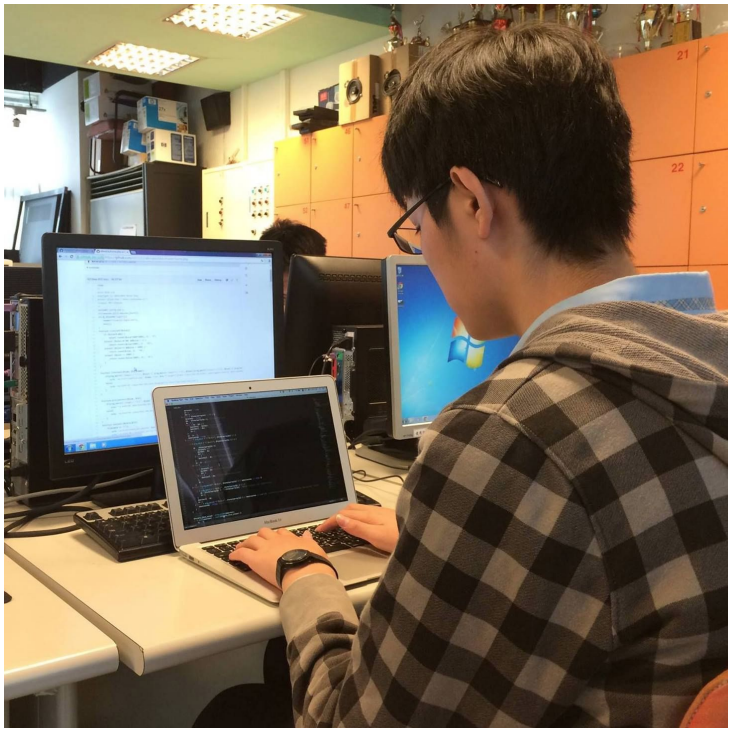
應用密碼學入門

@HITCON CMT 2018

我是誰

- Allen Chou
- 我好像沒什麼值得介紹的
- https://allenchou.cc/
- GitHub: s3131212
- s3131212@gmail.com
- FB: s3131212



背景知識

密碼學是什麼

- 不是研究怎麼設安全的密碼
- 不是教你怎麼破解別人 Facebook
- 你不會因為知道密碼學在幹嘛就變成天才駭客
- 很多數學
  - 我是說，真的很多
  - 不過我並沒有打算講很多數學理論
  - 我自己數學也不好 QQ
- 如果已經沒興趣了，可以趕快離開我不會介意

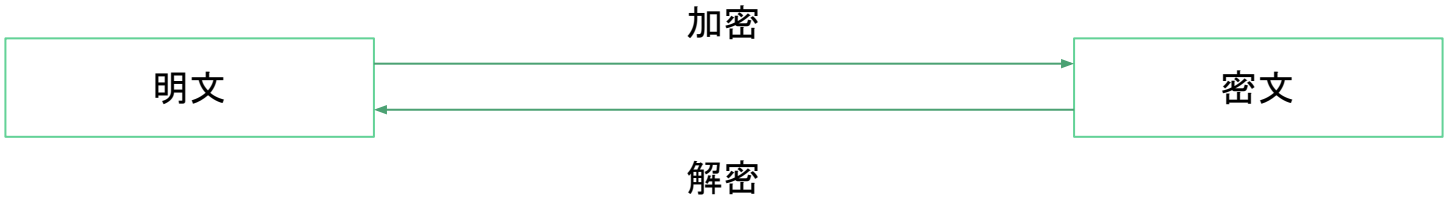
密碼學是什麼

- 古典密碼學
  - 資料保密、傳遞
  - 密碼破譯
- 現代密碼學
  - 古典密碼學的所有東西
  - 資料完整性驗證（Data integrity）
  - 資料的不可否認性（Non-repudiation）
  - 雜湊函數（Hash）
  - 亂數
  - 隱寫術（Steganography）
  - ...

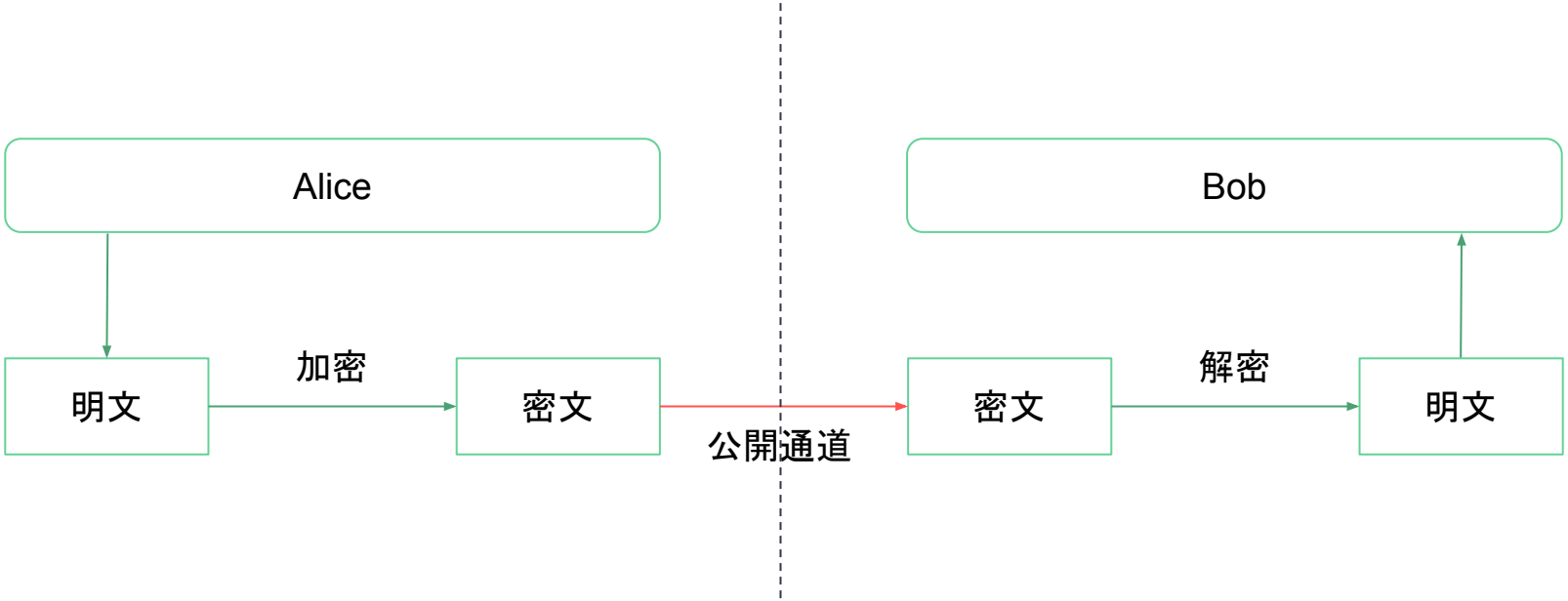
常見詞彙解釋

1. 加密 Encrypt：指將明文經過某種程序轉換成密文，該程序稱為加密
2. 解密 Decrypt：指將密文經過某種程序轉換成明文，該程序稱為解密
3. 明文 Plaintext：加密前的訊息
4. 密文 Cipertext：加密後的訊息
5. 演算法 Algorithm：解決複雜問題的程序
6. 密碼學演算法：做與密碼學相關程序（如加密、解密、簽章...）的演算法
7. 金鑰 / 密鑰 Key：加解密時所使用的「鑰匙」

加密 & 解密



加密 & 解密



柯克霍夫原則(Kerckhoffs's principle)

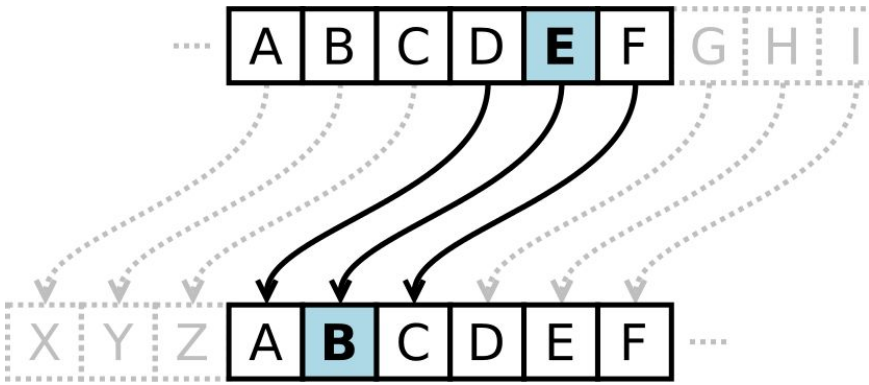
- 即使演算法完全洩漏，只要金鑰沒有洩漏，密文就是安全的
- Claude Shannon: "the enemy knows the system"
- Bruce Schneier: 任何以隱藏設計作為防護(Security through obscurity)的保安系統必然會失敗
- Kerckhoffs's principle 不是說密碼學演算法都必須公開，而是要確保即使公開也不會傷害安全性

古典密碼學

凱薩密碼（Caesar cipher）

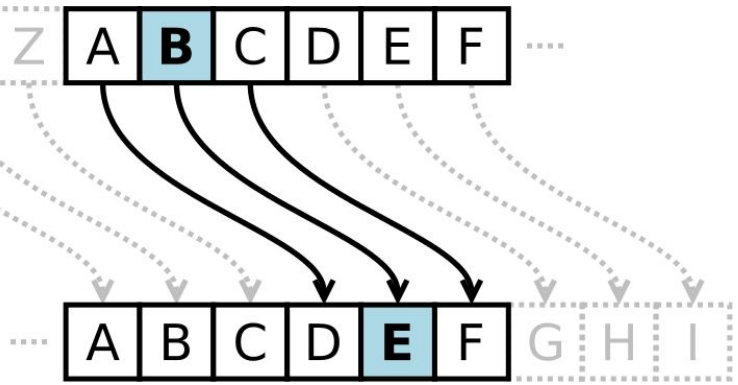
- 好像每次講密碼學都要從他開始講起 Orz
- 加密：簡單來說，就是把字母左右偏移 n 位，方向及 n 作為金鑰
- 解密：就...挪回來...

加密（以向左偏三位為例）



By Cepheus
https://commons.wikimedia.org/wiki/File:Caesar3.svg (Public domain)

解密



By Matt_Crypto
http://en.wikipedia.org/wiki/File:Caesar3.png (Public domain)

凱薩密碼(Caesar cipher)

- 加密(向左偏移三格):hitcon -> efqzlk
- 解密(向右偏移三格):efqzlk -> hitcon

- 攻擊:暴力破解太簡單，也才 26 種可能
- 據說凱薩當年就是用往左偏移三個字母來加密的
  - 阿不過，他的敵人大多不識字

單一字元替代密碼

- 和凱薩密碼一樣是字母一對一代換，但沒有規律
- 換字表（密鑰）：

  a -> h

  b -> e

  c -> q

  d -> k

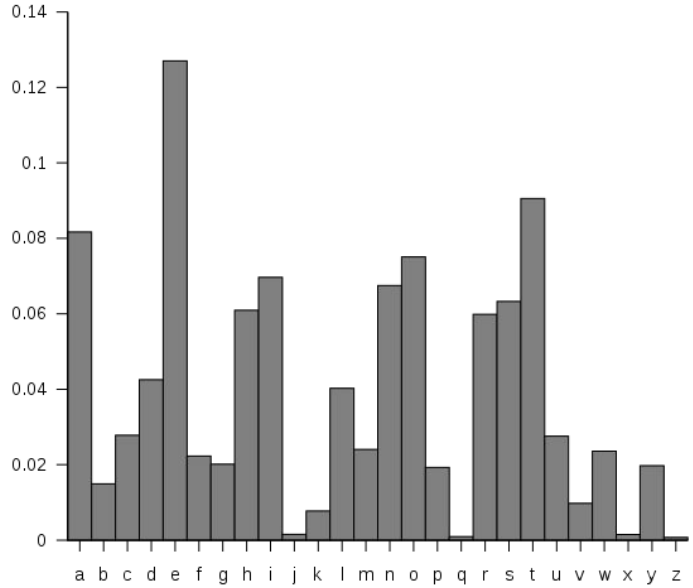
  ...
- 加密：dcba -> kqeh
- 解密：kqeh -> dcba
- 其實也不一定要換成另一個字母（e.g. 豬圈密碼）

單一字元替代密碼

攻擊：字頻分析（Frequency analysis）

最常出現的字母：e, t, a, o, i
最常出現的單字：the, to, of, and

自動化分析：https://quipqiup.com/



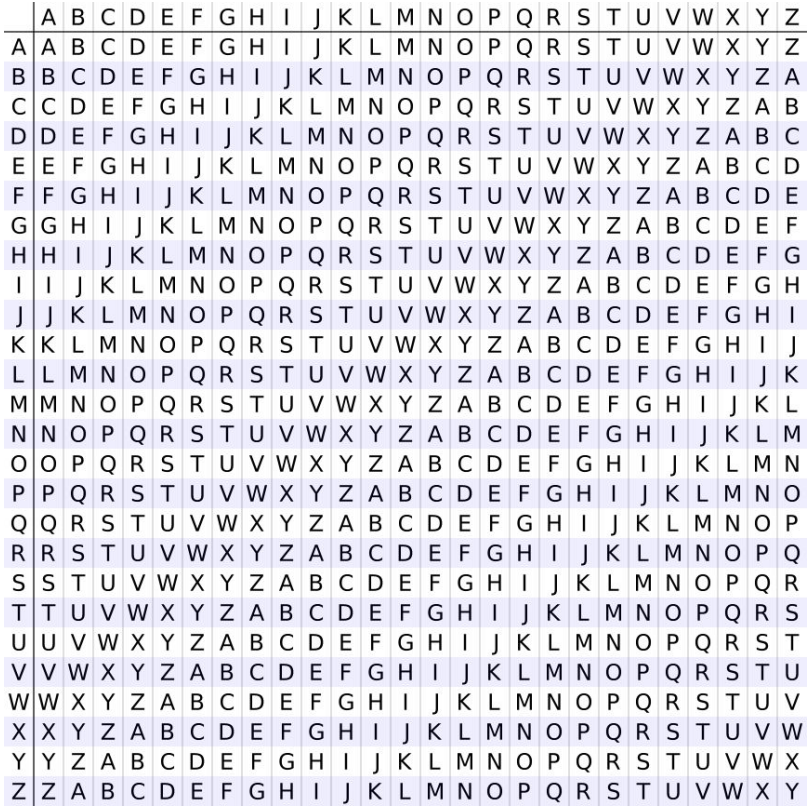
By Nandhp (Public domain)
https://commons.wikimedia.org/wiki/File:English_letter_frequency_(alphabetic).svg

維吉尼亞密碼(Vigenère Cipher)

- 基本上就是一系列的凱薩密碼

明文:platelet is great
密鑰:hitcon(重複填補到明文長度)
密文:wttvsylb bu uelim

- 字頻分析不能用了 QQ
- 沒關係, 還是有方法可以破解



	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

卡西斯基試驗(Kasiski examination)

密鑰:ABCDAB CD ABCDA BCD ABCDABCDABCD
明文:CRYPTO IS SHORT FOR CRYPTOGRAPHY
密文:CSASTP KV SIQUT GQU CSASTPIUAQJB

相差 16 位 => 密鑰為 16 的因數

Index of coincidence

- 已知在英文一段有意義的長文中，隨機取兩字母，相同機率為 0.068
- 已知同樣明文經同樣的密鑰加密後會出來同樣的密文

  C --- key=B ---> D
- 以重複間格不斷取密文，其字母重複的機率應該接近 0.068

  QPWKALVRXCQZIKGRBPFAEOMFL...

Index of coincidence

先把密文拆成 n 行（n=1,2,3...) 並計算每一行的字元出現次數

```
QPWKALVRXCQZIKGRBPFAEOMFL... =>
Q
P
W
K
A
L
.
.
.
```

```
QPWKALVRXCQZIKGRBPFAEOMFL... =>
QP
WK
AL
VR
XC
.
.
.
```

同一行的所有密文都是用同個金鑰加密

Index of coincidence

$$\mathbf{IC} = \frac{\sum_{i=1}^{c} n_i(n_i - 1)}{N(N-1)/c}$$

N = 密文長度
c = 字母數(英文為 26, 以下都以英文舉例)
n_i = 每個字母出現的次數

Size	Delta-bar I.C.
1	1.12
2	1.19
3	1.05
4	1.17
5	1.82
6	0.99
7	1.00
8	1.05
9	1.16
10	2.07

Index of coincidence

理想數值:

$$\mathrm{IC}_{\mathrm{expected}} = \frac{\sum_{i=1}^{c} {f_i}^2}{1/c}$$

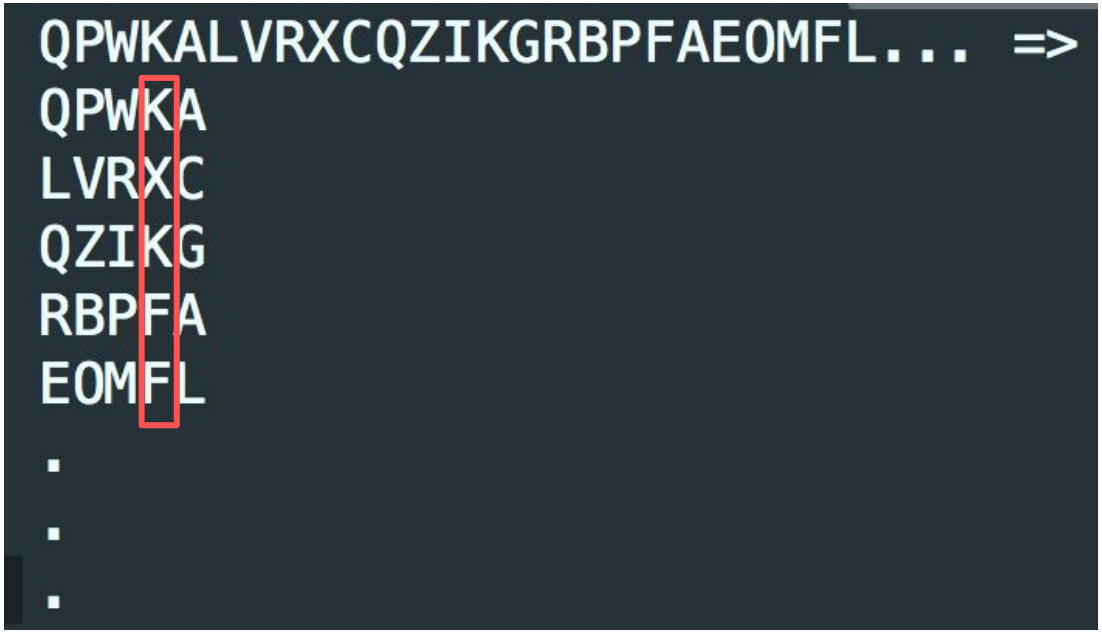
f_i = 該英文字母理論上出現的頻率

（英文的 $\mathrm{IC}_{\mathrm{expected}}$ 約 1.73）

Size	Delta-bar I.C.
1	1.12
2	1.19
3	1.05
4	1.17
5	1.82
6	0.99
7	1.00
8	1.05
9	1.16
10	2.07

Index of coincidence

- 得密鑰長度可能是 5
- 以 5 字元為單位分隔後，每行都是一個凱薩加密
- 每行都做一次字頻分析
- 組合後可得密鑰



其他有趣的古典密碼

- 籬笆密碼法
- 密碼棒
- Enigma

現代密碼學的基本概念

對稱式加密

簡單來說，就是加密解密用的 Key 是同一個。

編碼

- 密碼學是數學，要先把文字轉成數字才能運算
  - 例如 ASCII（A=41, a=97）
- 有時可能會需要二進制
  - ASCII 中 A = 0100 0001 , a = 0110 0001
- 編碼不是加密，可以在沒有金鑰的情況下直接還原出原文

XOR

$A \oplus A = 0$

$B \oplus B = 0$

$\Rightarrow (A \oplus B) \oplus A = B$

$(A \oplus B) \oplus B = A$

A	B	$A \oplus B$	$(A \oplus B) \oplus A$	$(A \oplus B) \oplus B$
0	0	0	0	0
0	1	1	1	0
1	0	1	0	1
1	1	0	1	1

XOR Cipher

Plaintext = Wiki (01010111 01101001 01101011 01101001)
Key = 11110011 *4

加密

```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
-----------------------------------
= 10100100 10011010 10011000 10011010
```

解密

```
  10100100 10011010 10011000 10011010
⊕ 11110011 11110011 11110011 11110011
-----------------------------------
= 01010111 01101001 01101011 01101001
```

XOR Cipher

攻擊：

- 已知明文與密文時可以直接回推 Key（明文⊕密文 = 金鑰）
- 遇到一長串 Null (0x00) 時會直接寫出 Key
  而這在二進制檔案中是很常見的事情
- 如果 Key 長度短於 Plaintext，那基本上就是變種的維吉尼亞密碼
  - 卡西斯基試驗
  - Index of coincidence
- 如果 Key 長度等於 Plaintext，又 Key 完全隨機且不重用（即 One Time Pads）
  是被證實無法破解的（暴力破解也不可行）

AES（Rijndael）

- 美國國家標準局 NIST 於 1997 年開始徵選下一代的對稱式加密系統
  - 稱為 Advanced Encryption Standard，簡稱 AES
  - 要求實作程式碼必須公開（不允許 Security by obscurity）
  - 必須無償給所有人使用
  - 除安全性外要考慮效能、記憶體使用量、是否易於實作等
  - 由全世界所有專家一起研究與評比
- 最後由比利時密碼學家 Joan Daemen 和 Vincent Rijmen 設計的 Rijndael 獲勝
- 金鑰長度（Key sizes）：128, 192 or 256 bits
- 區塊長度（Block sizes）：128 bits
  - 換而言之，AES 規定一次只能加密 128 bits
- 嚴格來說，AES（規範）是 Rijndael（演算法）的 subset

串流加密 vs 區塊加密

串流加密：

1. 逐 bit 加密
2. 明文不斷「流」進加密器
3. 金鑰通常為一個 seed 生成 Keystream
4. 常用於需要即時回應，或是訊息長度未定的情況
5. 常見演算法：Xor Cipher, Salsa20

區塊加密：

1. 一次加密 n 個 bits
2. 一整塊資料塞進加密器
3. 金鑰就是一個字串
4. 常用於已知訊息長度的情況
5. 常見演算法：AES, DES

Block cipher mode of operation

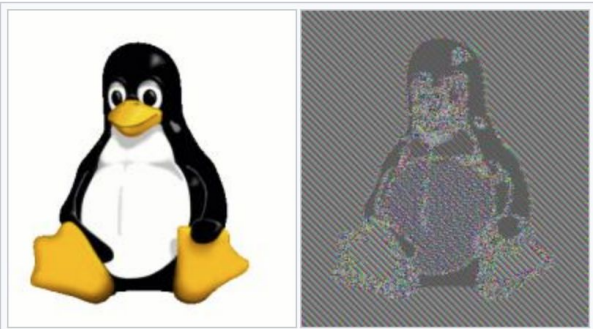
- 區塊加密的對稱式加密演算法通常只能加密特定長度或是長度在一定範圍內的訊息（稱為 Block size）
- 如何用同一個加密演算法與同一個金鑰加密比 Block size 還要長的訊息？
- 將很長的訊息切成數個長度為該演算法之 Block size 的區塊（Blocks）
- 以下只介紹幾個常見的 mode

ECB

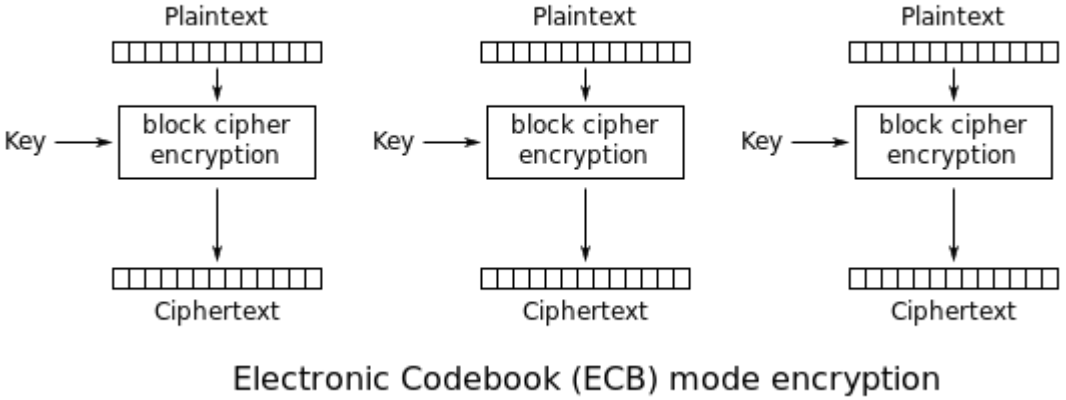
直接把每個 block 個別加密

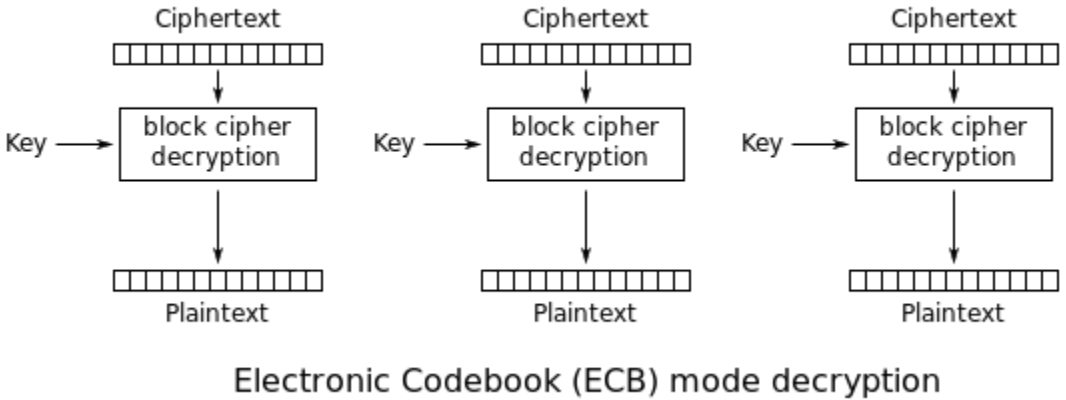
問題:

- 同樣的 block 會被加密成同樣的密文
- block 可以被任意調換位置
- 重送攻擊



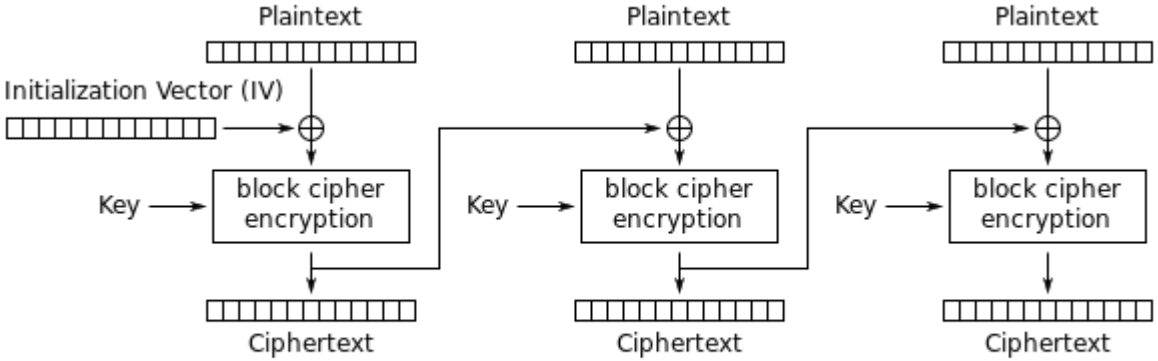
By Lunkwill http://en.wikipedia.org/wiki/Image:Tux_ecb.jpg



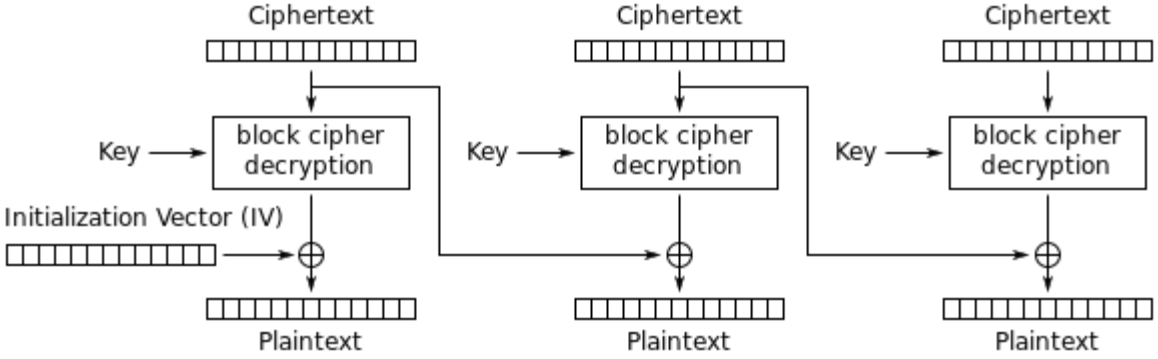


CBC

- 一定要等前一個 block 加密完才能往後加密
- 將前一個 block 的 ciphertext 設為初始向量 (IV)，第一個 block 可以自訂 IV
- 若密文有某個 block 的 bit 錯誤，只會影響該 block 及下一個 block 的解密
- 若密文有某個 block 的 bit 遺失，則會影響後續所有 block 的解密



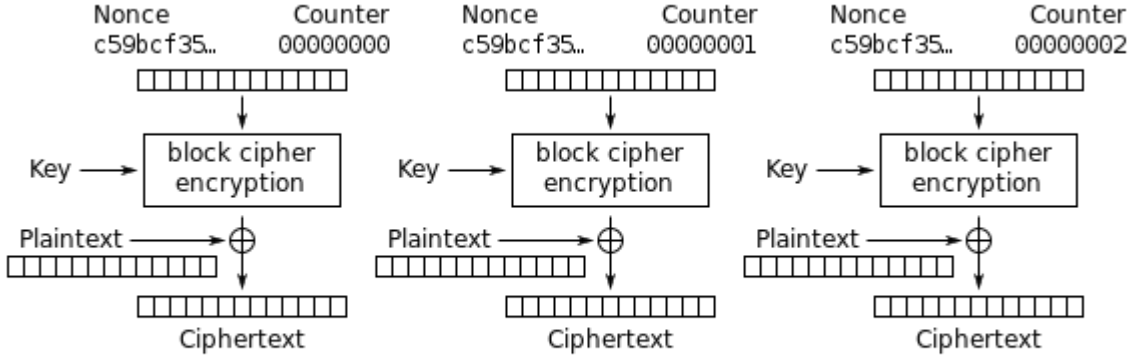
Cipher Block Chaining (CBC) mode encryption



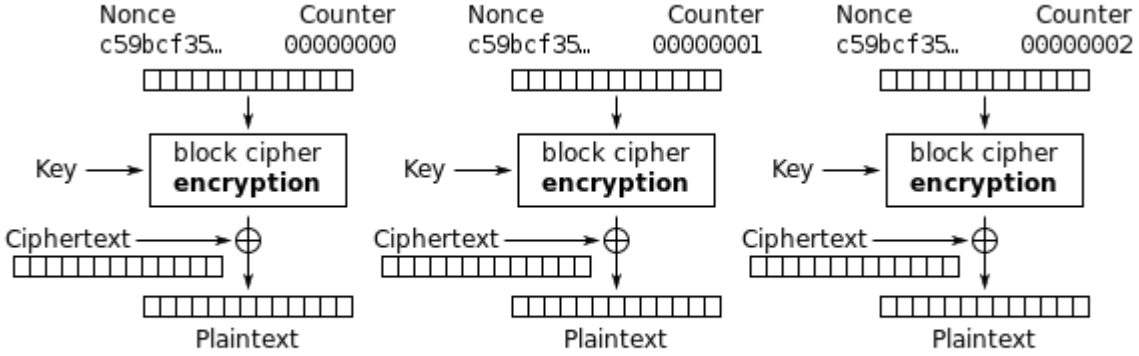
Cipher Block Chaining (CBC) mode decryption

CTR

- counter 從 0 開始遞加
- nonce 為隨機生成
- 模擬 Stream Cipher
- 可以同步加解密
- 丟失任何 block 都不影響其他 block 的解密



Counter (CTR) mode encryption



Counter (CTR) mode decryption

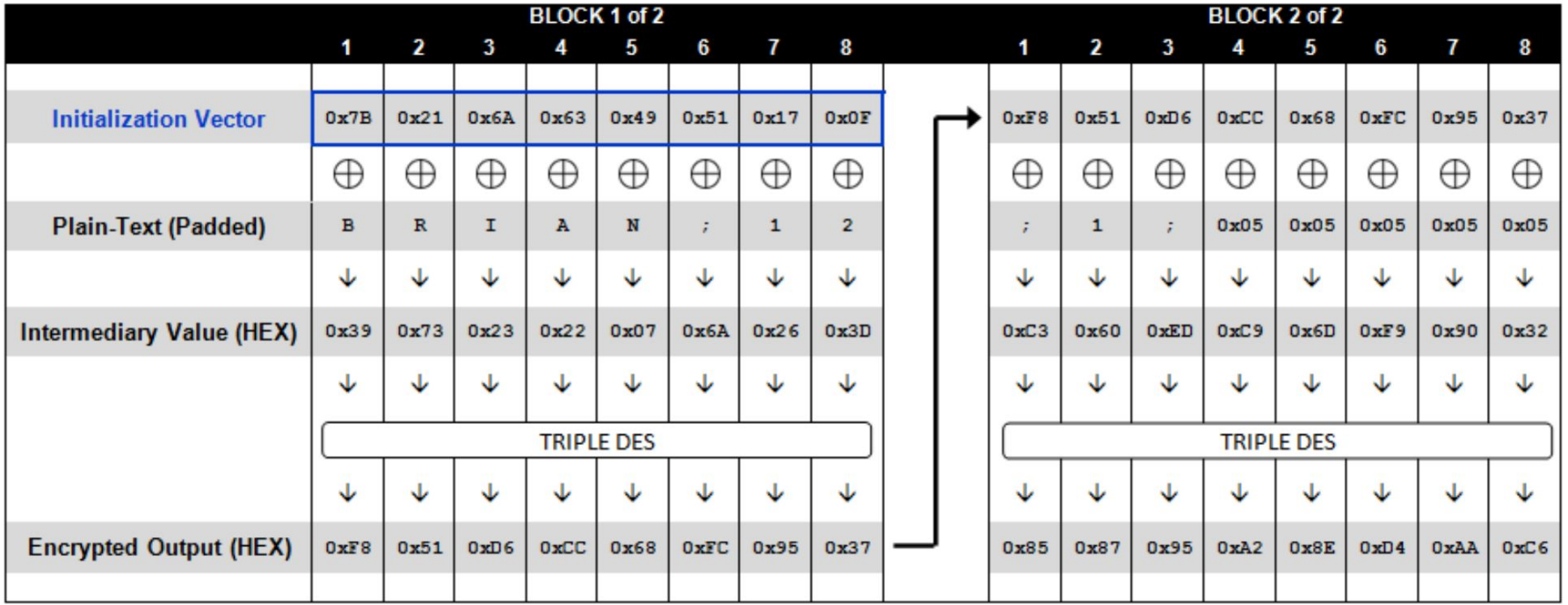
Padding

- 訊息長度不一定剛好為 block size 的整數倍
- 需要某種方式把最後一個 block「填滿」
- PKCS7：缺 N bytes 就用 N 帶入
  - 缺 1 bytes 就用 0x01 填滿
  - 缺 2 bytes 就用 0x02 填滿
  - 不要和 CBC Mode 一起使用（Padding Oracle Attacks）
- ANSI X.923：最後一個 byte 填寫有多少空缺，用 null bytes 填滿其餘空位
  - `DD DD DD DD 00 00 00 04`
- ZeroPadding：就全部用 null byte 填滿

Padding Oracle Attack

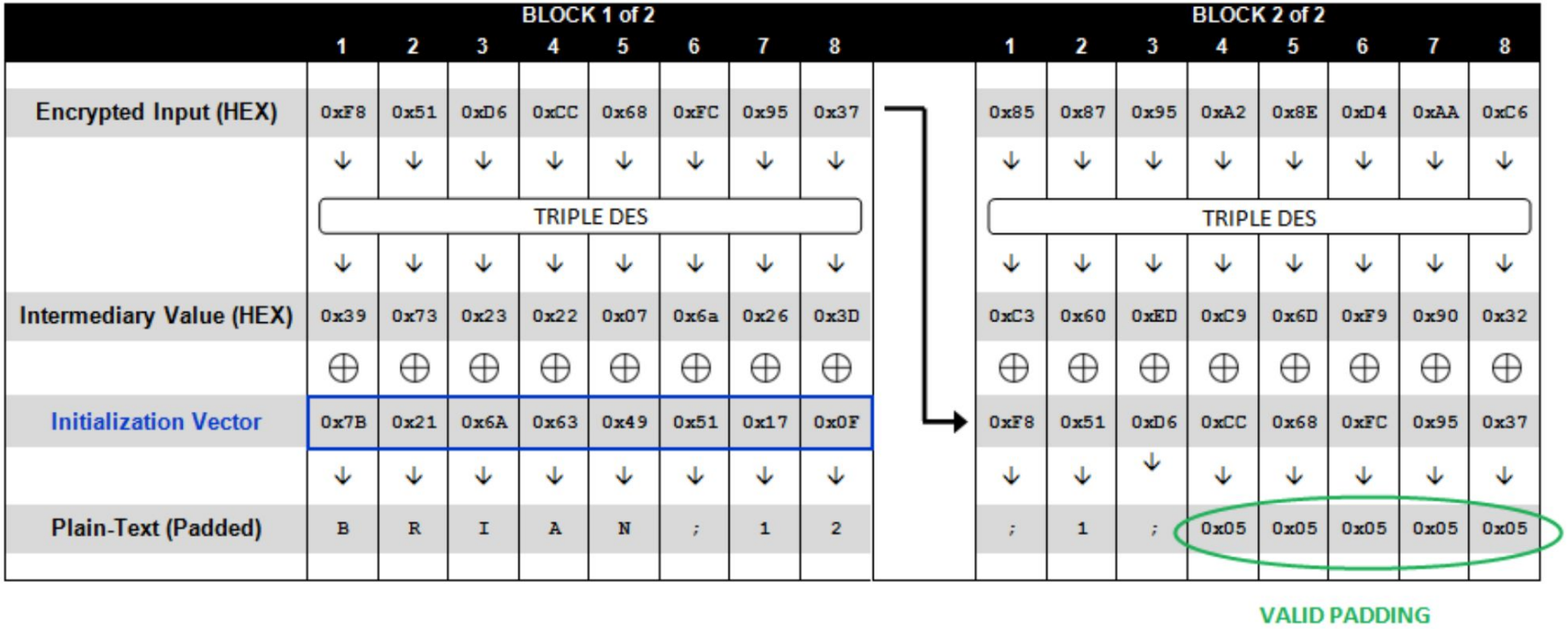
- CBC + PKCS7
- 場景：
  - 解密成功，內容正確：HTTP 200 OK, correct
  - 解密成功，內容錯誤：HTTP 200 OK, error
  - 解密失敗：HTTP 500
- 常見於 Web 的漏洞（e.g. CVE-2010-3332）

Padding Oracle Attack

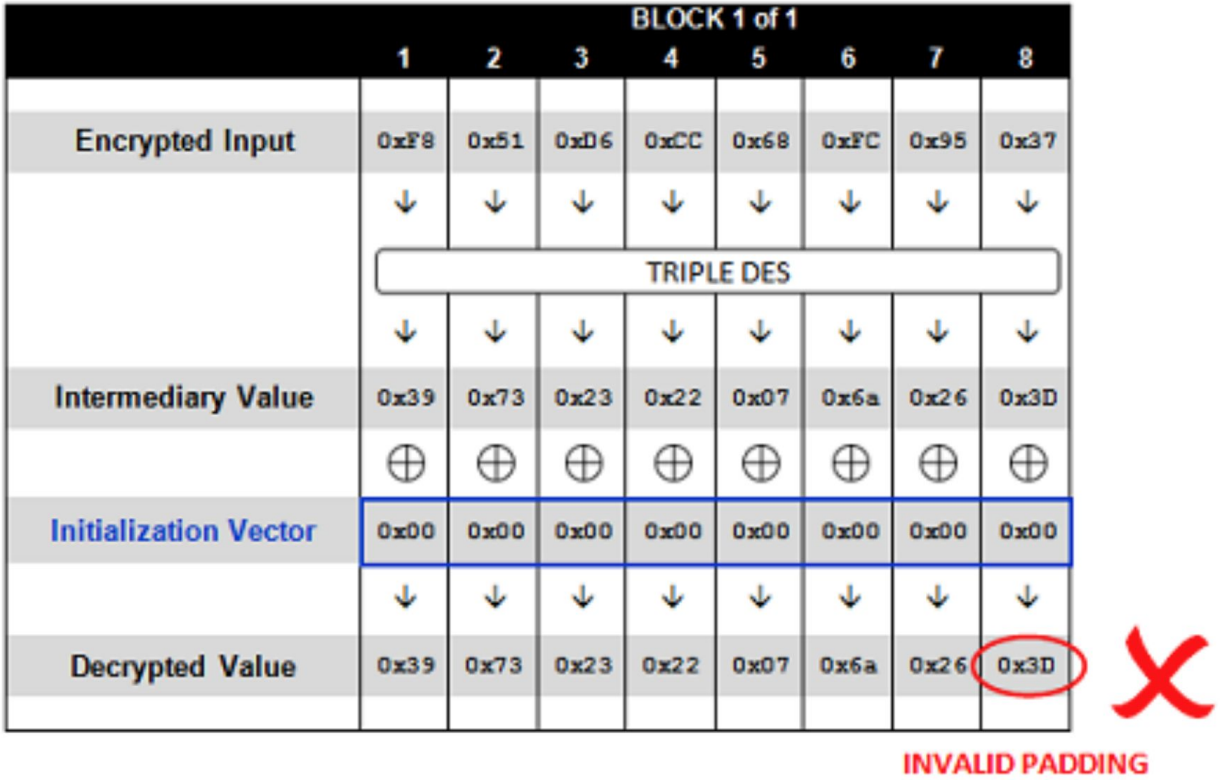


圖皆取自 [Fun with Padding Oracles - OWASP]

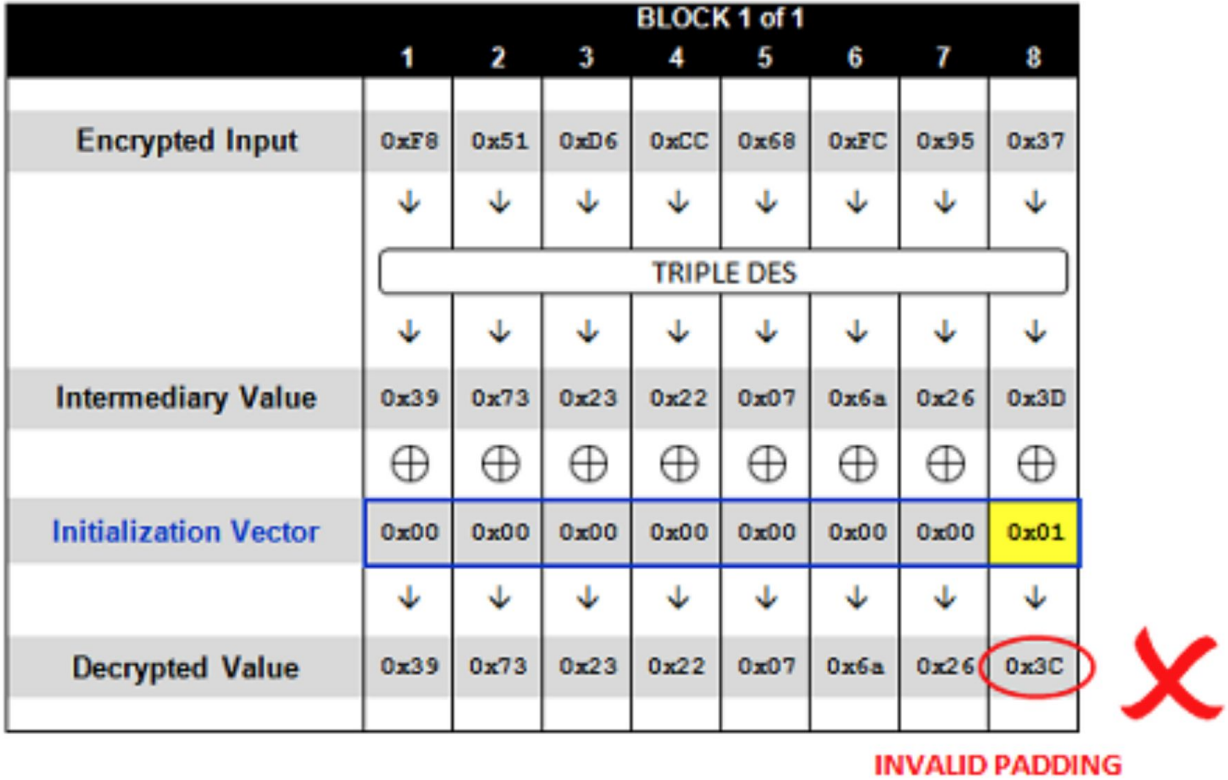
Padding Oracle Attack



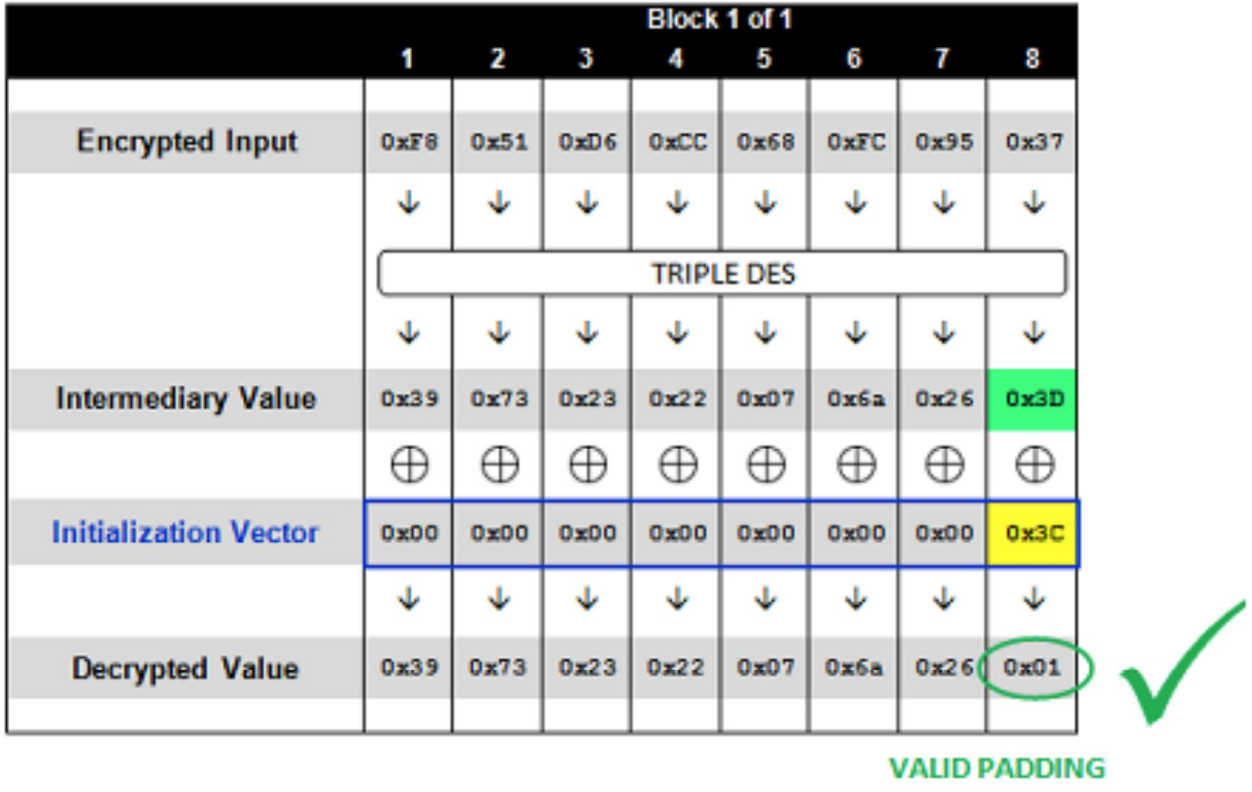
Padding Oracle Attack



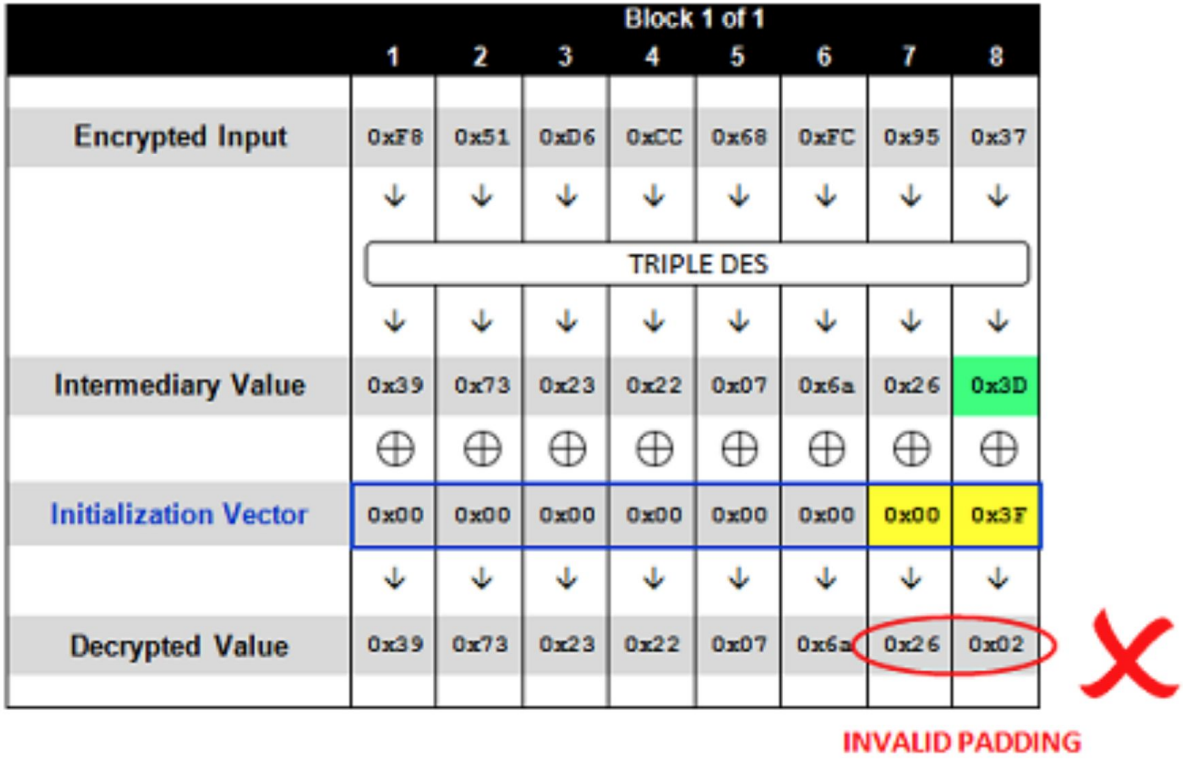
Padding Oracle Attack



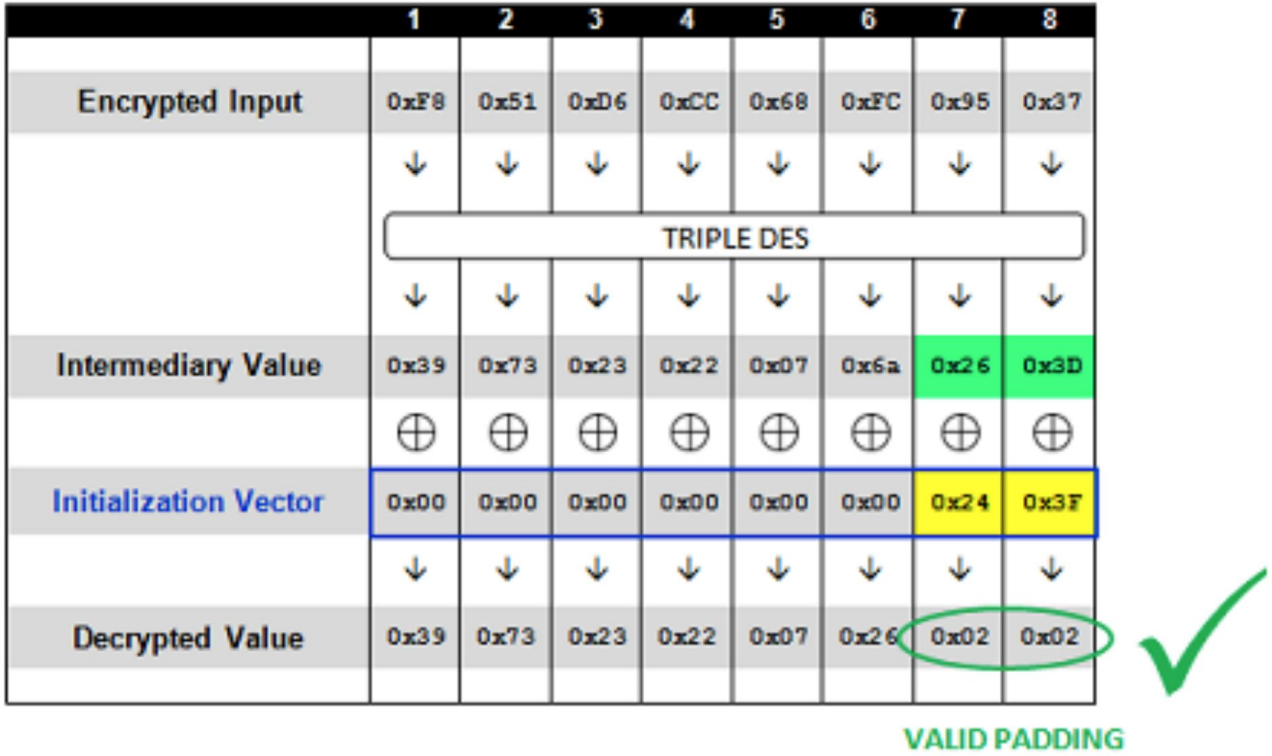
Padding Oracle Attack



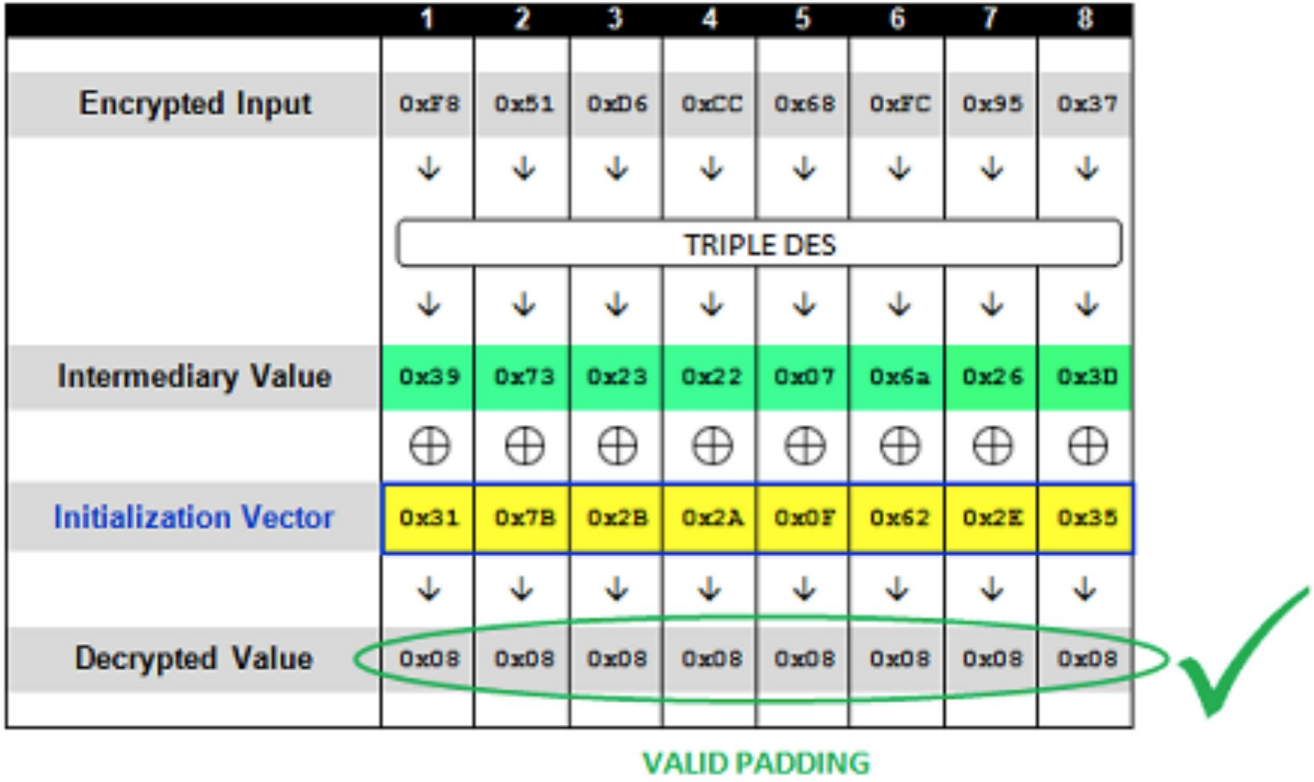
Padding Oracle Attack



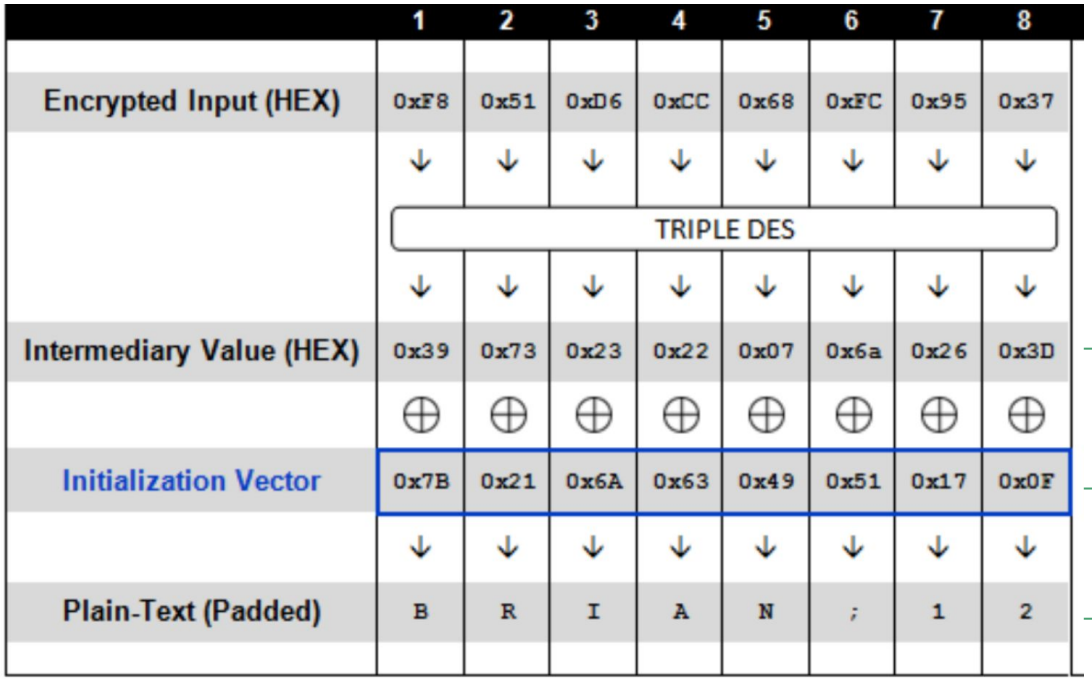
Padding Oracle Attack



Padding Oracle Attack



Padding Oracle Attack



剛剛推出來的中間 值

原本就知道的 IV

兩者 XOR 得明文

其實我寫了一個程式可以繞過密碼

- yoyodiy

Padding Oracle Attack

- 並不是演算法本身安全，密文就不會被破解
- 正確使用演算法、cipher mode、padding mode
- 預防方式
  - 解密前確認密文沒被竄改（訊息認證碼 MAC）
  - 隱藏錯誤訊息（Timing Attack ?)
  - 不要用 CBC Mode

金鑰分配問題

Alice 如何安全的把金鑰送給 Bob？

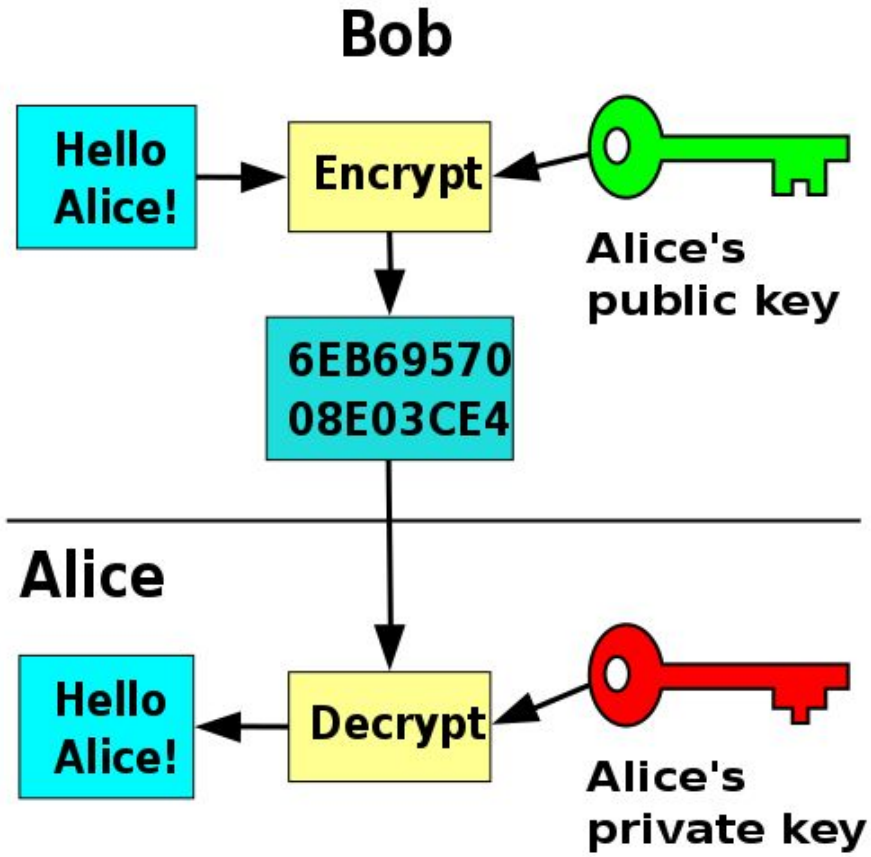
- 事先約定
  - Alice 把金鑰寫在紙條上，偷偷拿給 Bob
- 金鑰管理
- 非對稱式加密系統
  - 有兩把金鑰，用於加密的可以公開給別人，用於解密的要私藏
- Diffie-Hellman key exchange
  - 可以靠著溝通創造出共有金鑰而讓竊聽者無法得知該金鑰

非對稱式加密

加密解密使用不同的 Key

非對稱式加密

1. Bob 要傳訊息給 Alice
2. Bob 拿 Alice 的公鑰（Public Key）對訊息加密
3. Bob 傳加密過後的訊息給 Alice
4. Alice 拿自己的私鑰（Private Key）對訊息解密



非對稱式加密

- Public Key 可以公開
- Private Key 必須自己保存
- 任何人都可以用 Alice 的 Public Key 加密訊息
- 只有 Alice 能將這些訊息以她的 Private Key 解密
- 常見演算法：RSA, ElGamal, ECC
- 運算速度比對稱式加密慢很多，故現在幾乎都是混合對稱式加密（Hybrid cryptosystem）
    - Alice 生成一個會議金鑰（Session Key），以會議金鑰對稱式加密訊息
    - Alice 以非對稱式加密拿 Bob 的 Public Key 加密會議金鑰
    - 兩者合併傳送給 Bob
    - Bob 用自己的 Private Key 解開會議金鑰
    - Bob 以會議金鑰解密訊息

RSA

- 最常見的非對稱式加密系統
- 基於大數質因數分解困難
- Ron Rivest、Adi Shamir、Leonard Adleman 共同發明

RSA

製作 Public Key 與 Private Key：

1. 選擇 2 個超大相異質數 p, q 並計算 $N = pq$
2. 計算 $r = (p-1) \times (q-1)$
3. 選一整數 e 滿足 $e < r$ 且 $gcd(e, r) = 1$
4. 尋一整數 d 滿足 $ed \equiv 1 \pmod{r}$
5. 銷毀 p 與 q，得 Public Key (N, e) 與 Private Key (N, d)

加密與解密：

1. Bob 要傳訊息給 Alice，訊息依據特定方法轉成整數 m 滿足 $m < N$
2. Alice 將 Public Key (N, e) 交給 Bob
3. Bob 運算 $c \equiv m^e \pmod{N}$ 得 c 並交給 Alice（加密）
4. Alice 運算 $c^d \equiv m \pmod{N}$ 得 m，再依約定方法轉回原始內容（解密）

其實看不懂沒關係啦 ...
維基百科上有很詳細的證明可以讀

RSA Padding

- 解決 RSA 的一些神奇特性所造成的問題：
  - 同樣明文、同樣金鑰會得同樣密文
  - 當 $m^e < N$, $m^e \pmod N = m^e$
  - $0^e = 0$
    $1^e = 1$
  - Homeomorphic Property:
    $RSA(k,A) \times RSA(k,B) = RSA(k,A \times B)$
- 訊息一定要先處理過才能 RSA 加密
- 常見：PKCS#1 padding, OAEP

OAEP

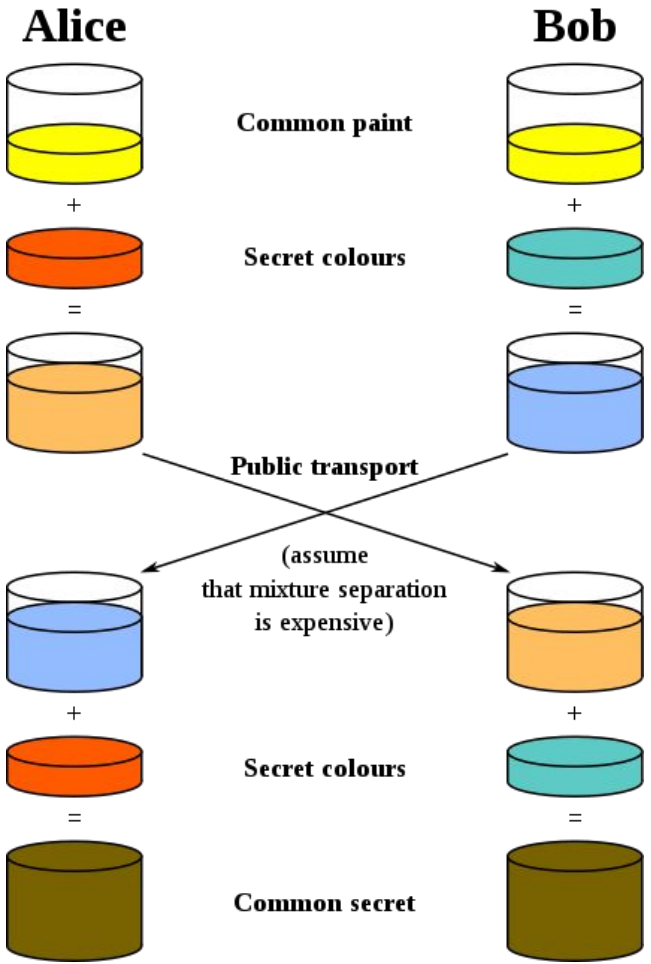
- Optimal Asymmetric Encryption Padding
- 欲加密訊息 m, 隨機生成亂數 r
- 加密:
  - X=(m+000...) ⊕ G(r)
    Y = r ⊕ H(X)
  - 其中 G, H 是公開的 Hash 函數
  - RSA 加密 (X||Y)
- 解密:
  - RSA 解密得 (X||Y)
  - r = Y ⊕ H(X)
  - m+000...= X ⊕ G(r)

Diffie-Hellman key exchange

- 可以靠著溝通創造出共有金鑰而讓竊聽者無法得知該金鑰
- 基於離散對數問題
- 由 Ralph C. Merkle、Bailey Whitfield Diffie、Martin Edward Hellman 提出

Diffie-Hellman key exchange

1. Alice 與 Bob 約定使用 $p=23$，$g=5$
2. Alice 創造一個整數 $a=6$ 並保密
   並計算 $A = g^a \bmod p$ 並傳給 Bob。
   $A = 5^6 \bmod 23 = 8$
3. Bob 創造一個整數 $b=15$ 並保密
   並計算 $B = g^b \bmod p$ 並傳給 Alice。
   $B = 5^{15} \bmod 23 = 19$
4. Alice 計算 $s = B^a \bmod p$
   $19^6 \bmod 23 = 2$
5. Bob 計算 $s = A^b \bmod p$
   $8^{15} \bmod 23 = 2$



密碼學用於資料與身份驗證

雜湊函數（Hash）

- 將任意長度的字串轉成固定長度
- Avalanche effect：字串有些微變動，Hash 差異很大
  - md5(1234) = 81dc9bdb52d04dc20036dbd8313ed055
    md5(1235) = 9996535e07258a7bbfd8b132435c5962
- Pre-image resistance：可以從 X 算出 Hash(X)，但無法從 Hash(X) 算出 X
- Second-preimage resistance：已知 X，很難找到 X’ 符合 Hash(X) = Hash(X’)
- Collision resistance：很難找到兩個不同字串 X 與 X’ 符合 Hash(X) = Hash(X’)
- 例如：md5, sha256, Argon2

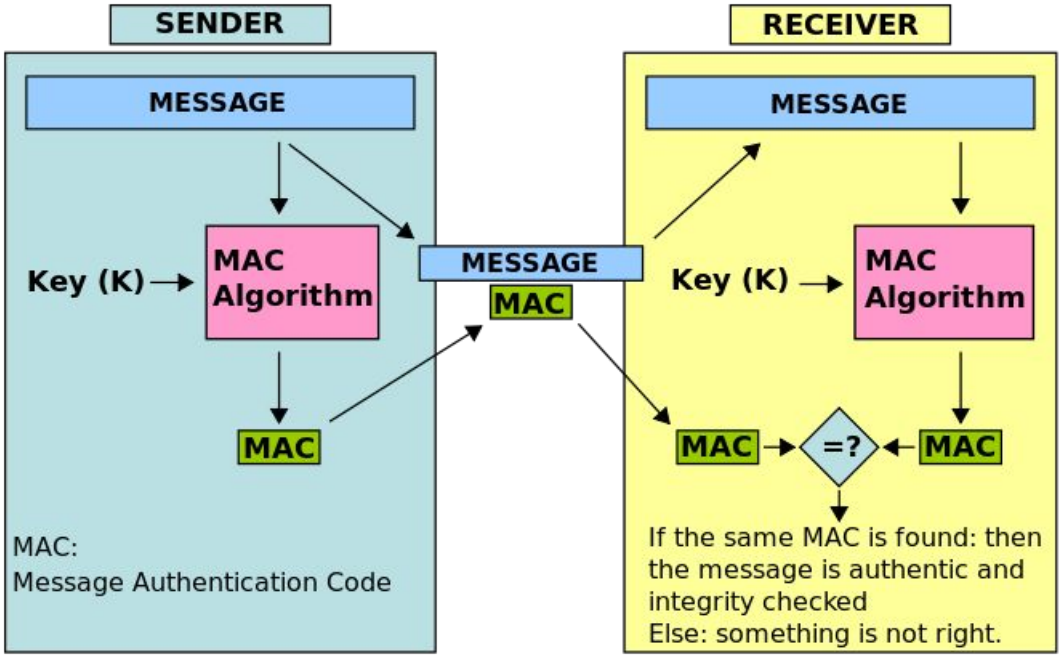
雜湊函數(Hash)

用途:

- 驗證資料完整性(Data integrity)
  - 用不安全通道傳很大的檔案
    用安全通道傳該檔案的 Hash
    以節省加密解密所需的資源
- 在不取得明文的情況下驗證資料正確性
  - 儲存使用者密碼的 Hash 在資料庫，確保管理員看不到使用者密碼的明文

訊息認證碼(MAC)

- Message authentication code
- 驗證完整性
- 可驗證是誰傳來的
  - 只有當密鑰只有 sender 和 recipient 知道時成立
- 常用的有 HMAC(帶有密鑰功能的 Hash)
  和 CBC-MAC(用 Block Cipher 創造 MAC)

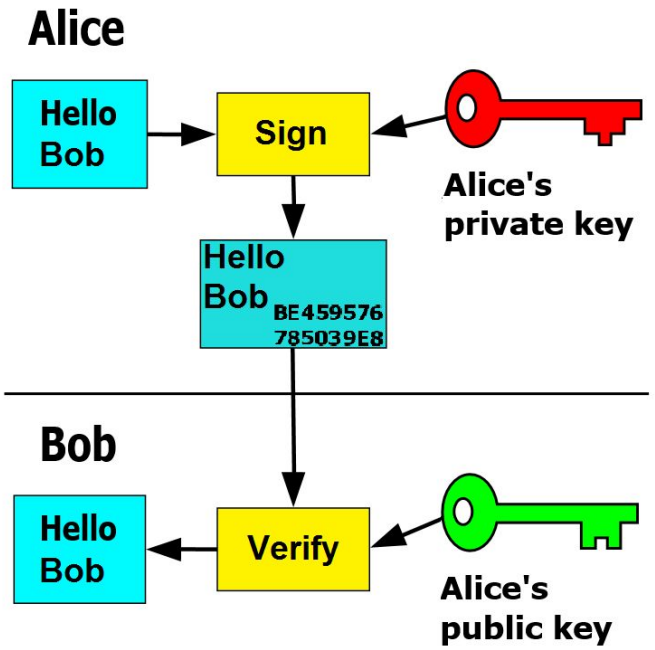


數位簽章（Digital Signature）

- 類似於在紙上簽名，證明這份資料是我認可的
- 只有擁有私鑰的人可以簽章，所有人都可以驗證簽章
- 因為訊息很長，所以通常會先將訊息 Hash 過再簽章
- 基於非對稱式加密系統的應用
- 具有不可否認性（Non-repudiation）

數位簽章（Digital Signature）

1. Alice 想要簽一筆資料 ***D***
2. Alice 把 ***D*** 拿去算 hash 得 ***H***
3. Alice 拿自己的私鑰對 ***H*** 簽章得 ***S***
4. Alice 把 ***(D, S)*** 傳給 Bob
5. Bob 拿 ***D*** 去算 hash 得 ***H'***
6. Bob 把 ***S*** 用 Alice 的公鑰驗證得 ***H***
7. 比較 ***H*** 與 ***H'*** 是否一樣



https://en.wikipedia.org/wiki/File:Digital_Signature_diagram.svg

Hash vs. MAC vs. Digital Signature

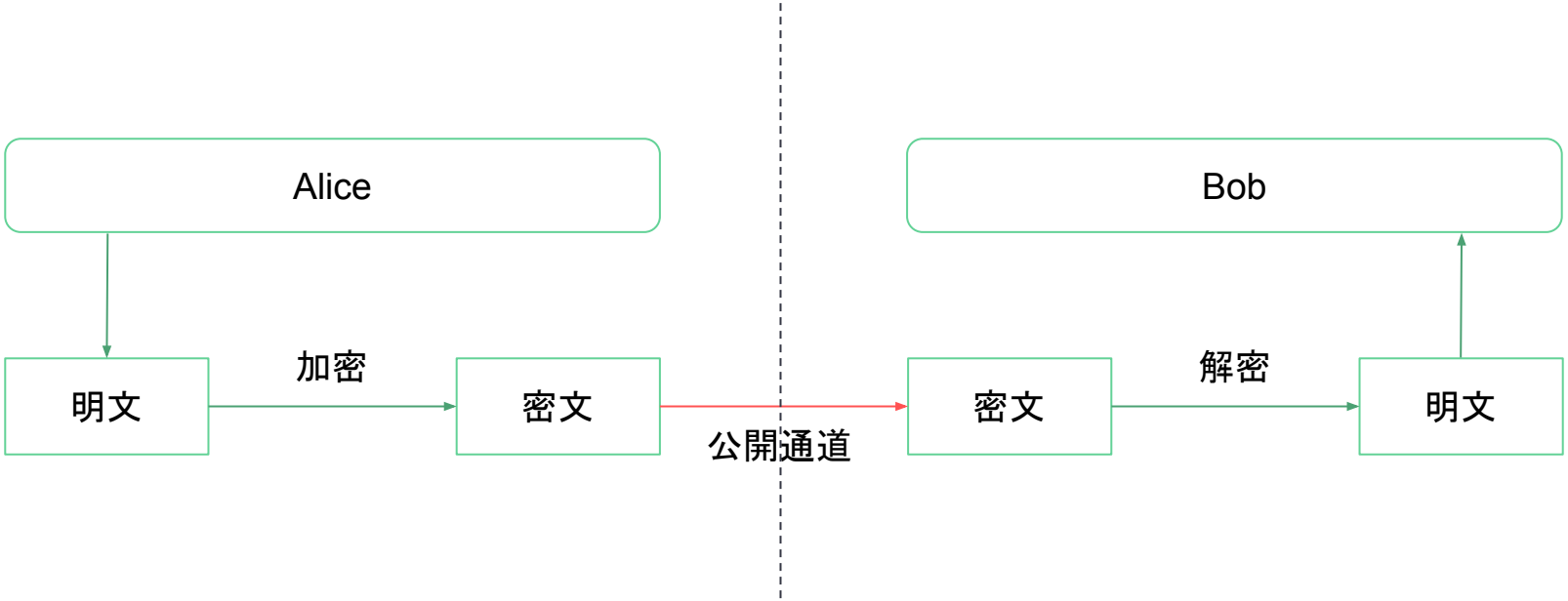
	Hash	MAC	Digital Signature
完整性 Integrity	Yes	Yes	Yes
可驗證性 Authentication	No	Yes	Yes
不可否認性 Non-repudiation	No	No	Yes

完整性：Bob 可以確認 Alice 傳來的訊息是完整的，沒有缺漏或被意外更改

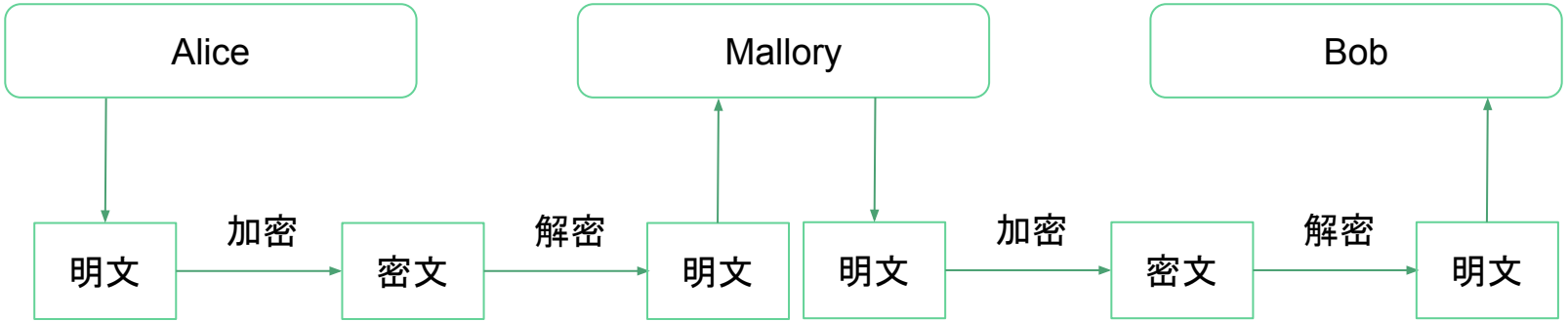
可驗證性：Bob 可以確認訊息確實是 Alice 傳來的

不可否認性：Bob 把訊息拿給第三方，第三方可以確認該訊息是 Alice 傳的，
且 Alice 無法不承認這個訊息源自於她

Man-in-the-middle attack



Man-in-the-middle attack



Alice 如何知道他拿到的公鑰真的是 Bob 的？

Certificate Authority

- 負責身份驗證並發放、管理、註銷憑證的權威機構
- 大家都信任這個機構發放的簽章

亂數

亂數（Random number）

密碼學中，使用亂數的時機：

- 生成金鑰
- 生成 Nonce
- 生成 IV

亂數（Random number）

由 Seed 搭配演算法產出亂數（具有確定性）：

- 偽亂數生產器
  Pseudorandom number generator，簡稱 PRNG
- 密碼學安全偽亂數生成器
  Cryptographically secure pseudorandom number generator，簡稱 CSPRNG

由物理世界的現象產出亂數（不具有確定性）：

- 真亂數生成器
  True random number generator，簡稱 TRNG

亂數（Random number）

- Seed 很重要
- `key = srand(time(NULL))`
- 如果已知 PRNG 與大略的生成時間 Orz
- 請使用 /dev/urandom 和 CryptGenRandom
- 演算法不要亂來，請用 NIST 系列的（DUAL_EC_DRBG 除外）

亂數(Random number)

隨機性:看起來夠亂，沒有規律，所有數字分佈平均
不可預測性:無法從之前的亂數數列猜出下一個亂數的值
不可重複性:以後不可能再有同樣的數列

	隨機性	不可預測性	不可重複性
PRNG	O	X	X
CSPRNG	O	O	X
TRNG	O	O	O

只有 CSPRNG 和 TRNG 可以用於密碼學

那些沒時間介紹但很有趣的東西

- Feistel cipher
- Merkle–Damgård construction
- Length Extension Attack
- PGP
- SSL/TLS
- Public key infrastructure
- Merkle Tree
- Secret Sharing
- Zero-Knowledge Proof
- Blind Signature
- Homomorphic encryption
- ...

密碼學的世界還在快速發展著呢！

慟！怎麼學都學不完啊 Orz

圖：https://github.com/ry/deno/issues/25

Got Your PW

Got Your PW 是一個簡易的資安資源網站，包含常用的工具，適合初學者的入門教材，和許多值得追蹤的資安相關網站。

https://gotyour.pw/

謝謝大家 <(_ _)>

References

- 密碼學與網路安全應用 - 結城浩（旗標）
- Understanding Cryptography: A Textbook for Students and Practitioners - Christof Paar, January Pelzl (Springer)
- https://en.wikipedia.org/wiki/Cryptography
- https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle
- https://en.wikipedia.org/wiki/Substitution_cipher
- https://en.wikipedia.org/wiki/Caesar_cipher
- https://en.wikipedia.org/wiki/Classical_cipher
- https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher
- https://en.wikipedia.org/wiki/Index_of_coincidence
- https://zh.wikipedia.org/wiki/ASCII
- https://reverseengineering.stackexchange.com/questions/2062/what-is-the-most-efficient-way-to-detect-and-to-break-xor-encryption
- https://stackoverflow.com/questions/1135186/whats-wrong-with-xor-encryption
- https://en.wikipedia.org/wiki/XOR_cipher
- https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
- https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
- http://securitylesson.blogspot.com/2012/02/blog-post_15.html

References

- https://en.wikipedia.org/wiki/Public-key_cryptography
- https://en.wikipedia.org/wiki/Hybrid_cryptosystem
- https://en.wikipedia.org/wiki/RSA_(cryptosystem)
- https://blog.cryptographyengineering.com/2018/04/07/hash-based-signatures-an-illustrated-primer/
- https://crypto.stackexchange.com/questions/5646/what-are-the-differences-between-a-digital-signature-a-mac-and-a-hash
- https://en.wikipedia.org/wiki/Padding_(cryptography)
- https://www.owasp.org/images/e/eb/Fun_with_Padding_Oracles.pdf
- https://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator
- https://en.wikipedia.org/wiki/Avalanche_effect
- https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
- https://speakerdeck.com/inndy/no-more-crypto-fails
- https://gpgtools.tenderapp.com/kb/faq/what-is-ownertrust-trust-levels-explained

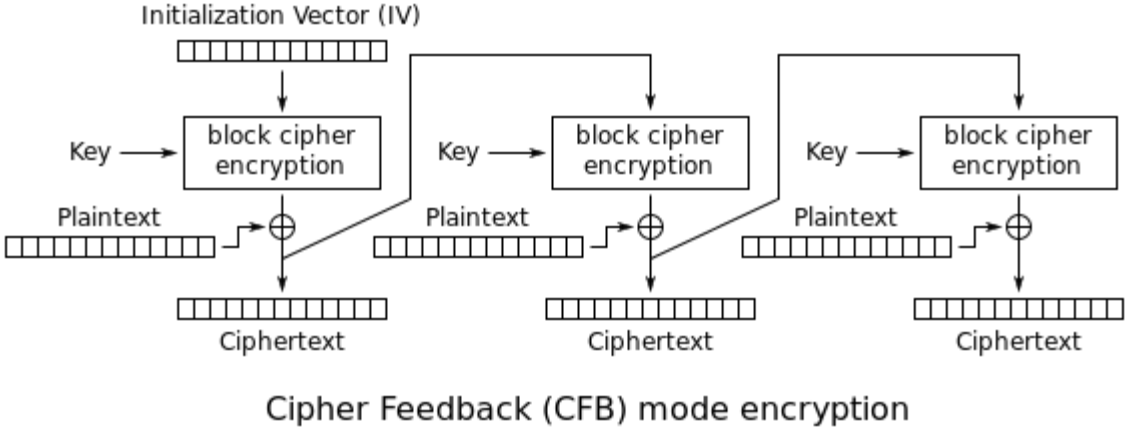
那些被刪掉的簡報

不要自己設計密碼演算法

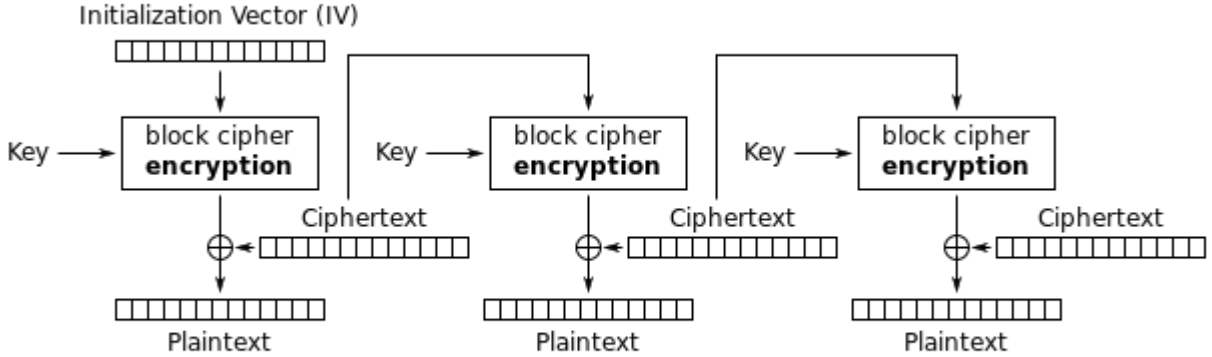
- 現在主流的演算法都是公開並經過許多密碼學家驗證過，絕對比你自己設計的演算法安全
- 所以永遠不要使用自己設計的密碼系統
  - 除非你已經成為密碼學專家了

CFB

- 類似 CBC
- 模擬 Stream Cipher
- IV 作為 Seed
- 當遺失整個 block 時並不影響後續加密
- 重送攻擊
- 請改用 CTR



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

GCM

- Galois/Counter Mode
- GMAC + CTR
- 兼具資料加密（CTR）與驗證（GMAC）
- 一種認證加密（Authenticated encryption）的模式
- Google 與 Facebook 都在用 GCM 做加密