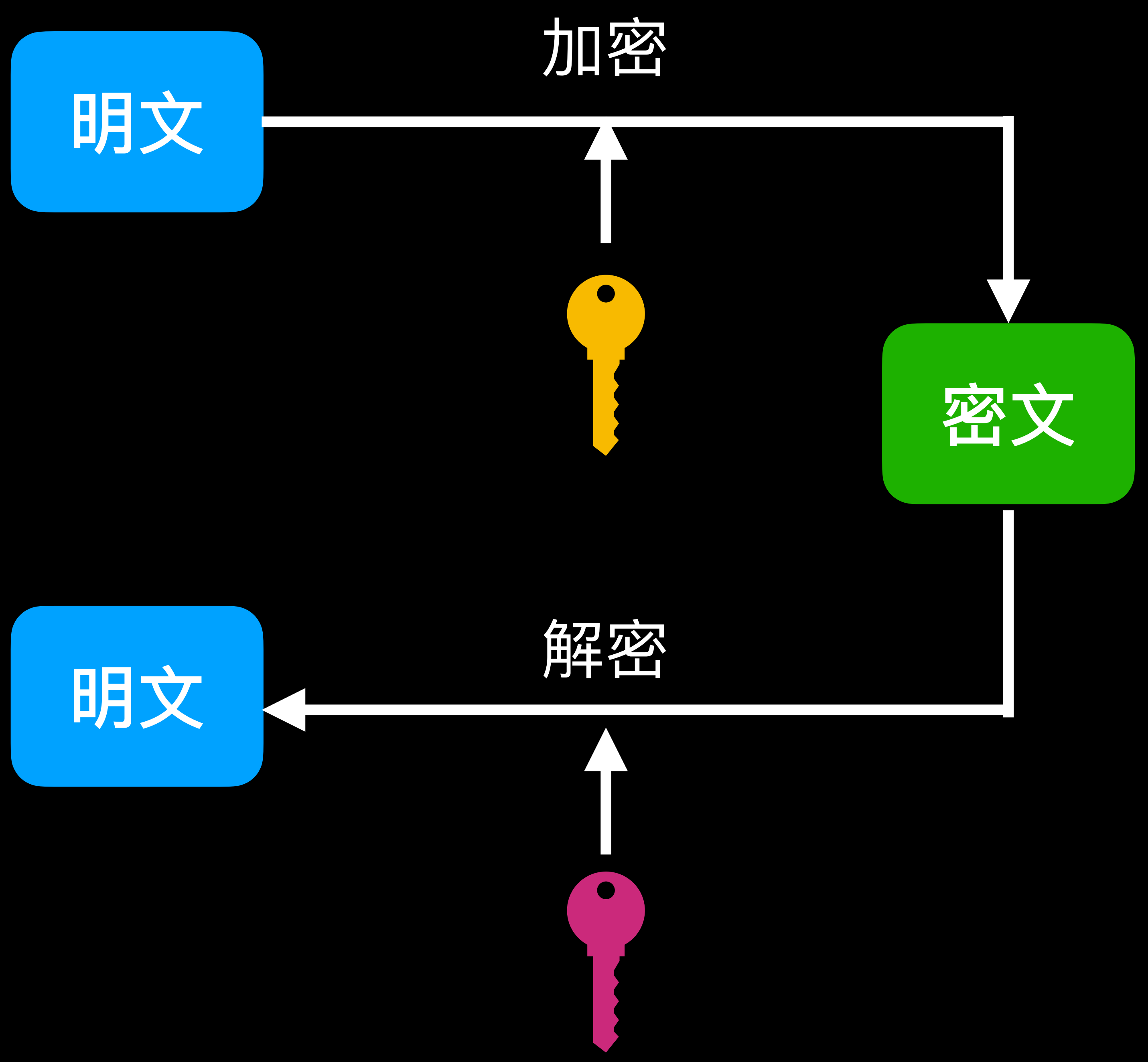
對稱式加密 v.s. 非對稱式加密

對稱式加密



加、解密使用相同金鑰

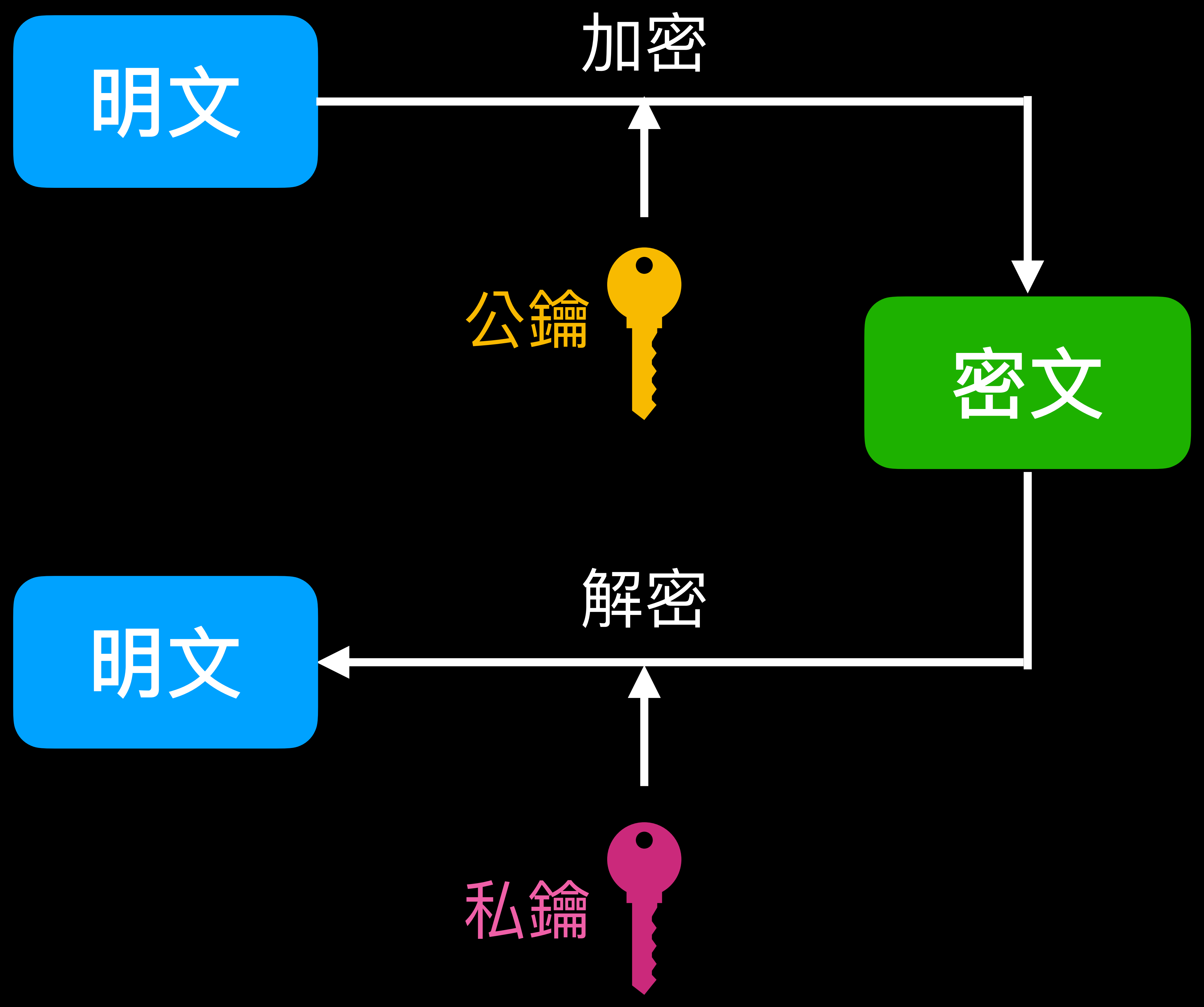
非對稱式加密



加、解密使用不同金鑰

非對稱式加密

- 一組彼此對應的公鑰、私鑰
  - 公鑰：大家都能取得，用於加密訊息
  - 私鑰：只有自己知道，用於解密訊息



Lab：Alice 如何傳訊息給 Bob？

- 桌前有三個人：Alice、Bob、老師
- Alice 有一張紙條 🏷️ 要傳給 Bob，內容不能讓老師看到
- Bob 有一個未上鎖 🔓 的盒子 📦 和鑰匙 🔑
    - 盒子不用鑰匙就能上鎖，但需要鑰匙才能解鎖
    - 鑰匙不能交給 Alice 或老師
- 老師可以檢視任何 Alice 和 Bob 傳遞的東西

解法：Alice 如何傳訊息給 Bob？

Bob	老師	Alice
🔑		🏷️
📦🔓		

解法：Alice 如何傳訊息給 Bob？

Bob

老師

Alice

解法：Alice 如何傳訊息給 Bob？

Bob 老師 Alice

🔑

🏷️📦🔓

解法：Alice 如何傳訊息給 Bob？

Bob　　　老師　　　Alice

🔑

🏷️📦🔒

解法：Alice 如何傳訊息給 Bob？

Bob　　　　老師　　　　Alice

解法：Alice 如何傳訊息給 Bob？

Bob

老師

Alice

解法：Alice 如何傳訊息給 Bob？

Bob　　　　老師　　　　Alice


私鑰：只有自己知道，用於解密訊息


公鑰：大家都能取得，用於加密訊息

非對稱式加密的概念

非對稱式加密的概念

1. Bob 生成一組公、私鑰

非對稱式加密的概念

2. Bob 將公鑰發佈到網路上

Alice

不安全的通訊環境

公鑰 k_e

Bob 私鑰 k_d

攻擊者

非對稱式加密的概念

3. Alice 取得公鑰

Alice 公鑰 k_e

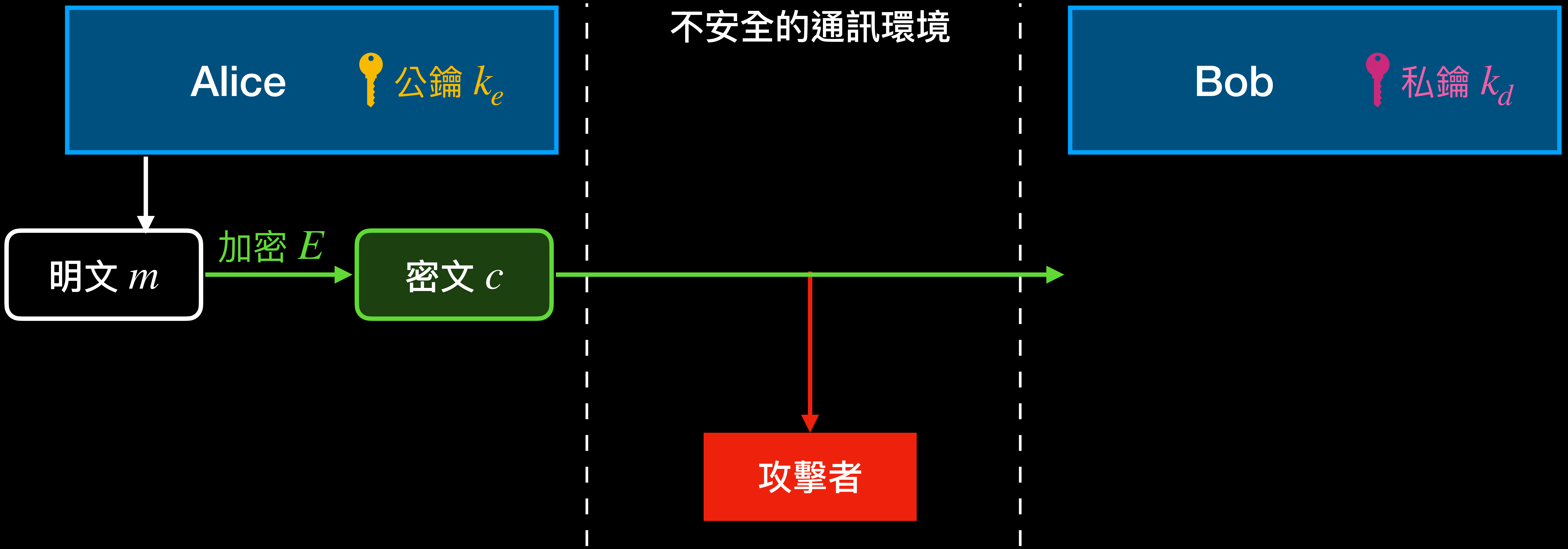
不安全的通訊環境

Bob 私鑰 k_d

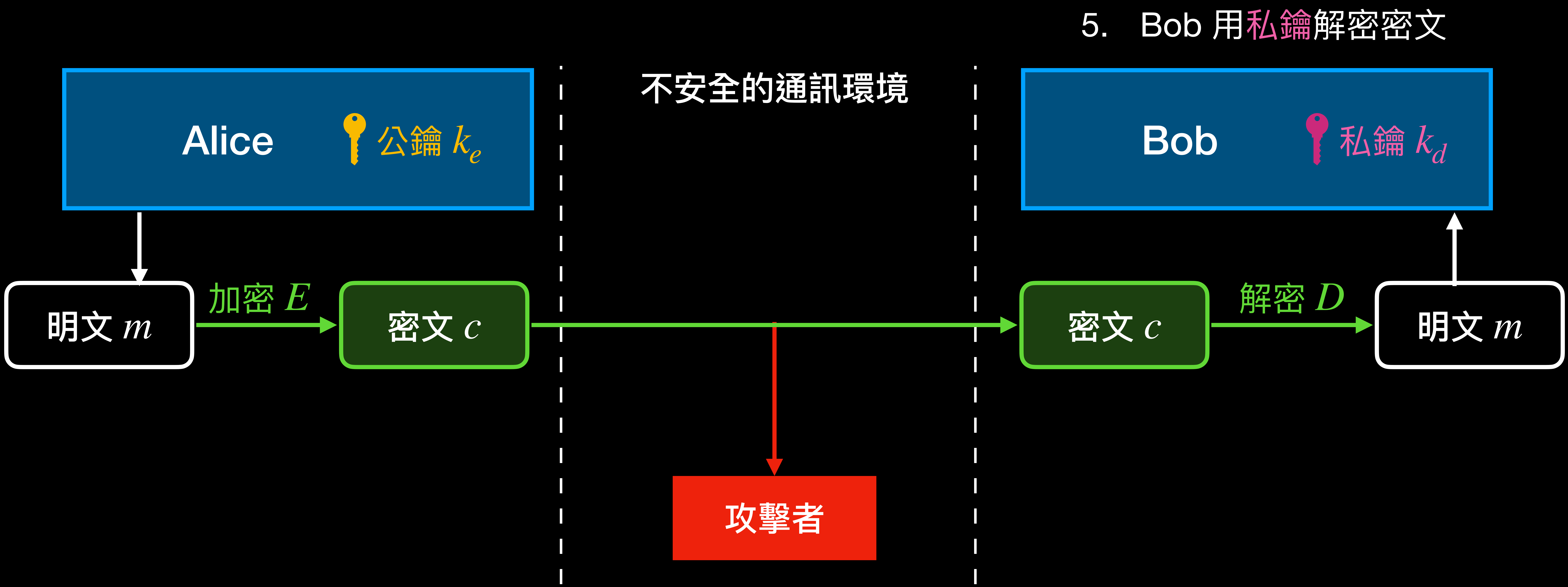
攻擊者

非對稱式加密的概念

4. Alice 用公鑰加密明文並傳送



非對稱式加密的概念


5. Bob 用私鑰解密密文
不安全的通訊環境
Alice
公鑰 k_e
Bob
私鑰 k_d
明文 m
加密 E
密文 c
密文 c
解密 D
明文 m
攻擊者

非對稱式加密簡介

- 也稱作「公開金鑰加密」，由 Diffie 與 Hellman 於 1976 年提出
- 現代密碼學最重要的發明
    - 金鑰交換
    - 數位簽章

金鑰交換
Key Exchange

- 金鑰交換：如何在不安全的網路環境中，安全地交換加密用的金鑰？
- 可擴展性：有很多人要溝通時，此方法仍需適用

金鑰交換 & 可擴展性：對稱式加密

- 金鑰交換
  - 物理上傳遞密鑰 ➡ 不實際
  - ~~使用 Diffie-Hellman 金鑰交換~~
- 擴展性差
  - n 個人彼此通訊，需要 $\dfrac{n(n-1)}{2}$ 密鑰

金鑰交換 & 可擴展性：非對稱式加密

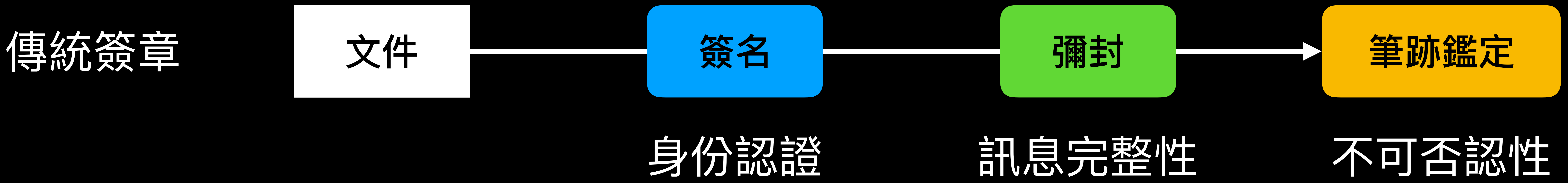
- 金鑰交換
  - 前面 Alice, Bob 與老師的例子 🏷️📦🔒🔑
- 擴展性佳
  - n 個人彼此通訊，需要 $2n$ 金鑰

金鑰交換：混合運用的加密系統

- 為什麼是金鑰交換，而不是訊息加密？
- 相對於對稱式加密，非對稱式需要更多計算資源與時間
- 混合運用的加密系統
  1. 使用非對稱式加密交換對稱式加密的密鑰
  2. 使用對稱式加密加密訊息

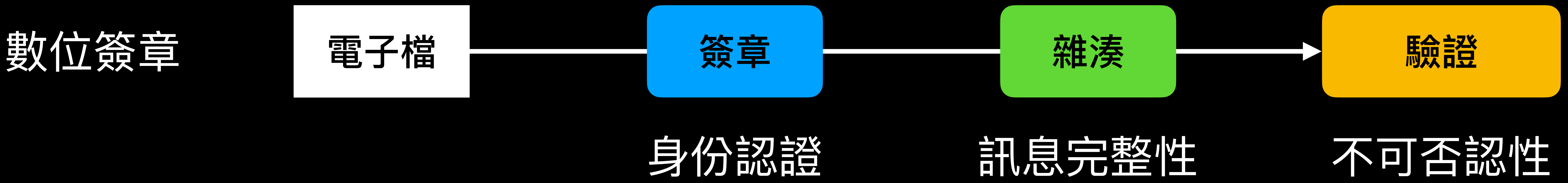
數位簽章
Digital Signature

- 傳統簽章
  - 身份認證：簽名需要物理上傳輸
  - 訊息完整性：彌封可偽造
  - 不可否認性：筆跡鑑定不易驗證、可偽造



數位簽章
Digital Signature

- 數位簽章
  - 身份認證：簽章可透過網路傳輸
  - 訊息完整性：雜湊非常難偽造
  - 不可否認性：透過數學驗證，難以偽造



數位簽章
Digital Signature

- 身份認證
  - 網站、App、Wi-Fi 登入等
- 不可否認性
  - 電子文件簽名
  - 網購刷卡紀錄

Lab：非對稱式加密如何實現數位簽章

- 公司 A：一份合約 📜、私鑰 🔑 、未上鎖的盒子 📦🔓
  - 盒子可用私鑰或公鑰上鎖，並用另外一把解鎖
- 公司 B、競爭者：公司 A 的公鑰 🗝️
- 公司 A 要將合約傳給公司 B
- 文件在兩間公司間傳遞時有可能被篡改
- 怎麼做公司 B 才能檢查合約是否在傳遞過程中被篡改？（不須擔心被偷看）

Lab：非對稱式加密如何實現數位簽章

公司 A

競爭者

公司 B

解法：非對稱式加密如何實現數位簽章

公司 A	競爭者	公司 B
🔑	🗝️	🗝️
📦🔓		
📜		

解法：非對稱式加密如何實現數位簽章

公司 A

競爭者

公司 B

解法：非對稱式加密如何實現數位簽章

公司 A	競爭者	公司 B
🔑	🗝️	🗝️
📜📦🔒		

解法：非對稱式加密如何實現數位簽章

公司 A

競爭者

公司 B

解法：非對稱式加密如何實現數位簽章

公司 A	競爭者	公司 B
🔑	🗝️	🗝️
		📜📦🔓

解法：非對稱式加密如何實現數位簽章

公司 A

競爭者

公司 B

解法：非對稱式加密如何實現數位簽章

公司 A

競爭者

公司 B

解法：非對稱式加密如何實現數位簽章

公司 A

競爭者

公司 B

解法：非對稱式加密如何實現數位簽章

公司 A

競爭者

公司 B

解法：非對稱式加密如何實現數位簽章

公司 A

🔑

競爭者

🗝️

公司 B

🗝️

📜📦🔒

使用公鑰 🗝️ 重新上鎖，

會導致公司 B 無法打開

非對稱式加密如何實現數位簽章

1. 使用私鑰對訊息 m 進行加密，得到的結果稱為簽章 $E(m)$
2. 將訊息 m 與簽章 $E(m)$ 一起傳送給接收者
3. 接收者使用公鑰解密簽章，檢查結果 $D(E(m))$ 是否與收到的訊息 m' 相同

$$m \longrightarrow m, E(m) \longrightarrow m' = D(E(m))$$

$$\xrightarrow[\text{竄改}]{\qquad\qquad} m' \neq D(E(m))$$

數位簽章的應用

- PDF 數位簽名
- 身份驗證
  - SSH 金鑰登入
  - 通行金鑰 PassKey
    - 比密碼更安全的身份驗證方式
    - Apple、Google、GitHub、Uber 都已支援
  - HTTPs

中間人攻擊：金鑰交換

Man-in-the-Middle Attack

Bob	老師	Alice
🔑		🏷️
📦🔓		

中間人攻擊：金鑰交換

Man-in-the-Middle Attack

Bob

老師

Alice

中間人攻擊：金鑰交換
Man-in-the-Middle Attack

Bob	老師	Alice
🔑	🗝️	🏷️
	📦🔓	
	🎁🔓	

中間人攻擊：金鑰交換

Man-in-the-Middle Attack

Bob

🔑

老師

🗝️

📦🔓

Alice

🏷️

🎁🔓

中間人攻擊：金鑰交換

Man-in-the-Middle Attack

Bob　　　老師　　　Alice

中間人攻擊：金鑰交換

Man-in-the-Middle Attack

Bob　　老師　　Alice

中間人攻擊：金鑰交換

Man-in-the-Middle Attack

Bob 老師 Alice

🔑 🗝️

🏷️📦🔒

🎁🔓

中間人攻擊：金鑰交換
Man-in-the-Middle Attack

Bob 老師 Alice

中間人攻擊：金鑰交換

Man-in-the-Middle Attack

Bob 老師 Alice

Alice 與 Bob 無從得知
內容被監聽，甚至調換

中間人攻擊：數位簽章

Man-in-the-Middle Attack

公司 A	競爭者	公司 B
		
		

中間人攻擊：數位簽章

Man-in-the-Middle Attack

公司 A 競爭者 公司 B

🔑 🗝️ 🗝️

📜📦🔒

中間人攻擊：數位簽章
Man-in-the-Middle Attack

公司 A

競爭者

公司 B

中間人攻擊：數位簽章

Man-in-the-Middle Attack

公司 A	競爭者	公司 B
🔑	🗝️	🧷
	📌🧷	
	🏷️📦🔒	

中間人攻擊：數位簽章

Man-in-the-Middle Attack

公司 A	競爭者	公司 B
🔑	🗝️	🧷
	📌🧷	
		🏷️📦🔓
		公司 B 無從得知內容竄改

非對稱式加密的中間人攻擊

- 解決了訊息的不安全通訊環境與身份驗證，卻變成公鑰有同樣的問題
- 不安全的通訊環境：取得的公鑰不一定是真的
- 公鑰的身份驗證：無法驗證公鑰的提供者身份

公鑰基礎建設 PKI
Public Key Infrastructure

- 實務上大量降低中間人攻擊的可能性
- 例子：臺南一中的網站
  - 憑證頒發機構 CA：Google
  - 服務提供者：臺南一中
  - 使用者：學生

公鑰基礎建設 PKI：臺南一中網站

臺南一中　　Google　　學生

Google 的私鑰 🗝️

📌 Google 的公鑰

臺南一中的公鑰 🔑

公鑰基礎建設 PKI：臺南一中網站

臺南一中

Google

🗝️

🔑

學生

📌

公鑰基礎建設 PKI：臺南一中網站

臺南一中	Google	學生
	🗝️	📌
	🔑	
	🏷️	

Google 簽發的臺南一中數位憑證🏷️，

包含由 🗝️ 簽發的 🔑 簽章和相關資訊

公鑰基礎建設 PKI：臺南一中網站

臺南一中	Google	學生
🔑🏷️	🗝️	📌

公鑰基礎建設 PKI：臺南一中網站

臺南一中	Google	學生
	🗝️	📌
		🔑🏷️

公鑰基礎建設 PKI：臺南一中網站

臺南一中

Google

🗝️

學生

📌

🔑🏷️

使用 📌 檢查 🏷️ 中臺南一中的公鑰簽章

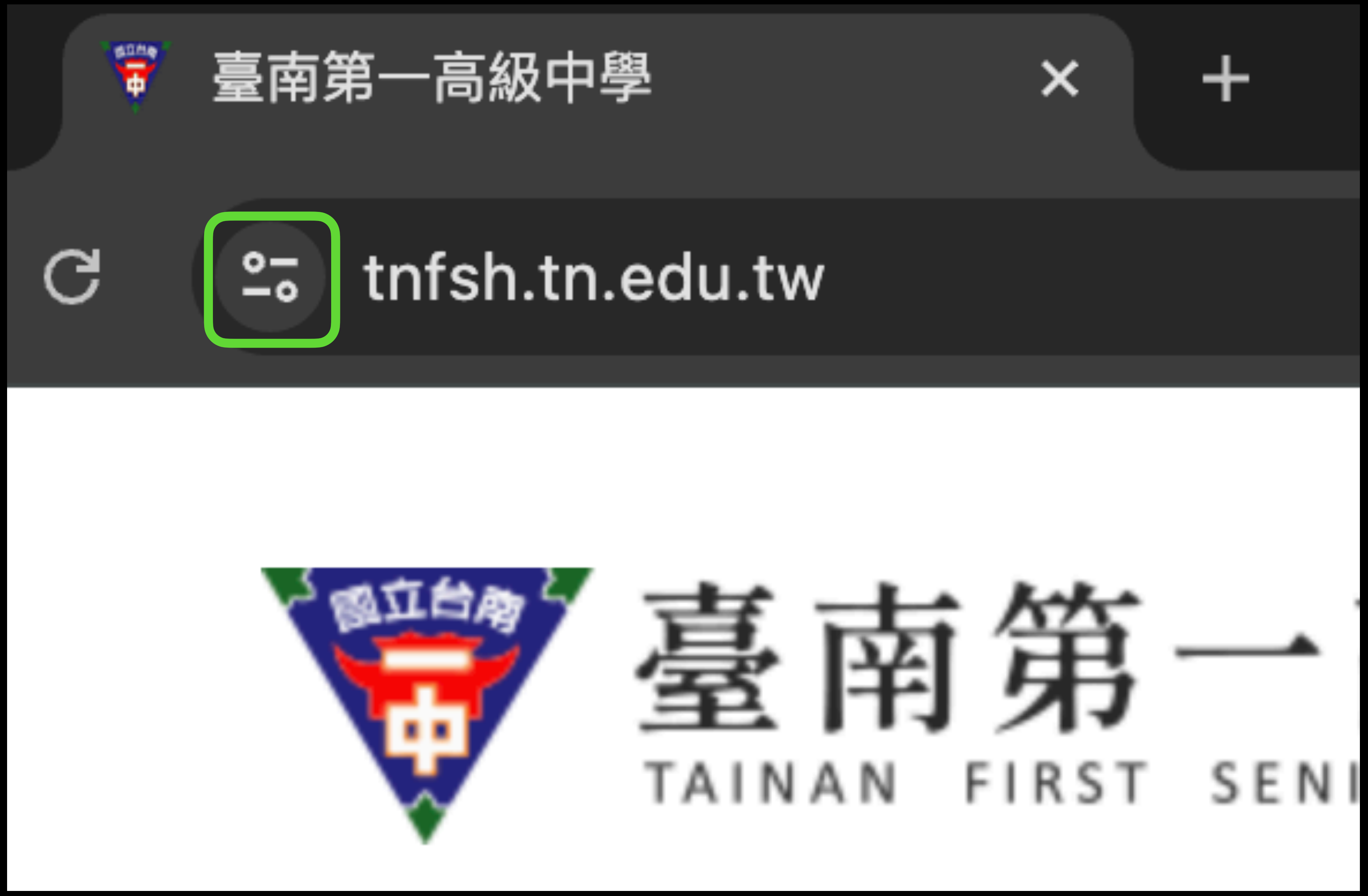
與收到的 🔑 是否相符合

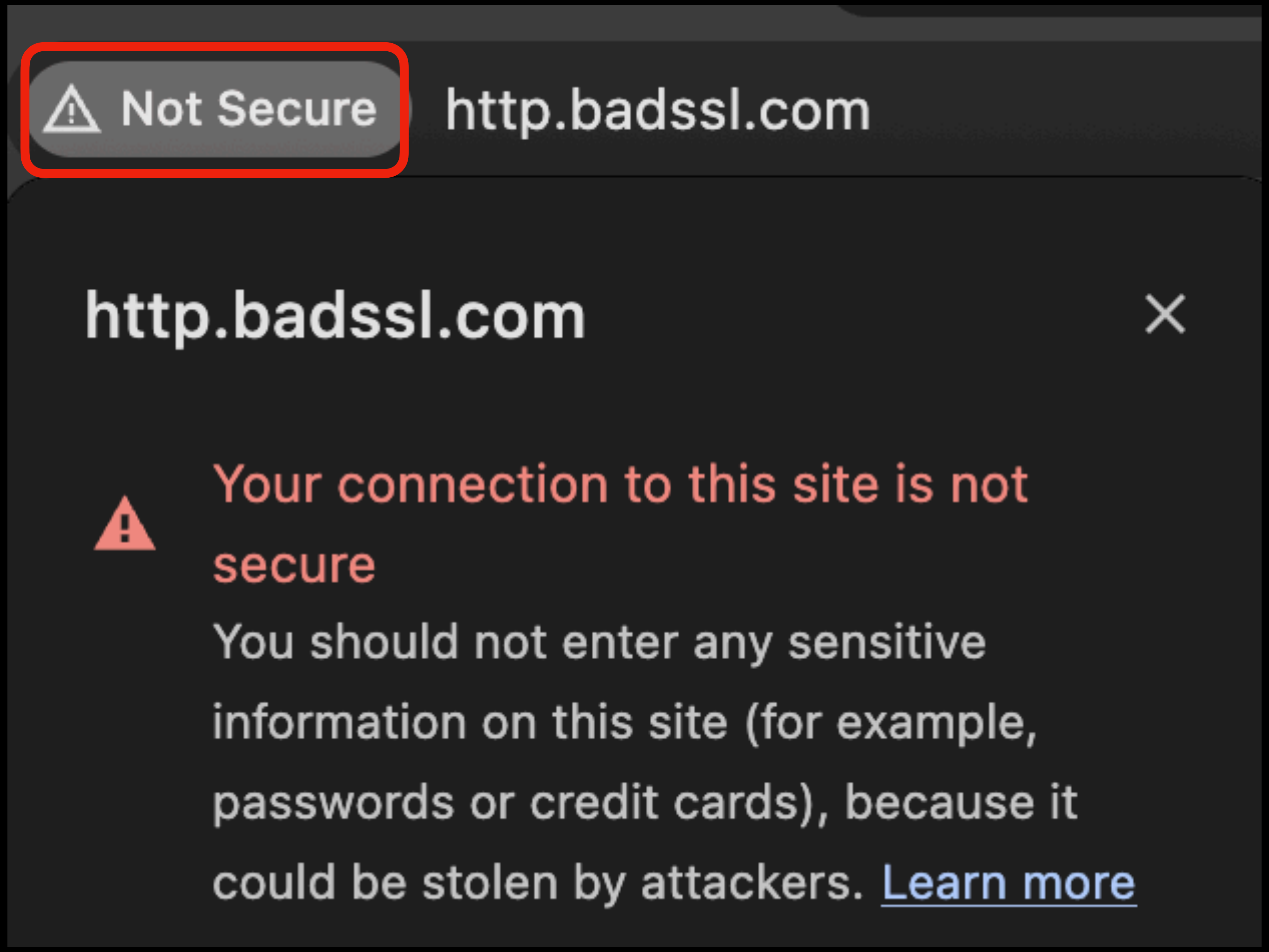
公鑰基礎建設 PKI
Public Key Infrastructure

- 用非對稱式加密解決非對稱式加密的問題 ➡ 理論上沒有解決問題
  - 取得的憑證頒發機構 (Google) 公鑰不一定是真的
  - 無法驗證憑證頒發機構 (Google) 公鑰的提供者身份
- 憑證頒發機構公鑰來源 ➡ 實務上大致解決問題
  - 作業系統內建
  - 瀏覽器內建

HTTPs

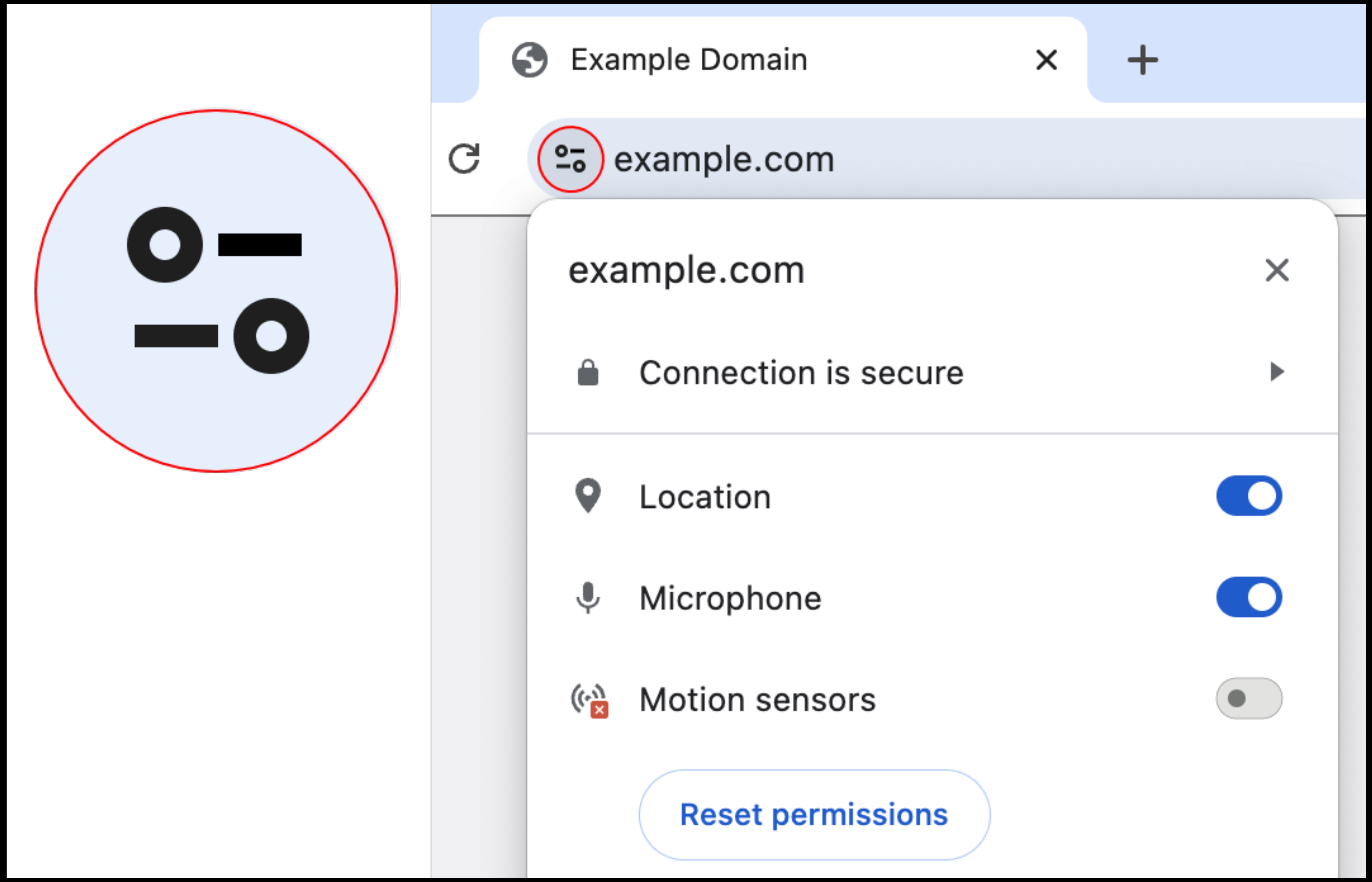
- 用非對稱式加密與 PKI 確保網站連線沒有被監聽或竄改





HTTPs 掛鎖圖標的意義

- 網站連線沒有被監聽或竄改
- 有圖標**不代表網站安全！！**
    - 釣魚網站
    - 惡意程式下載
    - …
- Chrome瀏覽器上的掛鎖圖示即將退役



休息一下～

RSA

RSA 簡介

- 世界上最廣泛使用的非對稱式加密演算法之一
- 1977 年由 Ron **R**ivest, Adi **S**hamir, Leonard **A**dleman 提出
- 基於極大整數的因數分解問題
  - RSA 金鑰長度 2048 bits ➡ $2^{2048} \approx 10^{617}$ 位的整數
  - 可觀測宇宙中的原子總數 $\approx 10^{82}$

模運算
Modular Arithmetic

- 模就是「餘數」；可以把模運算理解為「對餘數的運算」
- $a \equiv b \pmod{n}$ 讀作「對於 n，a 與 b 同餘」，表示 a, b 對 n 的餘數相同
  - 有時也讀作「$a \mod n$ 餘 b」

$$2 \div 5 = 0 \ldots 2$$

$$7 \div 5 = 1 \ldots 2 \quad ➡ \quad 7 \equiv 2 \pmod{5}$$

$$-3 \div 5 = -1 \ldots 2 \quad ➡ \quad -3 \equiv 2 \pmod{5}$$

模運算定律
Modular Arithmetic

- 令 $a \equiv b \pmod{n}$ 和 $p \equiv q \pmod{n}$，則

$$a + c \equiv b + c \pmod{n}$$

$$a - c \equiv b - c \pmod{n}$$

$$ac \equiv bc \pmod{n}$$

$$a^c \equiv b^c \pmod{n}$$

$$a + p \equiv b + q \pmod{n}$$

$$ap \equiv bq \pmod{n}$$

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：$\dfrac{1}{a}$
    - 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

歐拉總計函數

Euler’s Totient Function

- 簡稱 phi 函數 ϕ
- $\phi(n)$：所有小於 n 的正整數中，與 n 互質的數的數量
  - $\phi(3) = 2,\ \{1,2\}$
  - $\phi(12) = 4,\ \{1,5,7,11\}$
  - $\phi(17) = 16,\ \{1,2,3,\ldots,16\}$
- 當 p 是質數，$\phi(p) = p - 1$

歐拉定理
Euler’s Theorem

- 當 $a, n \in \mathbb{N}$，且 a, n 互質，則

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

RSA 公私鑰生成

1. 隨機選 2 個大質數 p, q（須保密）
2. 計算 $N = p \times q$
3. 計算 $\phi(N) = \phi(p) \times \phi(q) = (p-1)(q-1)$
4. 選擇整數 e，$1 < e < \phi(N)$ 且 $e, \phi(N)$ 互質
5. 計算 e 關於 $\phi(N)$ 的模反元素 d，使得 $ed \equiv 1 \pmod{\phi(N)}$
6. 得到公鑰 (N, e)、私鑰 (N, d)

RSA 加解密

- 加密 $E(m, N, e): m^e \equiv c \pmod N$
- 解密 $D(c, N, d): c^d \equiv m \pmod N$

$$
\begin{aligned}
c^d &\equiv (m^e)^d \pmod N \\
&\equiv m^{1+h\phi(N)} \pmod N \\
&\equiv m \times (m^{\phi(N)})^h \pmod N \\
&\equiv m \times 1^h \pmod N \\
&\equiv m
\end{aligned}
$$

Python 環境安裝

- `python3 -m pip --version`
- `python3 -m pip install pycryptodomex gmpy2`

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

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

Python 複習

- `pow(b, e, n)`
  - b：底數
  - e：指數
  - n：取餘數
- e = -1：取模運算的乘法反元素
  - $3 \times 2 \equiv 1 \pmod 5 \Rightarrow 3^{-1} = 2 \pmod 5$

$$pow(b, e, n) = b^e \mod n$$

```
>>> pow(2, 3)
8
>>> pow(2, 3, 3)
2
>>> pow(3, -1, 5)
2
```

Python 複習

- `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

```
int.to_bytes(n, endian)
int.from_bytes(bytes, endian)
```

範例

```
(16).to_bytes(1, 'big')          # 'b\x10'
int.from_bytes(b'\x10', 'big') # 16
```

Python 複習

- `gmpy2.iroot(n, e)`
  - $\sqrt[e]{n}$：對 n 開根號 e
  -

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

變數型別

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

成員運算子

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

Lab：Just RSA

- 嘗試讀懂 template.py，並修復 decrypt 函數，以取得 flag。

休息一下～

Lab：Just RSA

```python
from math import gcd

def gen_keys(p, q):
    n = p*q
    phi_n = (p-1)*(q-1)
    e = 3
    assert gcd(e, phi_n) == 1
    d = pow(e, -1, phi_n)
    return (n, e), (n, d)
```

1. 隨機選 2 個大質數 p, q（須保密）
2. 計算 $N = p \times q$
3. 計算 $\phi(N) = \phi(p) \times \phi(q) = (p-1)(q-1)$
4. 選擇整數 e，$1 < e < \phi(N)$ 且 $e, \phi(N)$ 互質
5. 計算 e 關於 $\phi(N)$ 的模反元素 d，使得 $ed \equiv 1 \pmod{\phi(N)}$
6. 得到公鑰 (N, e)、私鑰 (N, d)

Lab：Just RSA

```python
def encrypt(pub_key, m):
    n, e = pub_key
    m = int.from_bytes(m, 'big')
    c = pow(m, e, n)
    return c
```

- 加密 $E(m, N, e)$：$m^e \equiv c \pmod{N}$

Lab：Just RSA

```python
def decrypt(pri_key, c):
    n, d = pri_key
    # TODO: decrypto c back to m
    m = pow(?, ?, ?)
    return m

p = 1072633900407442949066242771240517880137
q = 1123656898923941482414467987989973072728
c = 8592192951923821689725654840143724498994

pub_key, pri_key = gen_keys(p, q)
print('public key:', pub_key)
print('private key:', pri_key)

m = decrypt(pri_key, c)
print('m:', m.to_bytes(64, 'big').decode())
```

修改 3 個問號

解法：Just RSA

```python
def gen_keys(p, q):
    n = p*q
    phi_n = (p-1)*(q-1)
    e = 3
    assert gcd(e, phi_n) == 1
    d = pow(e, -1, phi_n)
    return (n, e), (n, d)

def encrypt(pub_key, m):
    n, e = pub_key
    m = int.from_bytes(m, 'big')
    c = pow(m, e, n)
    return c
```

$N = p \times q$

$\phi(N) = \phi(p) \times \phi(q) = (p-1)(q-1)$

選擇 e

$ed \equiv 1 \pmod{\phi(N)}$

$m^e \equiv c \pmod{N}$

解法：Just RSA

$$c^d \equiv m \pmod{N}$$

```python
def decrypt(pri_key, c):
    n, d = pri_key
    # TODO: decrypto c back to m
    m = pow(c, d, n)
    return m
```

```
❯ python3 template.py
public key: (12052724822125
21313086609775049460245828836
20277662235563309080854357
private key: (12052724822125
72131308660977504946024582830
12027766223556303309080854357
04289382149459164679091634549
00541116618311131282713067917
06869486377377815592128948755
m: FLAG{[redacted]}
```

弱點：e 過小

- 加密 $E(m, N, e)$：$m^e \equiv c \pmod{N}$
- 當 e 過小時，則導致

$$m = \sqrt[e]{c}, \quad m^e < N$$
$$m = \sqrt[e]{c + kN}, \quad m^e \geq N$$

Lab：small e

- 嘗試讀懂 template.py，並修復 decrypt 函數，以取得 flag。

Lab：small e

- iroot(a, n)：對 a 開 n 次方

```python
from gmpy2 import iroot


def decrypt(n, e, c):
    # TODO: fix decrypt
    m, ok = iroot(?, ?)
    return int(m)


n = 120527248221251923629021200558946883255
e = 3
c = 1727687814057013874866059617223342339697

m = decrypt(n, e, c)
print('m:', m.to_bytes(64, 'big').decode())
```

修改 2 個問號

解法：small e

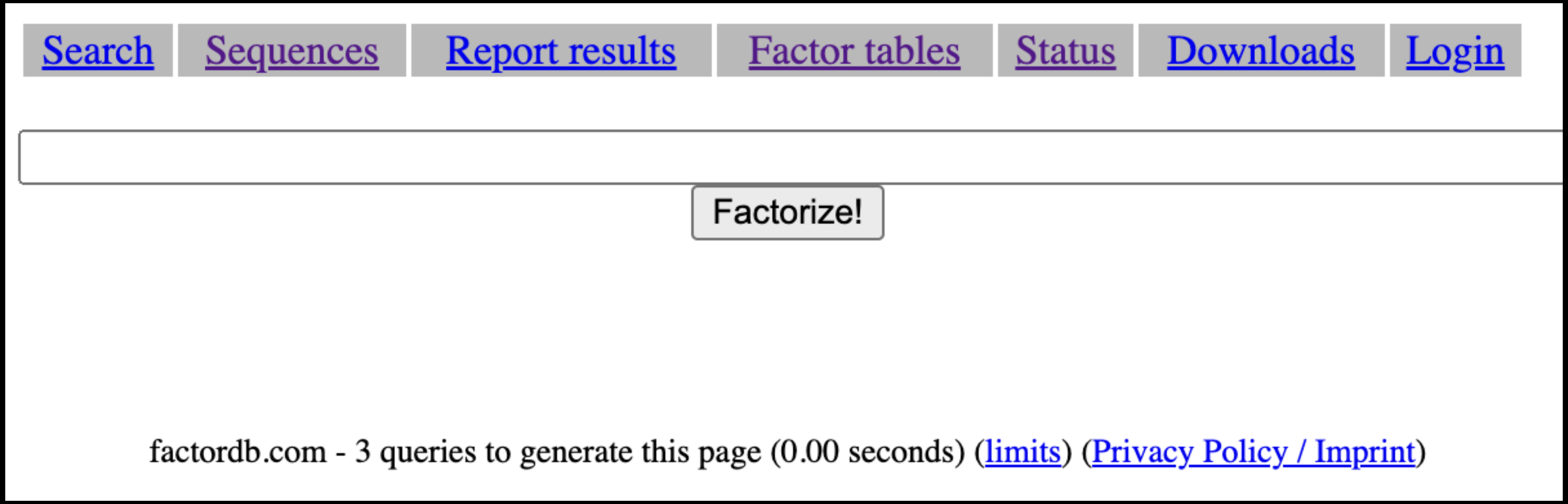
$$m = \sqrt[e]{c}, \quad m^e < N$$

```python
def decrypt(n, e, c):
    # TODO: fix decrypt
    m, ok = iroot(c, e)
    return int(m)
```

```
❯ python3 template.py
m: FLAG{[redacted]}
```

弱點：N 過小

- 當 N 過小時 (512 bits)，則可以暴力猜測 p, q 的所有組合
- 必備線上工具：factordb.com



Lab：small N

- 嘗試讀懂 template.py，並修復 decrypt 函數，以取得 flag。

Lab：small N

```python
def decrypt(n, e, c):
    # TODO: decrypt it

    return m

n = 347242881983155291210545834692548833043529854935876965261873
e = 65537
c = 285951821163368664569734439946604343253778819531645145369177

m = decrypt(n, e, c)
print('m:', m.to_bytes(64, 'big').decode())
```

填上程式碼

解法：small N

[Search](#) [Sequences](#) [Report results](#) [Factor tables](#) [Status](#) [Downloads](#) [Login](#)

347242881983155291210545834692548833043529854935876965261873

Factorize!

Result:

status (?)	digits	number
FF	60 (show)	3472428819...73<60> = 578738136638592152017576391161<30> · 599999999999999999999999999993<30>

解法：small N

```python
def decrypt(n, e, c):
    # TODO: decrypt it
    p, q = 578738136638592152017576391161, 5999999999999999999999999999993
    phi_n = (p-1)*(q-1)
    d = pow(e, -1, phi_n)
    m = pow(c, d, n)
    return m

n = 3472428819831552912105458346925488330435298549358769652 61873
e = 65537
c = 2859518211633686645697344399466043432537788195316451453 69177

m = decrypt(n, e, c)
print('m:', m.to_bytes(64, 'big').decode())
```

RSA 的安全性

- 基於極大整數因數分解問題
  - 標準實作方式沒有問題
  - 有許多數學上的特性導致的弱點需要考慮
  - 非常不適合自行實作加密演算法
- 量子電腦
  - 秀爾演算法

秀爾演算法

- 1994 年提出，於量子電腦上計算質因數分解的演算法
- RSA 金鑰長度 2048 bits ➡ $2^{2048} \approx 10^{617}$ 位的整數
  - 現在的超級電腦：近乎不可能
  - 成熟的量子電腦：一年內
- 量子電腦離成熟還有一段時間

ElGamal

Diffie-Hellman 金鑰交換

- 金鑰交換方法
  - 可用於對稱式加密
- 基於離散對數問題

Diffie-Hellman 金鑰交換概念

1. 選擇質數 p, 對於 p 的乘法循環群的生成元 g，並公開 p, g
2. Alice 選擇密鑰 a，並將 $g^a \pmod p$ 傳給 Bob
3. Bob 選擇密鑰 b，並將 $g^b \pmod p$ 傳給 Alice
4. Alice 計算 $g^{ab} \pmod p$，作為共用金鑰
5. Bob 計算 $g^{ab} \pmod p$，作為共用金鑰
6. Alice 與 Bob 使用共用金鑰生成對稱式加密使用的密鑰

Reference

- oalieno/Crypto-Course
- CTF Wiki - RSA 介紹
- New Directions in Cryptography
- Cryptography/Public Key Overview

Q&A