



Penetration Testing Practice

滲透測試實務應用

06 密碼學與密碼分析

孫士勝

Shi-Sheng Sun, Ph.D.

sssun@nccu.edu.tw

課程聲明

Disclaimer

- 本課程所教授之滲透測試相關技術僅能於課堂中進行，若有任何超出範圍的動作，皆屬個人行為。
The penetration testing techniques taught in this course are only to be used within this course. Any actions taken outside of this scope are the sole responsibility of the individual.
- 學員於課後使用任何網路攻擊技術對任何資訊設備進行攻擊，皆屬個人行為。
Students are solely responsible for any network attacks carried out on any equipment after the course using any network attack techniques.

法規名稱：中華民國刑法 EN

法規類別：行政 > 法務部 > 檢察目

所有條文

編章節

條號查詢

條文檢索

沿革

立法歷程(附帶決議)

※如已配合行政院組織改造，公告變更管轄或停止辦理業務之法規條文，請詳見沿革

第二編 分則

第三十六章 妨害電腦使用罪

- 第 358 條 無故輸入他人帳號密碼、破解使用電腦之保護措施或利用電腦系統之漏洞，而入侵他人之電腦或其相關設備者，處三年以下有期徒刑、拘役或科或併科三十萬元以下罰金。
- 第 359 條 無故取得、刪除或變更他人電腦或其相關設備之電磁紀錄，致生損害於公眾或他人者，處五年以下有期徒刑、拘役或科或併科六十萬元以下罰金。
- 第 360 條 無故以電腦程式或其他電磁方式干擾他人電腦或其相關設備，致生損害於公眾或他人者，處三年以下有期徒刑、拘役或科或併科三十萬元以下罰金。
- 第 361 條 對於公務機關之電腦或其相關設備犯前三條之罪者，加重其刑至二分之一。
- 第 362 條 製作專供犯本章之罪之電腦程式，而供自己或他人犯本章之罪，致生損害於公眾或他人者，處五年以下有期徒刑、拘役或科或併科六十萬元以下罰金。
- 第 363 條 第三百五十八條至第三百六十條之罪，須告訴乃論。

ACM 道德與專業行為準則

ACM Code of Ethics and Professional Conduct

- 1.1 增進人類社會福祉(Contribute to society and human well-being.)
- 1.2 避免傷害任何人(Avoid harm to others.)
- 1.3 誠實與值得信任(Be honest and trustworthy.)
- 1.4 公平且無犯罪意圖的行動(Be fair and take action not to discriminate.)
- 1.5 尊重智慧財產權(Honor property rights including copyrights and patent.)
- 1.6 維持智慧財產的完整性(Give proper credit for intellectual property.)
- 1.7 尊重他人隱私(Respect the privacy of others.)
- 1.8 遵守保密原則(Honor confidentiality.)



課前問券

- 密碼學與密碼分析 課前問券
- <https://forms.gle/NvpviCq4xnCzuGQw6>



- [illegible]

我慶宜 今年十八歲 始與大
馬家沈清原 贈與華相府
秋節起暫在帳房候三年
香拂地沈現磨墨等事

密碼學-發展

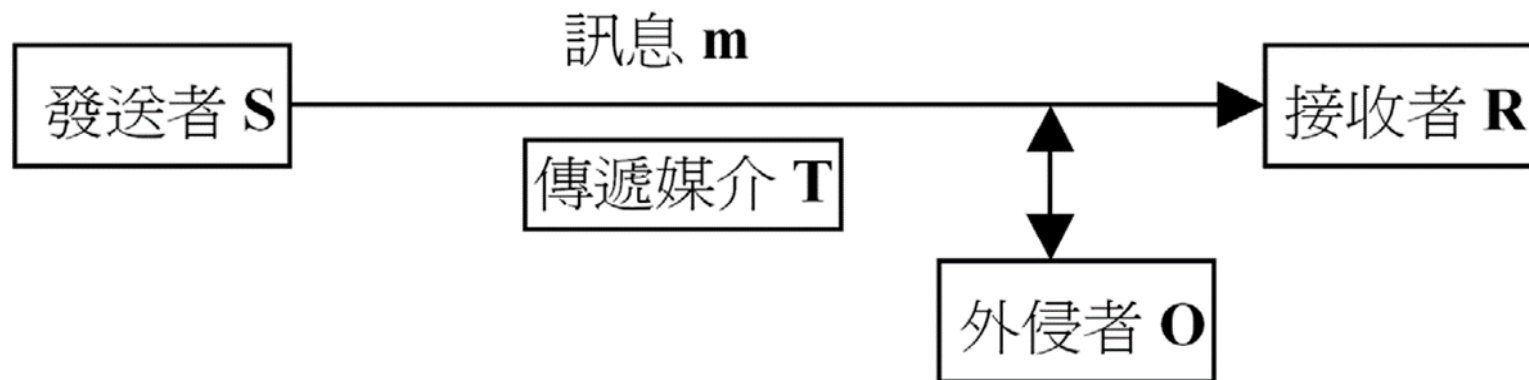
- 數據保密技術
 - 隱藏
 - 特殊通訊技術：方言
 - 數據保密技術：金鑰
- 古今密碼學
 - 古代：運用技術將要傳遞的訊息隱藏
 - 現代：以數學及演算法對資料進行加密以及解密

密碼學的意義 (1/4)

- 在資訊防護上，密碼(Cryptography) 是種有效的手段，使資訊隱匿，可抵制多種安全威脅(Threats)
- 一個良好的密碼方法，將使攔截、更改、捏造等，均難以得逞
- 密碼方法多是來自數學推導的演算法

密碼學的意義 (2/4)

- 資訊的傳遞過程



- 外侵者侵入後，可能會執行攔截、阻斷、攻擊、捏造等破壞行為，若以密碼方法保護訊息，將有極大的防護效果。

密碼學的意義 (3/4)

- 加密

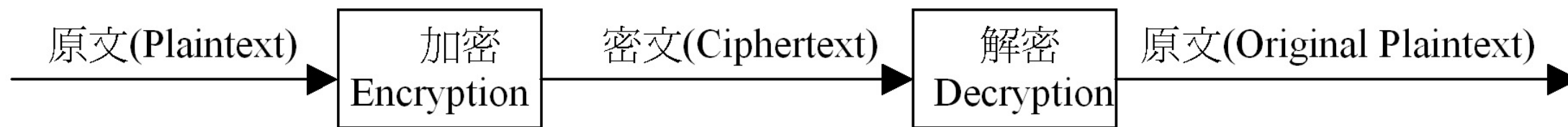
- 加利用某種方式將資訊打散，避免無權檢視資訊內容的人看到資訊的內容，而且只允許「獲得授權的人」，才能看到資訊的內容。
- 「獲得授權的人」是指擁有解密金鑰(key)的人
- 加密是讓毫無相干的人員難以讀取資訊的內容，即使得知加密系統所使用的加密演算法，但沒有金鑰，也就無從得知資訊的內容

密碼學的意義 (4/4)

- 密碼學的三種安全服務
 - 機密性(**Confidentiality**)：不論是在傳輸或儲存設備之中，都可以利用加密隱藏資訊；資料不得被未經授權之個人、實體或程序所取得或揭露
 - 完整性(**Integrity**)：不論是在傳輸或儲存設備之中，都可以利用加密確認資訊的完整性；對資產之精確與完整安全保證，只允許有權限的使用者可以修改資料內容
 - 可說明性/可歸責性(**Accountability**)：加密可以用來確認資訊的來源，且可讓資訊的來源無法否認資訊的出處，又包括以下特性
 - 鑑別性 (**Authenticity**)：確保一主體或資源之識別就是其所聲明者的特性，鑑別性適用於如使用者、程序、系統與資訊等實體
 - 不可否認性 (**Non-repudiation**)：對已發生之行動或事件的證明，使該行動或事件往後不能被否認的能力
- 與資訊安全的三大基本原則有所差異
 - 機密性(**Confidentiality**)
 - 完整性(**Integrity**)
 - 可用性(**Availability**)：已授權實體在需要時可存取與使用之特性，以確保資訊與系統能夠持續營運、正常使用

密碼學基礎術語(Terminology) (1/3)

- 原文(Plaintext)：未作任何變造之原始資訊
- 密文(Ciphertext)：經過加密變造之資訊，對非法攔截者來言，是一串無意義的資訊
- 加密(Encryption)：使用特定密碼方法，將原文變造成密文
- 解密(Decryption)：依特定密碼方法執行，將密文還原成原文



密碼學基礎術語(Terminology) (2/3)

- 加密演算法(Encryption Algorithm)
 - 將原文加密成密文的方法
- 解密演算法(Decryption Algorithm)
 - 將密文還原成原始原文的方法
- 通常演算法都使用「金鑰(Key)」配合執行，假設金鑰為K，則表達式可更改為：

$$\mathbf{C = E(K, P) ;}$$

$$\mathbf{P = D(K, C) ;}$$

$$\mathbf{P = D(K, E(K, P)) ;}$$

密碼學基礎術語(Terminology) (3/3)

- 密碼學(cryptography)：研究加密的原理與方法
- 密碼破解(cryptanalysis)：在不知道金鑰的情況下，分析密碼學演算法並嘗試找出缺陷
- 密碼技術(cryptology)：整合密碼學與密碼破解等領域的技術

春風塵土中行密
吳宮商賈誼碼炎
爐煙雨初學無人
三更相期難窮年
相望中原上臺上
搏考罐廠浞捲簾

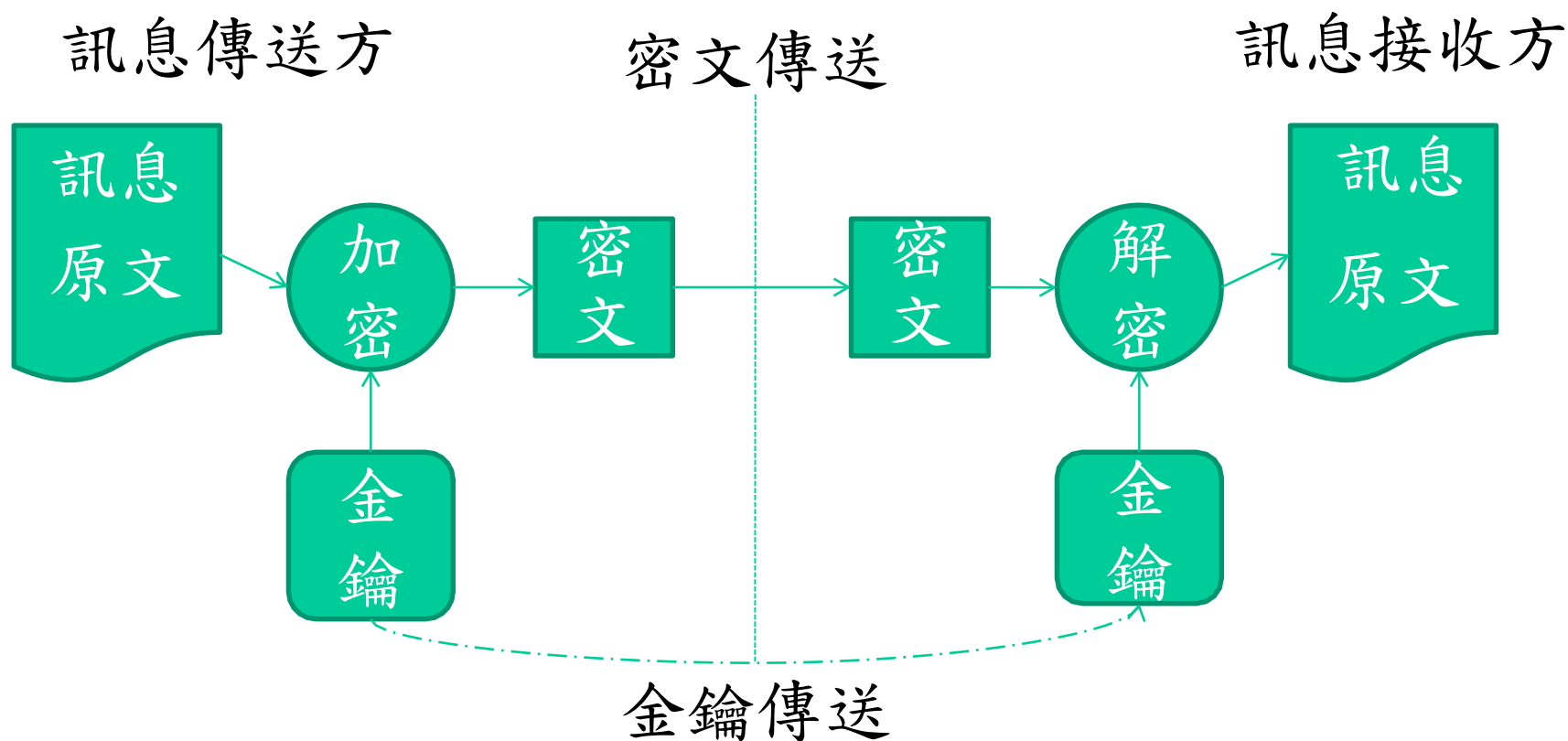
密碼學的分類

依下列三種不同的觀點來分類：

- 加密的方法
 - 取代(substitution)：明文中的每一元素都被對應到另一個元素
 - 置換(transportation)：明文中的每一元素都被重新排列
 - 混用(mixed)：絕大部分的系統都同時引用好幾回的取代與置換，以強化加密的安全性
- 金鑰個數
 - 單一金鑰或私密金鑰：傳送與接收者都持有一把相同的金鑰
 - 雙金鑰或公開金鑰：傳送與接收雙方使用不同的金鑰
- 明文的處理方式
 - 區塊加密(block cipher)：一次處理一大段元素，根據輸入的區塊產生輸出的區段
 - 資料流加密(stream cipher)：讀入連續的元素，但一次只輸出一個元素，直到所有讀入的元素都處理完畢為止

對稱式加密法symmetric-key cryptography

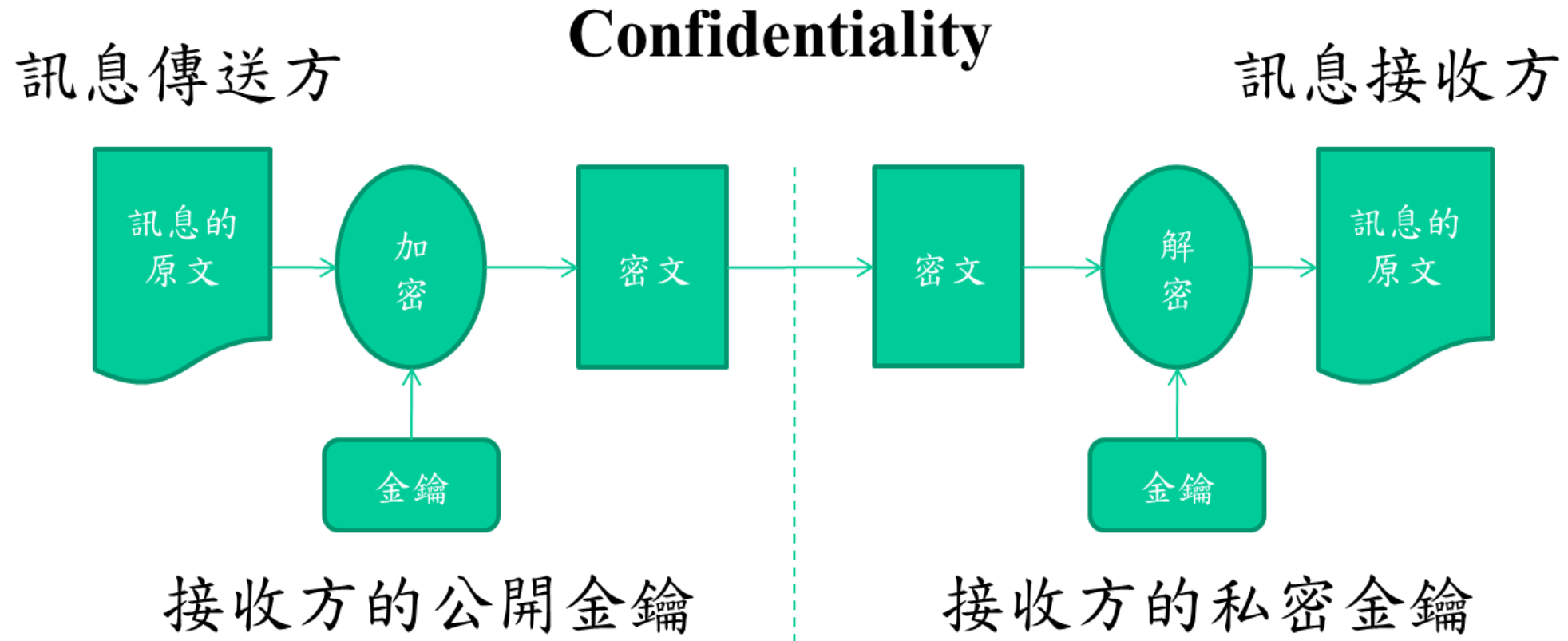
- 對稱式加密法是指加密與解密使用相同但逆向的演算法；且加、解密使用相同的金鑰。如AES、DES、3DES、RC5等，運算速度較快



非對稱式加密法asymmetric-key cryptography

- 非對稱式加密 (asymmetric-key cryptography) 又稱公開金鑰加密 (public key cryptography)。
- 它使用一對數學上相關的金鑰：
 - 私密金鑰 (secret key)：保持機密
 - 公開金鑰 (public key)：可自由傳遞
 - 若以其中一把金鑰加密，就只能使用另一把解密
- Diffie 與Hellman 於1976年提出，主要目的在解決前述對稱式加密中金鑰交換 (key exchange) 的困難

非對稱式加密法asymmetric-key cryptography



密碼學的安全定義

- 絕對安全(unconditionally secure)
 - 不論具備多少計算能力都無法破解
- 計算上的安全(computationally secure)
 - 在計算能力有限的情況下(例如，計算所需的時間比宇宙生成的時間還久)，無法破解此密文

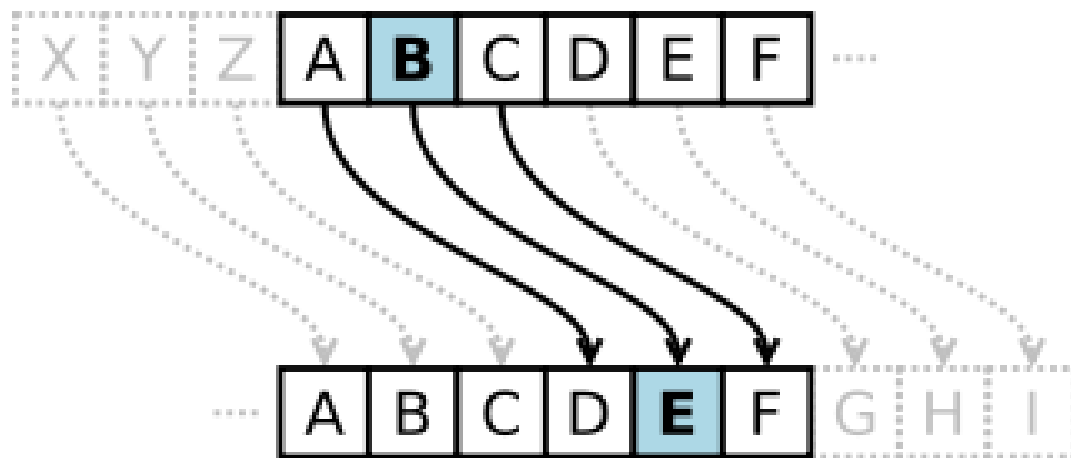


古典密碼學

- 常見的古典密碼學有四種手段：
 - 移位式加密：改變訊息的相對位置
 - 取代式加密：替換訊息的文字
 - 混淆式加密：插入部份垃圾訊息混淆視聽
 - 混合式加密：移位與取代並用

移位式加密(Transposition Cipher)

- 移位加密法(Transposition Cipher)為將明文做置換與重排加密，藉由重新安排字元的順序來隱藏訊息
- 單純的置換加密法很容易被破解，因為頻率分佈與原始文字一樣，所以容易破解



柵欄加密法(Rail Fence Cipher)

- 將明文寫成一連串的深度為K欄對角形式柵欄，然後一列一列地讀出

- 例如，Key=2, plain text為meet me after the toga party

m e m a t r h t g p r y
e t e f e t e o a a t

- 得到密文為mematrhtgpryetefeteoaat

- 例如，Key=3, plain text為HELLOWORLD

H O L
 E L W R D
 L O

- 得到密文為HOLELWRDLO

研究表明 漢字的序順並不一定能影響閱讀

列置換加密法(Row Transposition Cipher)

- 一個較為複雜的機制，在行數固定的情況下，將訊息一系列地寫成矩形的結構
 - 然後在一系列讀出之前，根據某把金鑰來重排行的順序
- 例如plain text為attack postponed until two am xyz
 - key: 4312567

金鑰: 4 3 1 2 5 6 7
明文: a t t a c k p
 o s t p o n e
 d u n t i l t
 w o a m x y z

— 密文: TTNAAPTMTSUOAODWCOIXKNLYPETZ

取代式加密(Substitution Cipher)

- 取代加密為一種古典加密法，加密的方法如下：
 - 將明文中的字元用其他的字元、數字或符號來取代
 - 如果我們將明文視為一連串的字元，那麼取代就是將明文的字元樣式代換成密文的字元樣式
 - 數種取代式加密包括：
 - Caesar 加密法
 - Playfair 加密法

Caesar加密法

- 取代式密碼的存在，已經超過2500年以上，最早利用的範例是在西元600年左右，內容是以顛倒的希伯來文構成的
 - 凱薩大帝(Julius Caesar)也曾經使用稱為Caesar密碼法，又稱為位移加密(Shift Cipher)
- 方法為將每個字母用其後的第三個字母來取代(Key=3)
- 範例：
 - meet me after the toga party
 - PHHW PH DIWHU WKH WRJD SDUWB
- 如果攻擊者充分收集密文，即可破解取代式密碼

Caesar加密法

- Caesar 加密法字元轉換方式定義如下，with a shift of 3:

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
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

- 若將每個字元指定一個數字

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- 則可Caesar 加密法定義如下
 - $C = E(p) = (p + k) \bmod (26)$
 - $p = D(C) = (C - k) \bmod (26)$
- 請破解密文 “GCUA VQ DTGCM” with a shift of 2

Caesar加密法的破解方式

- Caesar 加密法的特色
 - 已知的加密/解密演算法
 - 已知明文使用的語言，而且語言容易辨認
 - 只需嘗試26把金鑰，可利用暴力攻擊法逐一地測試
 - 採用Caesar 加密法的安全度嚴重不足
- 若計入字母不變的種類，僅有26種可能的加密方式
 - A 可能對應到 A,...,Z 其中一個字母
- 給定密文之後，只需測試所有可能的移位距離，可採用暴力法逐一地測試：
 - 因為英文字母只有26個字，加密的可能性只有26種，就期望值而言試13次就可破解
- 位移並不會改變字母的頻率
 - 以英文而言，字母e出現的頻率通常是最高的，如果密文G的頻率最高，可以合理推測key的值為2(從e到G)
 - 除了單獨看字母的頻率之外，還可以進一步分析連字的頻率，如th、er等

Playfair加密法

- 提升安全性的另一個方法是一次加密多個字元
- Playfair 加密法就是一個例子
 - 此法是 Charles Wheatstone 在 1854 年所提出，但是以其朋友 Baron Playfair 的名字來命名



Playfair 金鑰矩陣

- 將雙字元的明文視為單一元素，再將其轉成「雙字元」的密文
- Playfair演算法根據一個關鍵字來產生一個 5x5 階的字元矩陣
- 將關鍵字字元依序填入矩陣中 (刪除重複字元)，再將26字母剩餘字母，填入矩陣中(其中 I 跟 J 視為同一個字元)
- 例如，使用Key: MONARCHY
 - 填完Key後，接著依序將所有字母填入，已填過的字母則跳過

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

Playfair 加密與解密

- 將所要加密的資料，由左而右以兩個為一組，每次加密明文中的兩個字元：
 - 如果這兩個字元一樣，則在其間插入一個 'X'，例如 "balloon" 變成 "ba lx lo on"，此時OO並不會重複。另外II，IJ，JI，JJ也視為重複的兩個字，變成IXJ
 - 如果這兩個字元落在矩陣中的同一橫列，則用它們右邊的字元來取代。例如 "ar" 變成 "RM"。
 - 如果這兩個字元落在矩陣中的同一直行，則用它們下方的字元來取代。例如 "mu" 變成 "CM"。
 - 在其他的情況下，即兩個字在矩陣上不同行、不同列。每個字元都換成與它自己同一橫列，但是與另一個字元同一直行的該字元，即矩形的兩個對角頂點的另兩個字元。例如，"hs" 變成 "BP"，"ea" 變成 "IM" 或是 "JM" (由加密者自行決定)。
 - 最後若只剩一個字，插入/補上字母X

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

Playfair 加密與解密

- 練習：Key=playfairexample
 - Plain Text: HI DE TH EG OL DI NT HE TR EX ES TU MP
 - Cipher Text: BM OD ZB XD NA BE KU DM UI XM MO UV IF

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

Playfair 加密法的安全性

- 較為安全，因為有 $26 \times 26 = 676$ 組雙字元
 - 表格中需要 676 個項目才足以分析，相對地產生更多的密文
- 廣泛地使用許多年，第一次世界大戰時，美軍與英軍都採用此法
- 在給定幾百個字元的密文的情況下，此法仍可破解，因為其中仍存有不少明文的結構

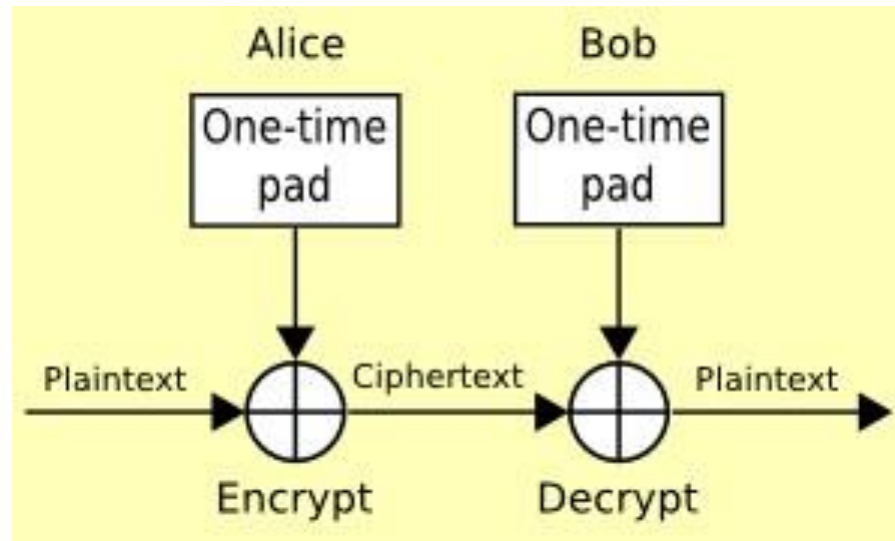
One-Time Pad

- 給予和明文一樣長的密碼本/隨機金鑰。此加密法是安全的，稱為 One-Time Pad (OTP，一次性密碼本)，前提為
 - 密碼產生的方式完全隨機，無法破解的原因是密文與明文間沒有任何統計上的關係
 - 密碼本使用一次後即銷毀
- 對任意的明文與任意的密文之間，都存在一把對應的金鑰
- OTP的金鑰只能使用一次，但是如何安全地產生大量金鑰與分送金鑰，就是一個大問題
- 常利用XOR運算作加/解密，以明文HELLO，KEY為APPLE，轉為ASCII後加密如下

Key	A 01000001	P 01010000	P 01010000	L 01001100	E 01000101
原文	H 01001000	E 01000101	L 01001100	L 01001100	O 01001111
加密	00001001	00010101	00011100	00000000	00001010

One-Time Pad

- 理論上來講，One-Time Pad是唯一無法破解的加密系統
- 但OTP只能使用一次，如果重複使用，就可能進行分析和破解
- 一般來說，OTP並不適用於高流量的網路環境
- 產生大量金鑰與分送金鑰，是最大的挑戰



混合式加密法

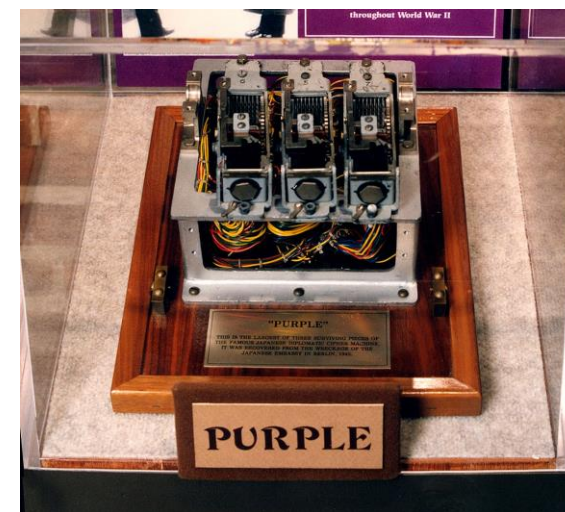
- 因為語言特性的關係，採用取代或置換加密法不夠安全
- 可以採用多重的加密法讓破解更困難：
 - 取代兩次比取代一次複雜
 - 置換兩次比置換一次複雜
 - 取代之後再置換會更加複雜
- 古典與當代加密法的橋樑

旋轉機(rotor)

- 旋轉機(rotor)是一個非常複雜、多重加密的取代加密法
- 在當代加密法出現之前，旋轉機是最常用的混合式加密法
- 旋轉機在第二次世界大戰時被廣泛地使用。
 - German Enigma & Japanese Purple
- 使用一系列的轉軸，每個轉軸就是一個取代加密法，每個字元加密後，這個轉軸都會旋轉一個刻度來改變
- 三個轉軸的旋轉機就提供了 $26^3=17,576$ 個取代法

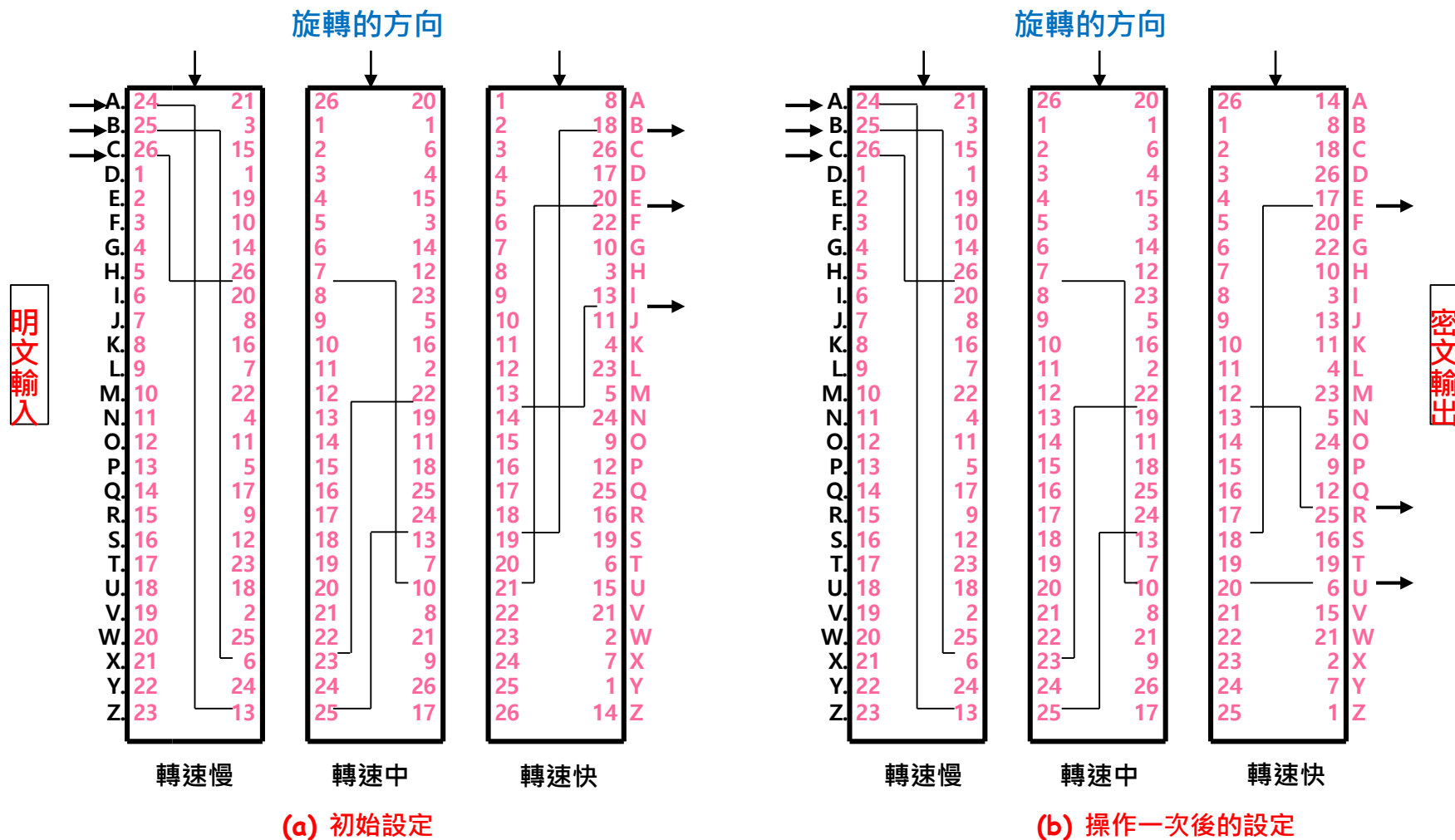


https://en.wikipedia.org/wiki/Enigma_machine



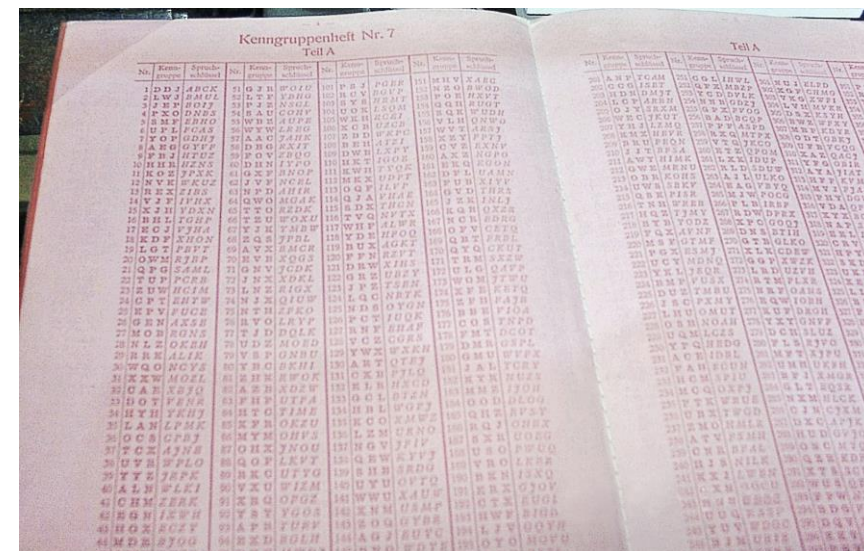
https://en.wikipedia.org/wiki/Type_B_Cipher_Machine

三軸向旋轉機



旋轉機(rotor)

- 二戰時德軍使用了Enigma密碼機做為通訊加解密使用
 - 每條線路中的Enigma都有不同的設定
 - 為了使一條訊息能夠正確地被加密及解密，傳送訊息與接收訊息的Enigma的設定必須相同，旋轉盤必須一模一樣，而且它們的排列順序，起始位置和接線板的連線也必須相同
 - 所有設定都需要在使用之前由在密碼本確認
 - 德國海軍的密碼本使用水溶性的紅色墨水在粉色紙上印製而成，德軍人員就可在它可能被敵人繳獲的時候輕鬆地將它銷毀
 - 右圖的密碼本是盟軍1944從德軍U-505號潛艇上繳獲



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

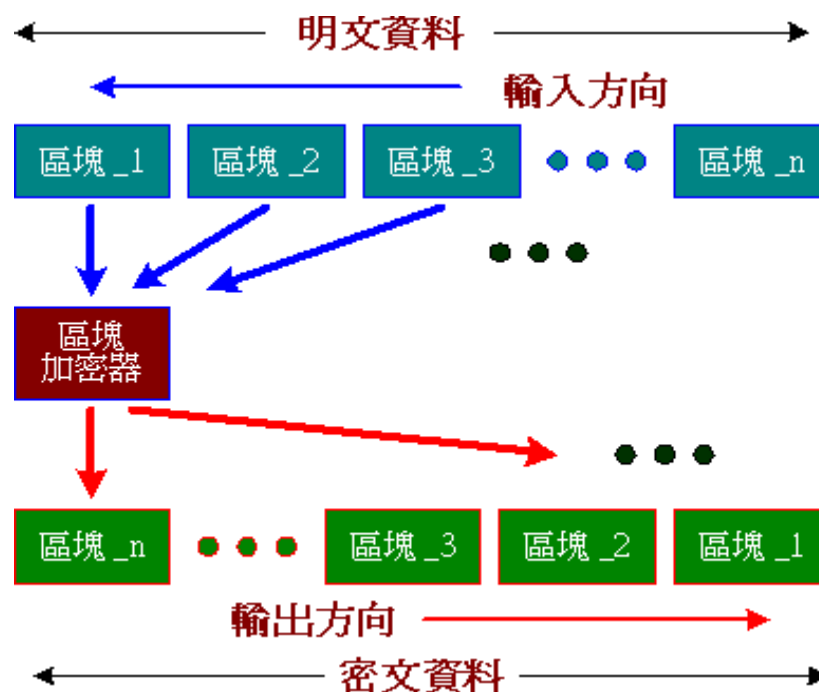


近代密碼學

- 區塊加密、串流加密
- 私鑰加密(對稱式加密)
 - DES
 - AES
- 公鑰加密(非對稱式加密)
 - RSA
 - 橢圓曲線演算法(ECC)
- 雜湊函數Hash

區塊加密

- 使用最廣泛的密碼學演算法類型之一
- 區塊加密法(block cipher)，又稱區段加密法：
 - 將訊息分為區塊來處理，每次對一整個區塊加解密，每一次對多個字元(64位元或更多)進行取代



區塊加密的原理

- 許多對稱式區塊加密法都是以「Feistel 加密法」的結構為基礎
- 其想法源自於對「有效地解密」的需求，也就是讓一個明文區塊產生唯一的密文區塊，讓加解密程序為可逆(reversible)
- 區塊加密可視為極大量的取代加密法
- 對 64 位元的區塊加密來說，需要 2^{64} 個項目的表格以用來對應加解密的想法，其原理來自於前述的取代及置換的混合式(mixed)加密法

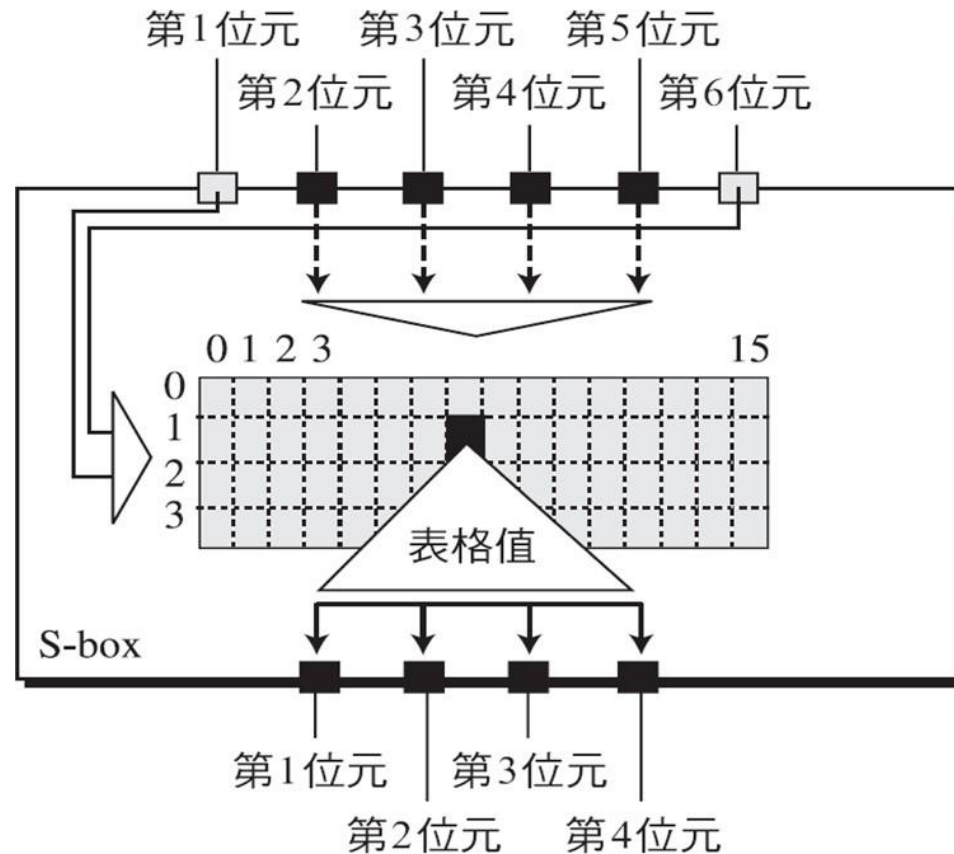
區塊加密的原理

Claude Shannon 與取代-重排加密法

- Claude Shannon 在 1949 年提出了取代-重排 (S-P) 網路的想法，成為當代區塊式加密法的基礎
- S-P 網路是以下列兩種基本密碼學運算為基礎
 - Substitution(取代，S-box)
 - Permutation(重排，P-box)
- Shannon 建議結合各項元素來達成下列目標：
 - 擴散(diffusion)：將一段密文內的明文統計結構消除
 - 混淆(confusion)：讓密文與金鑰間的關係愈複雜愈好
- one-time pad 在統計上可以達成這兩項要求

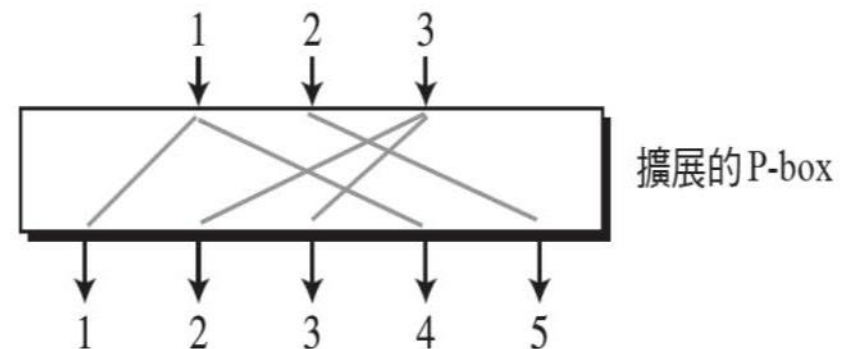
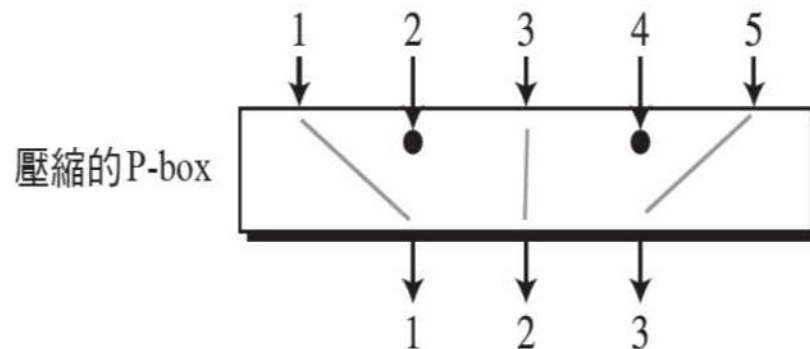
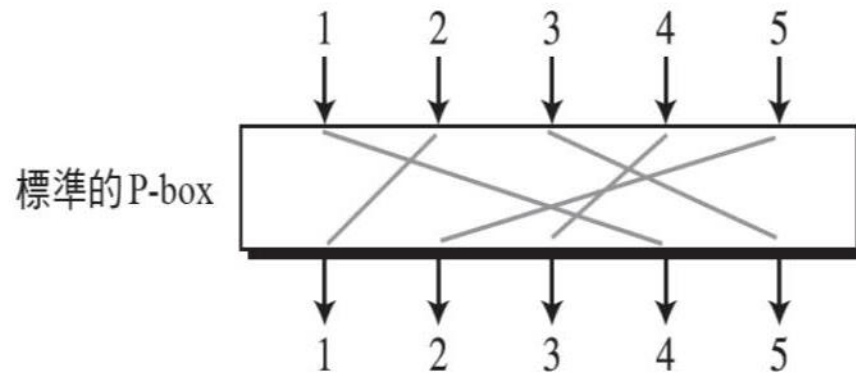
Substitution(取代, S-box)

- S-Box接受特定數量的輸入位元 m ，並將其轉換為特定數量的輸出位元 n ，其中 n 不一定等於 m

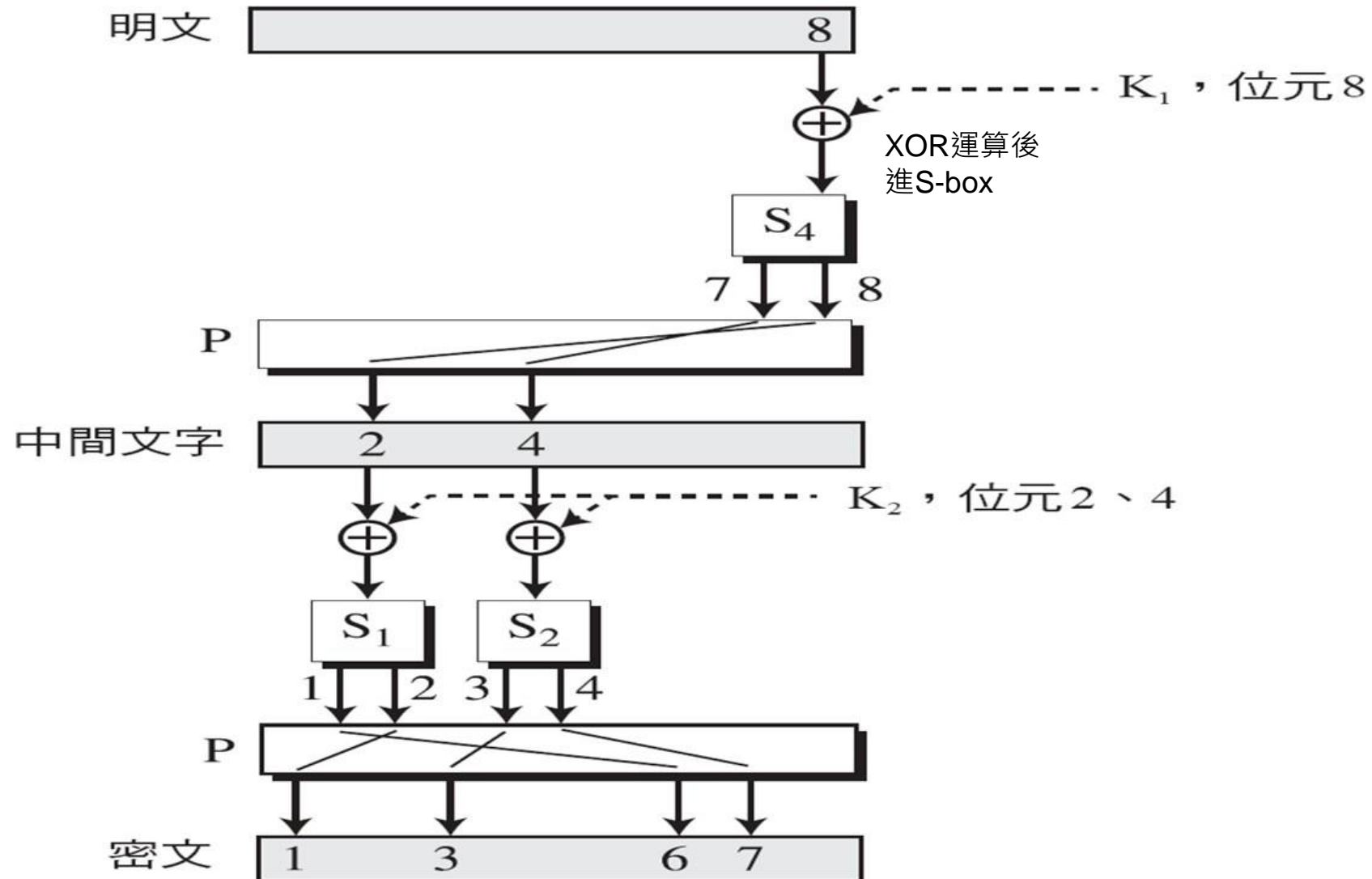


Permutation(重排 , P-box)

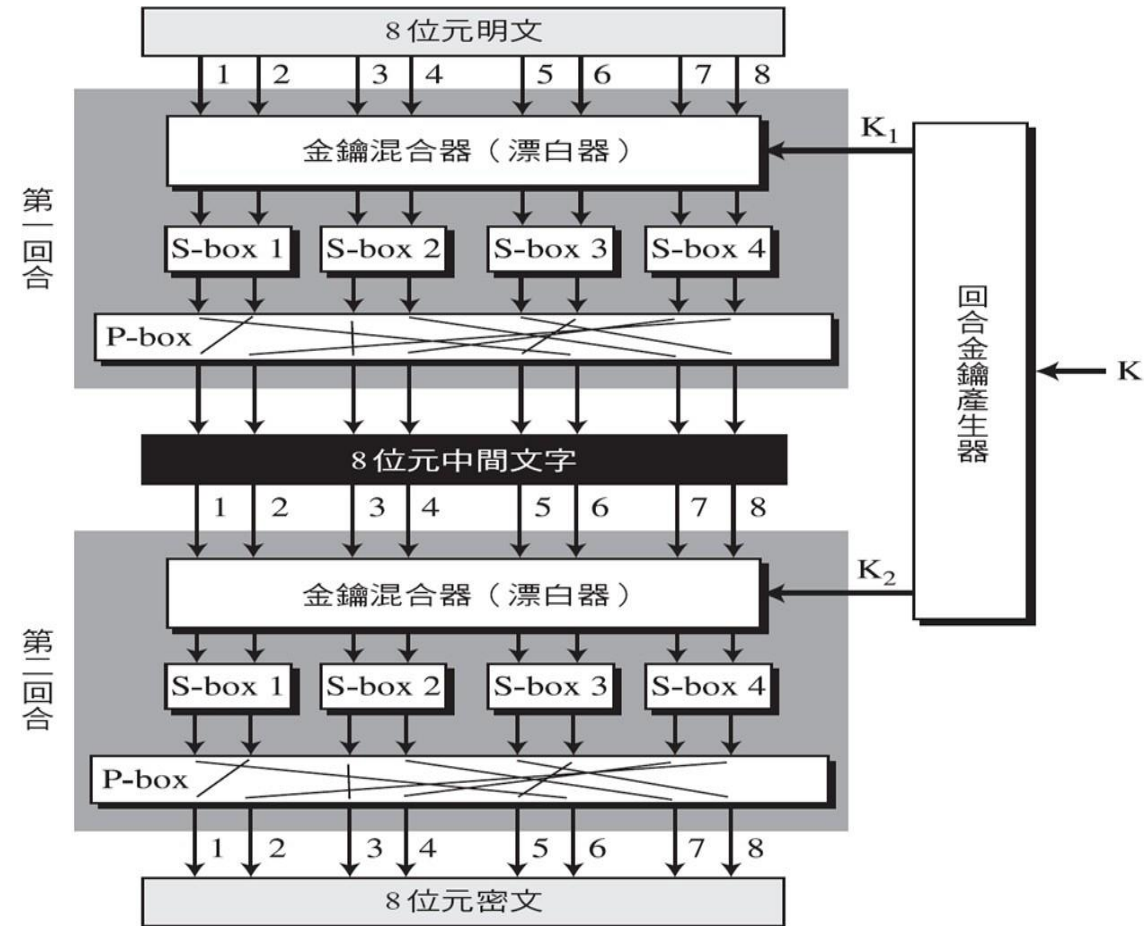
- P-box主要分為三類
 - 標準的：輸出位元數等於輸入位元數，此類為可逆的
 - 壓縮的：輸出位元數比輸入少
 - 擴張的：輸出位元數比輸入多



混淆(confusion)

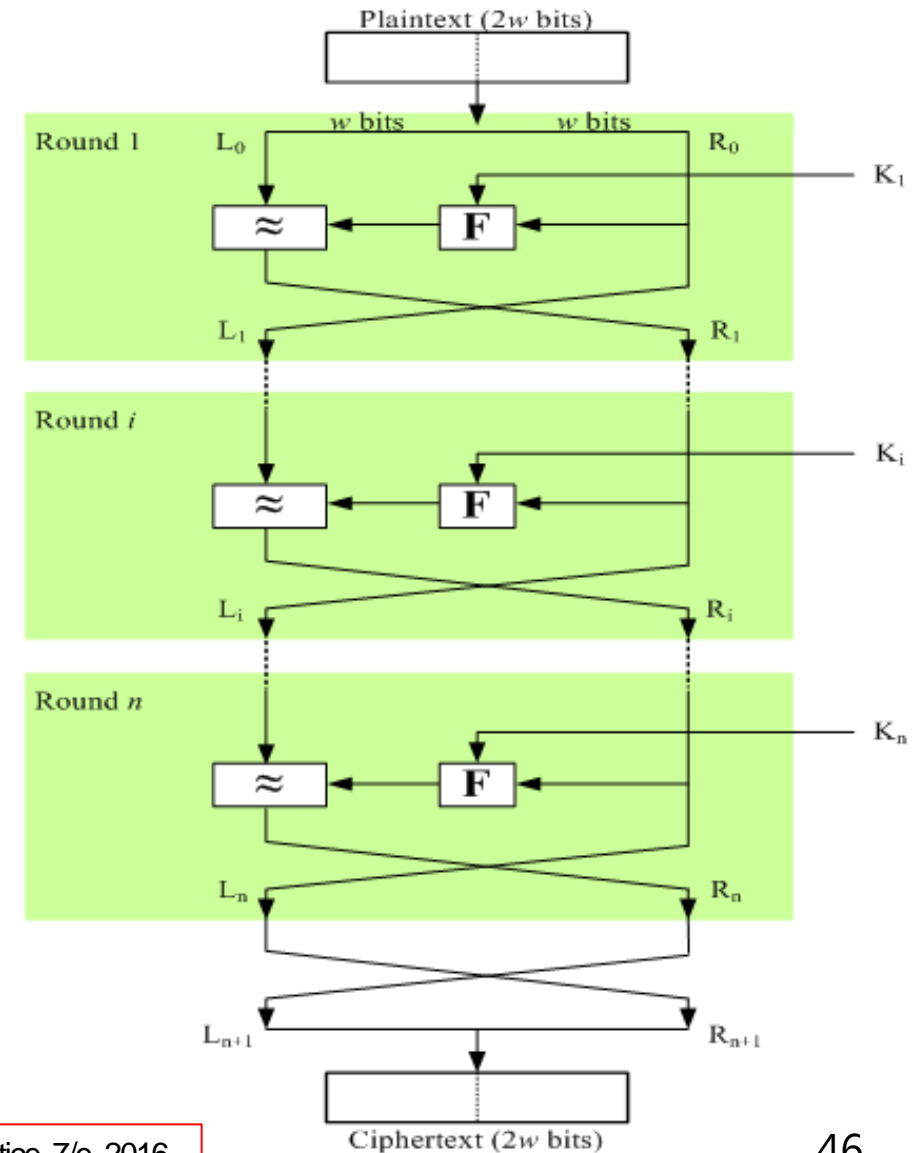


擴散(diffusion)



標準Feistel區塊加密

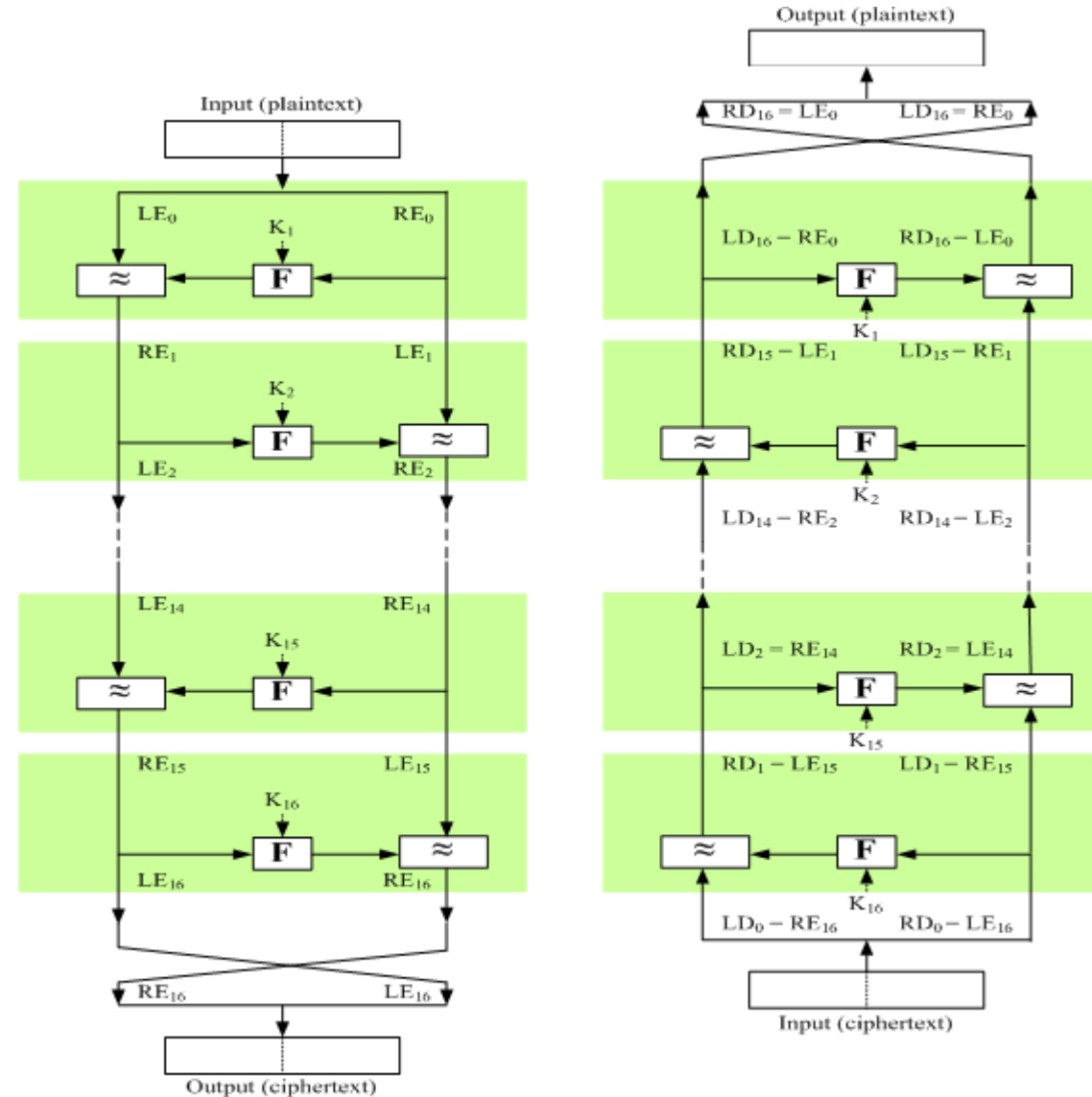
- Feistel 加密結構
- Horst Feistel 1973年在IBM提出了Feistel加密法
 - 以不可逆的混合式加密為基礎
- 將輸入的區段分為兩半
 - 分多個回合(round)來處理
 - 每回合左半部資料會執行一次取代運算
 - 取代運算會將右半部回合函式(round function) F 的結果，以XOR運算方式與左半部資料結合起來
 - 然後交換左右兩半資料
- 實踐 Shannon 的取代-重排網路概念



標準Feistel區塊加密

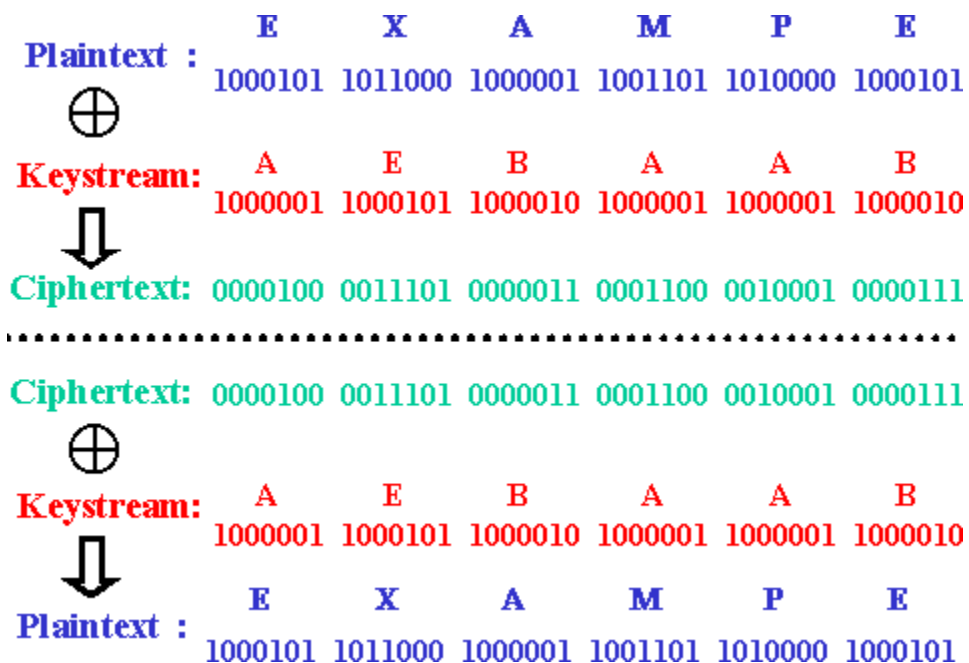
- 標準Feistel區塊加密的特性包括
 - 區段大小(block size)
 - 增大區段可以提升安全性，但是加解密會變慢
 - 金鑰長度(key size)
 - 增加長度可以提升安全性，使得暴力搜尋金鑰變得更困難，但是加解密會變慢
 - 回合數(number of rounds)
 - 增加回合數可以提升安全性，但是加解密會變慢
 - 子金鑰的產生(subkey generation)
 - 複雜的產生方法可以使分析變困難，但是加解密會變慢
 - 回合函式(round function)
 - 複雜的函式可以使分析變困難，但是加解密會變慢

Feistel加解密程序



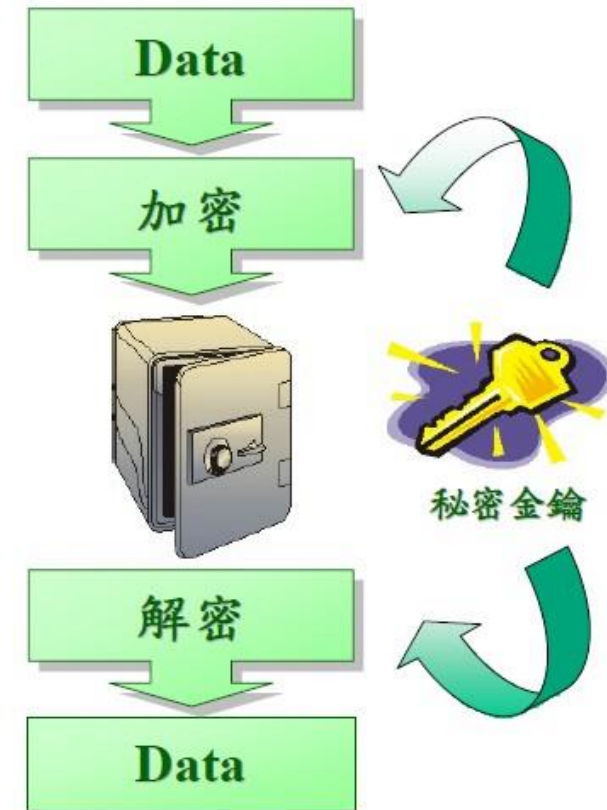
串流加密

- 在加解密時，一次處理一個bit或byte
- 串流加密法適用於通訊上，因為其電路設計較區塊加密法簡單，加密速度比區塊加密法快很多



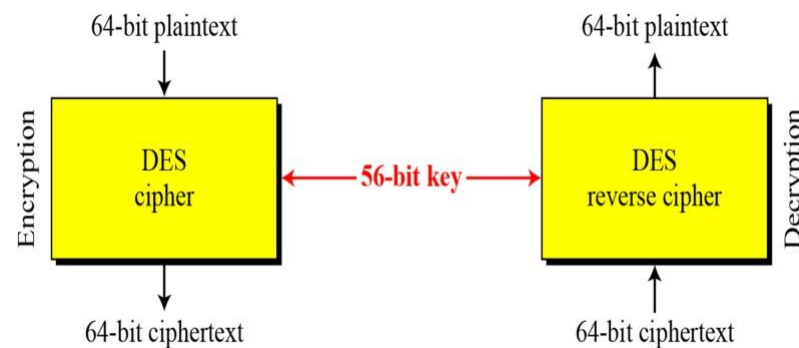
私鑰加密(對稱式加密, Symmetric Encryption)

- 加密與解密用同一把金鑰
 - EX:
 - 加密 $10110100 \oplus 11100001(\text{金鑰}) = 01010101$ (密文)
 - 解密 $01010101 \oplus 11100001(\text{金鑰}) = 10110100$ (明文)
 - DES(Data Encryption Standard)
 - 3DES(Triple DES)
 - AES(Advanced Encryption Standard)



私鑰加密(對稱式加密, Symmetric Encryption)

- DES
 - 每次加密或解密的區塊大小均為64位元
 - 8位元錯誤更正, 實際可用56位元
 - 傳送端與接收端需有同一把金鑰
- 3DES (Triple DES)
 - 由於DES安全性不足, 研究發現DES安全性可經由重複運算而產生
- AES(Rijndael加密法)
 - 目前最常用, 已取代DES、3DES
 - 美國公開甄選, 最好的加密方式之一
 - 區塊長度為128位元, 密鑰可選128、196、256位元, 安全性正比於密鑰長度

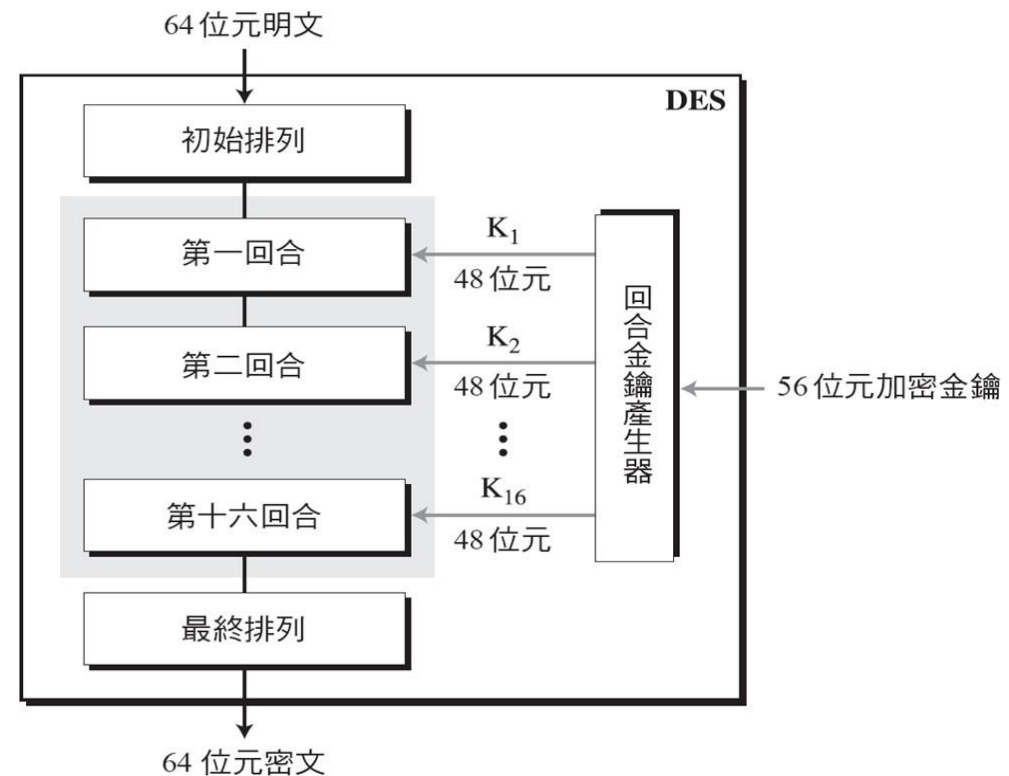


DES

- DES(Data Encryption Standard)是由IBM在1970年代初期發展出來的演算法
- 經過NSA審核、修正、認可之後， NIST (United States National Institute of Standards and Technology，美國國家標準與技術協會)，在1977年正式採用並做為DES的加密標準 FIPS PUB 46
- 於1983、1988、1993和1999年再次認定此標準

DES

- DES使用56位元金鑰，且使用7個8位元的位元組(每個位元組的第8個位元是做為同位元檢查)做為金鑰的內容。
- DES屬於區塊加密(block cipher)，一次處理 64位元明文。
- DES密碼共有16個循環，每個循環都使用不同的次金鑰(subkey)，且每一個金鑰都會透過自己的演算法取得16個次金鑰



DES

DES 的強度

- 具有56 位元長度的金鑰，相當於 $2^{56} = 7.2 \times 10^{16}$ 種可能的值
- 以目前電腦的運算，暴力攻擊法似乎使得破解成為可能，前提是我們必須能辨識明文
- 1997 年科學家透過網路合作，使用結合distributed.net在39天破解
- 1998 年EFF使用25萬美金打造了特定硬體- Deep Crack，在56小時內破解
- 1999 年科學家結合以上的方式，結合distributed.net的算力，在 22 小時15分鐘內破解

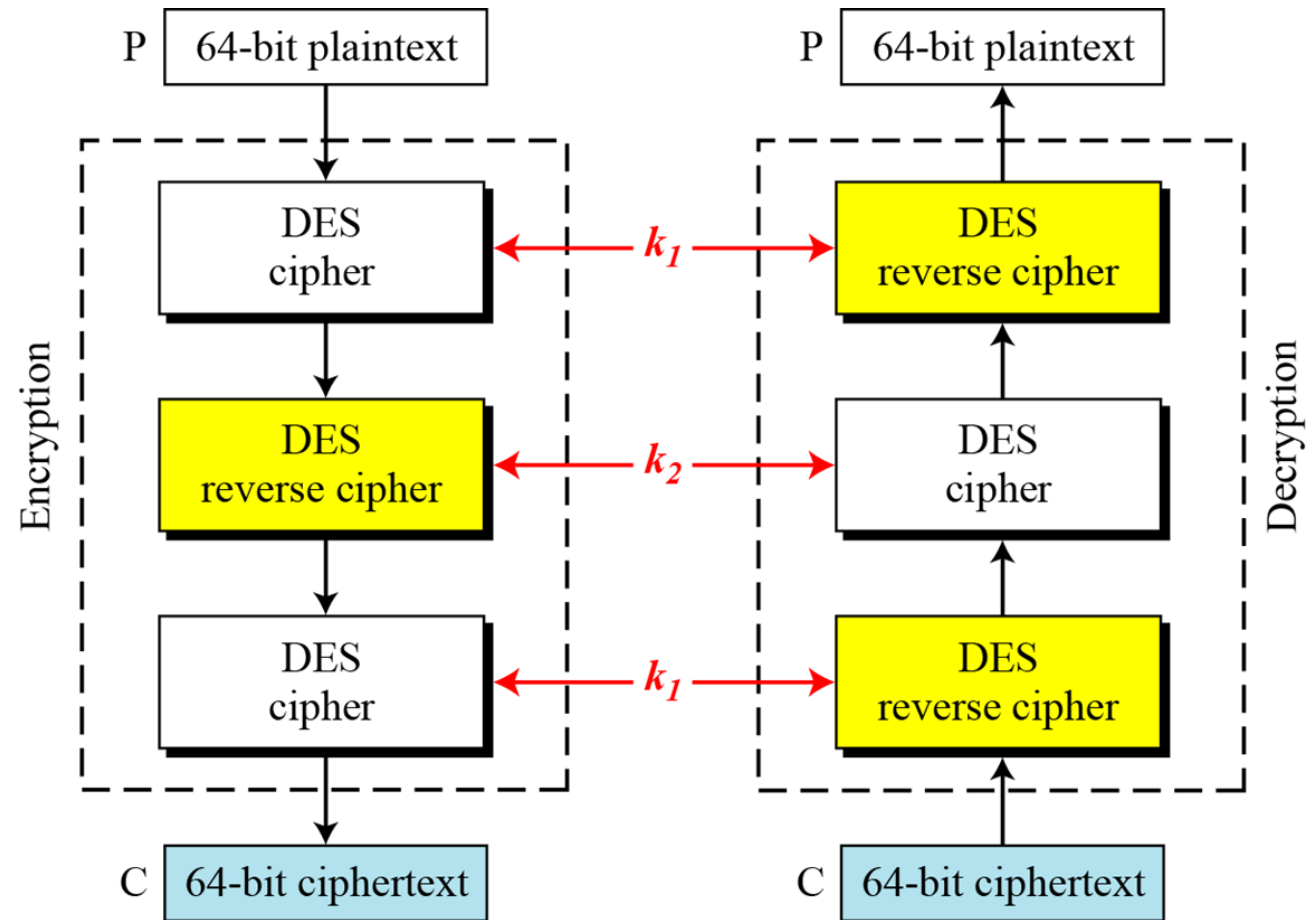


<https://stats.distributed.net/>

3DES

- 在1992年研究人員發現重複使用DES，可建立更牢靠的加密法則，因而產生了 3DES(Triple DES，或稱TDES)
- 3DES可以使用2組(k_1/k_3 、 k_2)或3組金鑰(k_1 、 k_2 和 k_3)
 - 如果僅僅使用兩組金鑰，那麼 k_3 和 k_1 是一樣的，僅 k_2 不同
- 透過混合式加密法，將資訊作取代與重排，因此3DES會比一般DES來得牢靠，成為後續多數人選用的方案

3DES



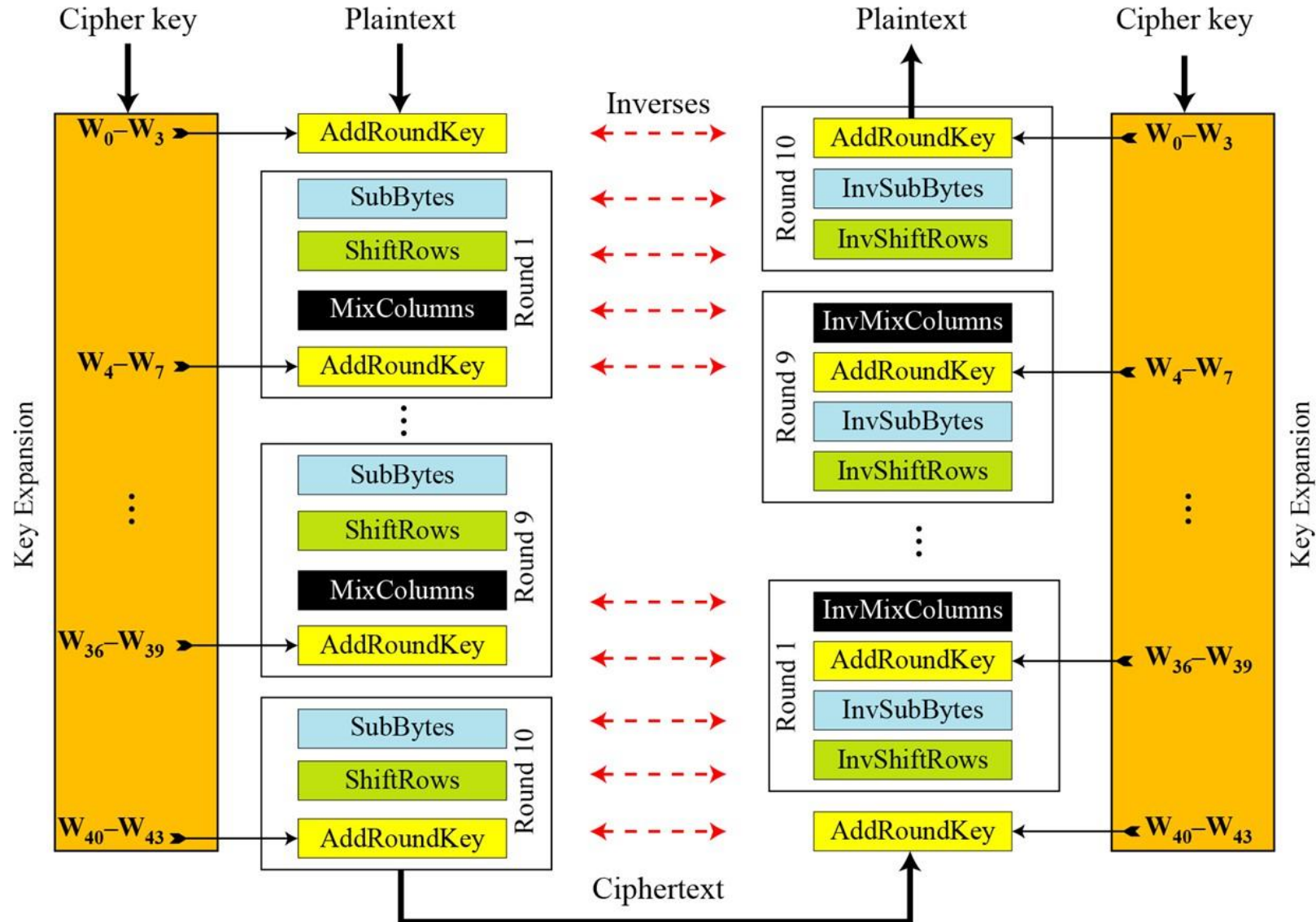
3DES

- 使用2把金鑰的3DES
 - 整個程序需要加密三次
 - 看起來似乎需要三把金鑰
 - 但採用 E-D-E 的方式，只要兩把金鑰
 - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
 - 加密與解密在安全性上來說是等價的
 - 已標準化為 ANSI X9.17 & ISO 8732
 - 目前尚未見到可行的破解方式
- 使用3把金鑰的3DES
 - 雖然現今仍無實際的破解方式，但是使用兩把金鑰的 3DES 安全度較低
 - 如果採用三把金鑰的3DES 就可以更安全
 - $C = E_{K3}[D_{K2}[E_{K1}[P]]]$
 - 已被某些網路協定採用，例如PGP以及S/MIME。

AES

- 在 AES 出現之前，大多是用多次的 DES 加密來處理
- 為了取代DES，NIST在1997年公布AES (Advanced Encryption Standard，進階加密標準)徵選活動。
- 在2000十月，NIST宣佈來自比利時的兩位密碼學家 - Joan Daemon-Vincent Rijmen，結合兩位作者的名字所提出的Rijndael演算法贏得這項競賽，並於2001年十一月完成評估，發佈為FIPS PUB 197標準
- AES牢靠度高、適用於高速網路、可利用硬體加速等因素，都是這個演算法獲選的原因

AES



AES

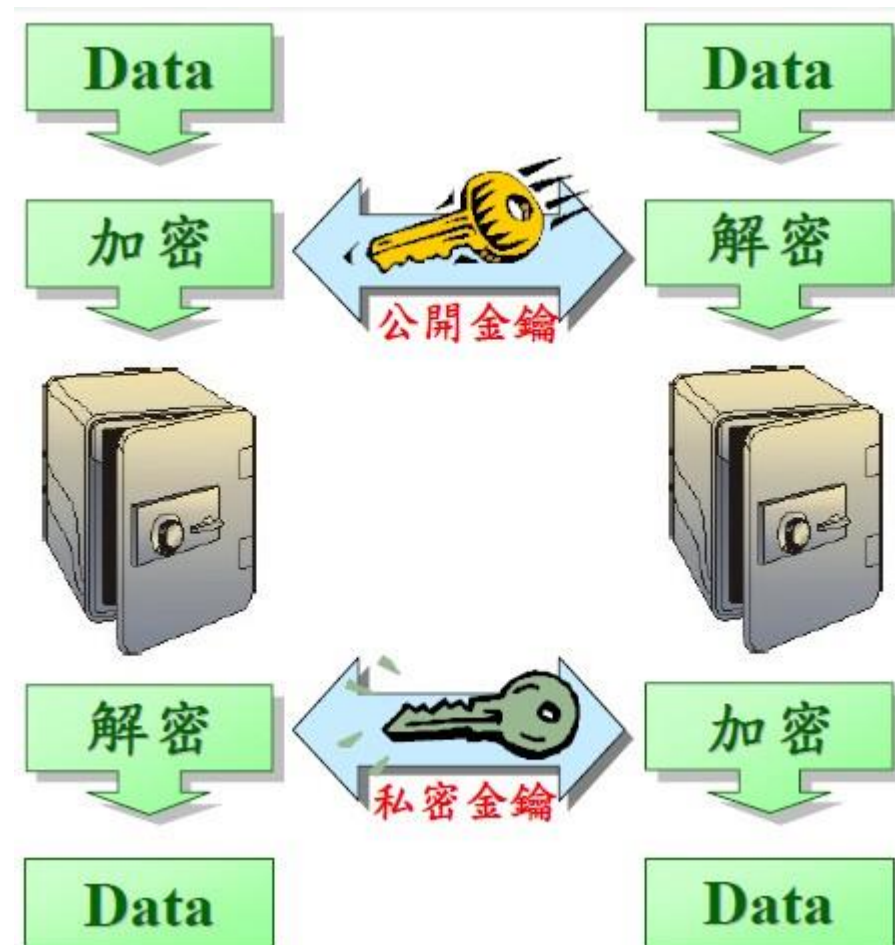
- 採用私密金鑰的對稱式區段加密法
- 資料區段為 128 位元，金鑰為 128/192/256 位元
- 運算速度比 3DES 更強更快。
- 具有完備的規格與設計細節供參考
- AES可採用 C 與 Java 來實作
- 截至目前為止，AES仍然沒有暴力破解金鑰長度的計算方法
- 演算法的循環是依據原文的區塊大小和金鑰的長度而定，演算法之中會包含10到14個循環
- 在認可Rijndael演算法之後，許多資訊系統已經開始出現Rijndael演算法。而且也成了取代3DES的最佳選擇
- 可以抵禦已知的攻擊法
- 在許多 CPU 上都可以快速執行，設計簡單且程式碼小

其他私鑰加密(對稱式加密)

- IDEA (International Data Encryption Algorithm，國際資料加密演算法)：由瑞士發展出來，IDEA使用128位元的金鑰，也可以應用在PGP
- RC5：RC5是由MIT的Ron Rivest發展出來的，它允許變動金鑰的長度
- Skipjack：Skipjack是由美國政府使用Clipper Chip發展出來的，它使用80位元金鑰長度，在可見的未來將會變得不夠牢靠
- Blowfish：Blowfish允許使用的金鑰長度最長可達448位元，在32位元處理器上執行會有最佳化的執行效率
- Twofish：Twofish使用128位元區塊，且可使用128位元、192位元或256位元金鑰
- CAST-128：CAST-128使用128位元金鑰，可應用在較為新版的PGP之中

公鑰加密(非對稱式加密)

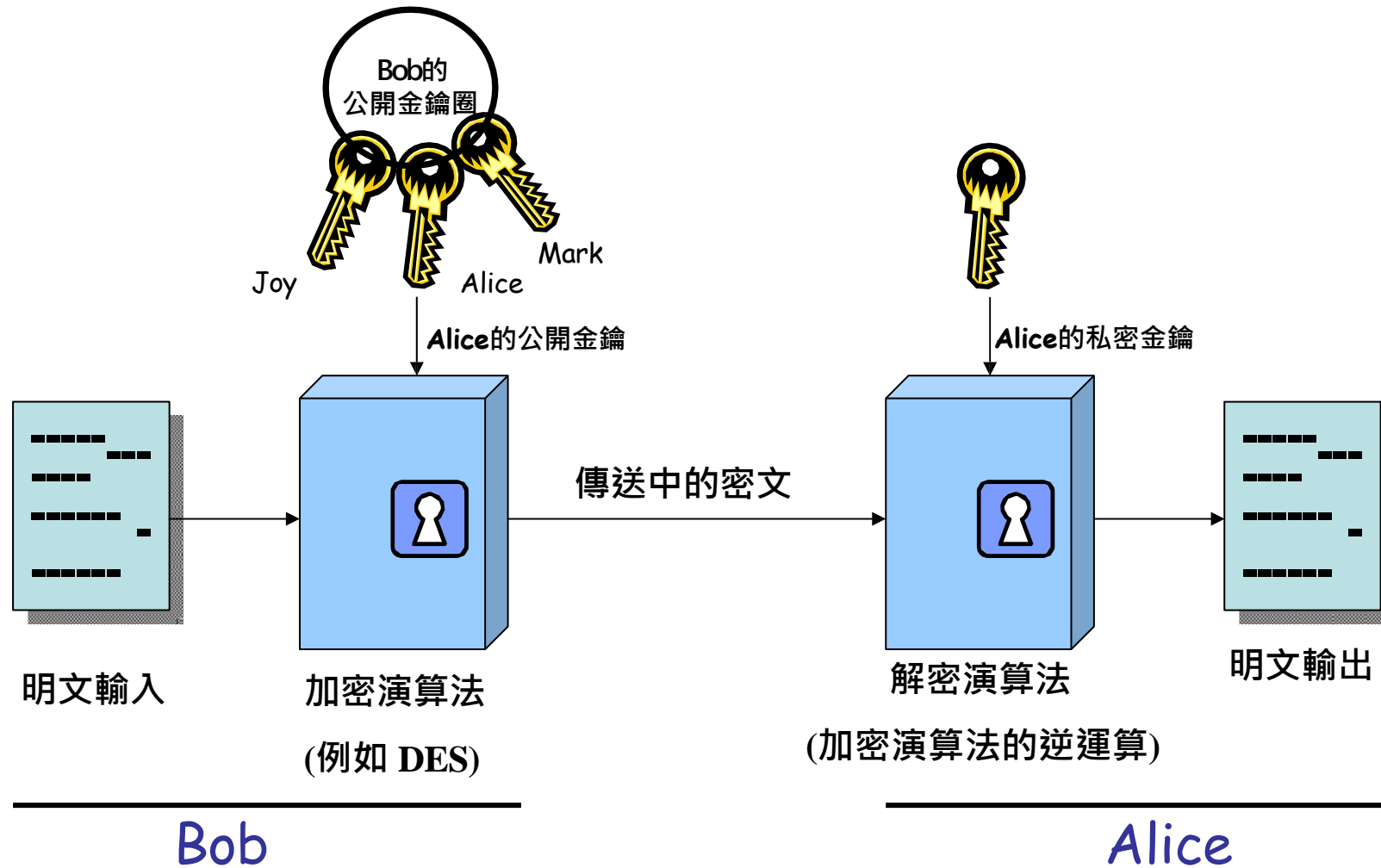
- 又稱非對稱式加密(Asymmetric encryption)
- 1976年由Whitfield Diffie 及 Martin Hellman 提出加密與解密使用不同鑰匙的概念
- 每個使用者有一把公鑰，一把私鑰
- 常見演算法：
 - RSA-分解大質數
 - 橢圓曲線演算法(ECC)



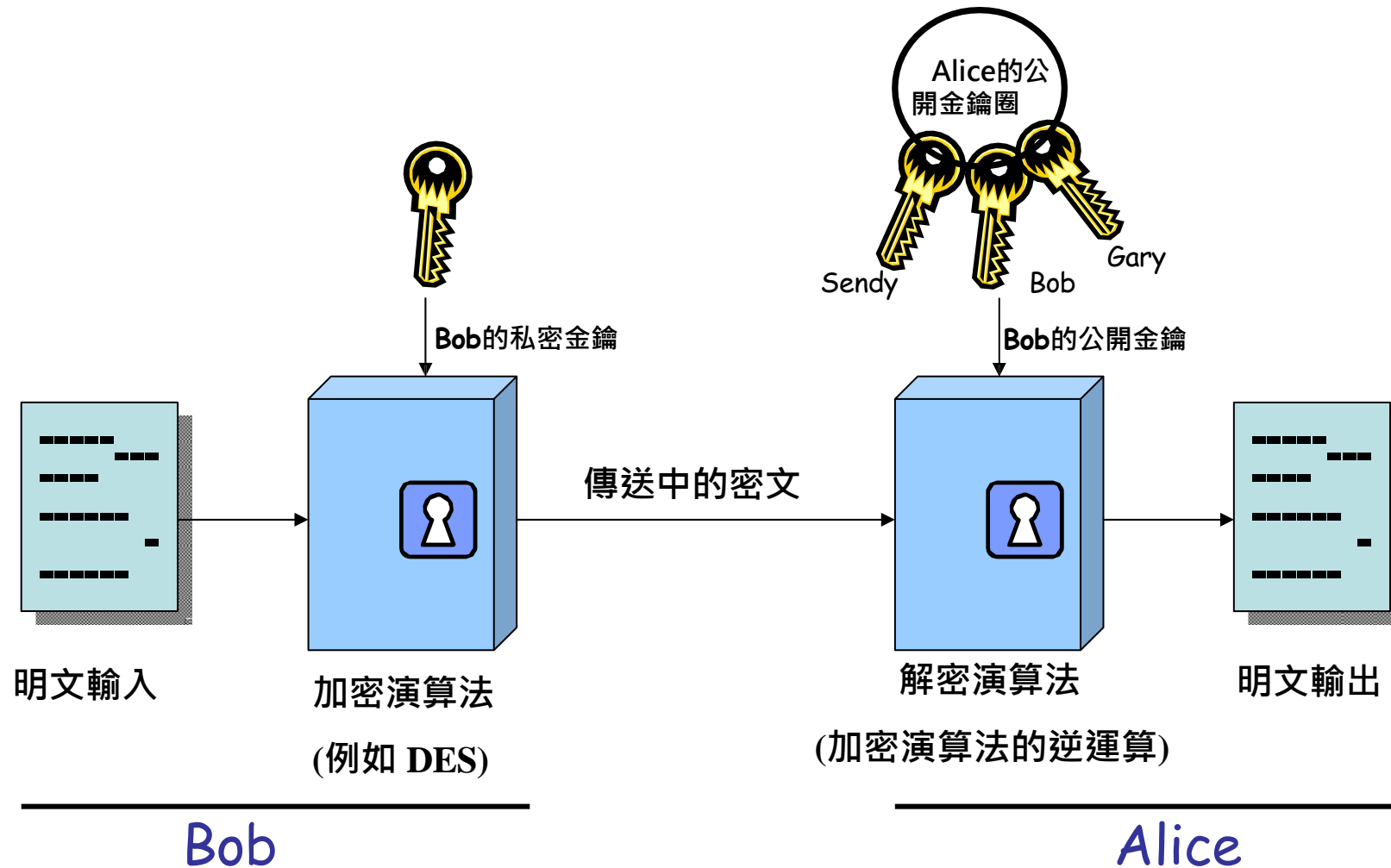
公鑰加密(非對稱式加密)

- 公開金鑰加密法使用兩把金鑰，一把公開金鑰與一把私密金鑰
 - 因為兩把金鑰不對等，所以稱為非對稱式(asymmetric)加密
 - 運用數論(number theory)的觀念應用在函式上
 - 一把公開金鑰，任何人都知道這把金鑰，可以用來對訊息加密或是驗證簽章
 - 一把私密金鑰，只有接收者知道這把金鑰，用來對訊息解密或是簽署(產生)簽章
 - 因為兩把金鑰為非對稱金鑰，故加密或驗證簽章的一方，無法解密或產生簽章
- 為什麼需要公開金鑰加密法？
 - 因為下列兩項關鍵議題才發展出來的：
 - 金鑰分送：如何在不需要 KDC(Key distribution center)的情況下，建立安全的通訊
 - 數位簽章：如何驗證訊息是否為傳送者所送出的

公鑰加密(非對稱式加密)：加密與解密



公鑰加密(非對稱式加密)：簽章與驗章



RSA演算法

- 由Rivest、Shamir及 Adleman 於 1977 年在 MIT 所發明的
- RSA為最多人知道且使用最廣的公開金鑰機制
- 以質數同餘有限體(Galois)的指數運算為基礎
- 若使用很大的整數(≥ 1024 位元)，就會難以找出最大公因數
- 安全性建立在分解大數的困難度上

RSA 金鑰設定

- 使用者以下列方式來產生公開與私密金鑰
- 任意選取兩個大質數 p 與 q ，計算其模數 $N=pq$
 - 請注意 $\varphi(N)=(p-1)(q-1)$
- 任意選取加密金鑰 e
 - 此處 $1 < e < \varphi(N)$, $\gcd(e, \varphi(N))=1$
- 解下列等式來求其解密金鑰 d
 - $ed=1 \bmod \varphi(N)$ 且 $0 \leq d \leq N$
 - 指數 e 與 d 互為反元素，因此可用反元素演算法來求另外一個
- 發佈公開加密金鑰： $KU=\{e,N\}$ ，保存解密金鑰： $KR=\{d,N\}$

產生金鑰	
選取 p,q	p 與 q 都是質數, $p \neq q$
計算 $N=pq$	
計算 $\varphi(N)=(p-1)(q-1)$	
選取整數 e	$\gcd(\varphi(N),e)=1$; $1 < e < \varphi(N)$
計算 d	$d \equiv e^{-1} \bmod \varphi(N)$
公開金鑰	$KU=\{e,N\}$
私密金鑰	$KR=\{d,N\}$

RSA加解密運算

- 針對機密性的基本加密演算法如下
 - 密文 $C = (原文M)^e \bmod N$
 - 原文 $M = (密文C)^d \bmod N$
 - 公開金鑰 $KU = \{e, N\}$
 - 私密金鑰 $KR = \{d, N\}$
- 上述算式的困難點在於：即使取得 e 和 N 也難以計算出 d
- 此公式假設金鑰對的擁有者隱密地保存私密金鑰，並公佈其公開金鑰

加密	
明文	$M < N$
密文	$C = M^e \pmod{N}$

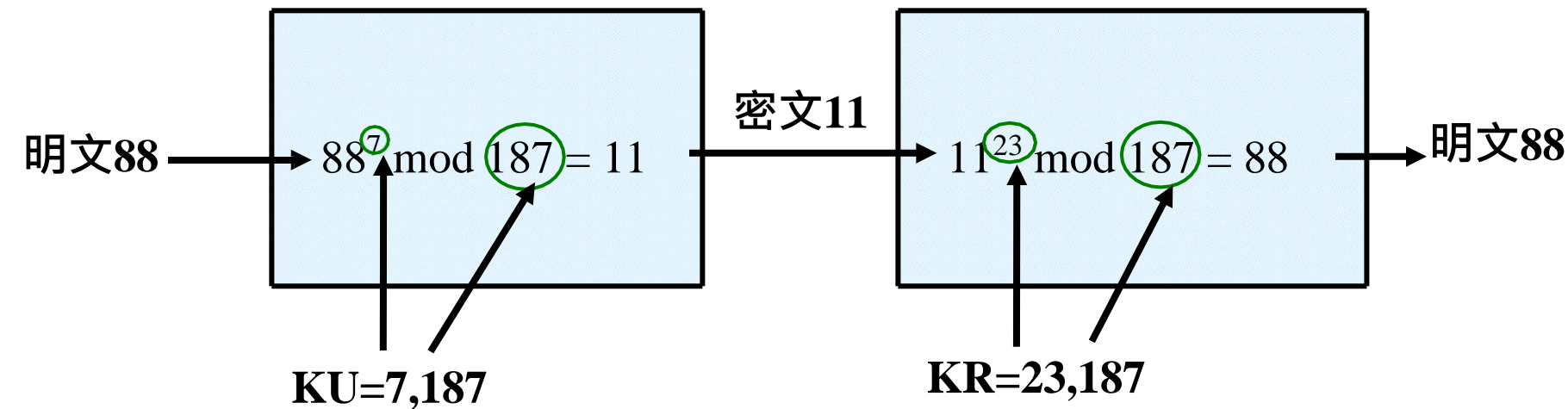
解密	
密文	C
明文	$M = C^d \pmod{N}$

RSA範例(1)

1. 選取質數： $p=17$ 與 $q=11$
2. 計算 $N = pq = 17 \times 11 = 187$
3. 計算 $\phi(N) = (p-1)(q-1) = 16 \times 10 = 160$
4. 選取 e 使得 $\gcd(e, 160) = 1$; 選取 $e=7$
5. 求取 d 使得 $de = 1 \pmod{160}$ 且 $d < 160$; 其值為 $d=23$ 因為 $23 \times 7 = 161 = 1 \times 160 + 1$
6. 發佈公開金鑰 $KU = \{7, 187\}$
7. 保存秘密私密金鑰 $KR = \{23, 187\}$

加密

解密



RSA 的加解密範例：

已知訊息明文 $M = 88$ ($88 < 187$)

加密： $C = 88^7 \pmod{187} = 11$

解密： $M = 11^{23} \pmod{187} = 88$

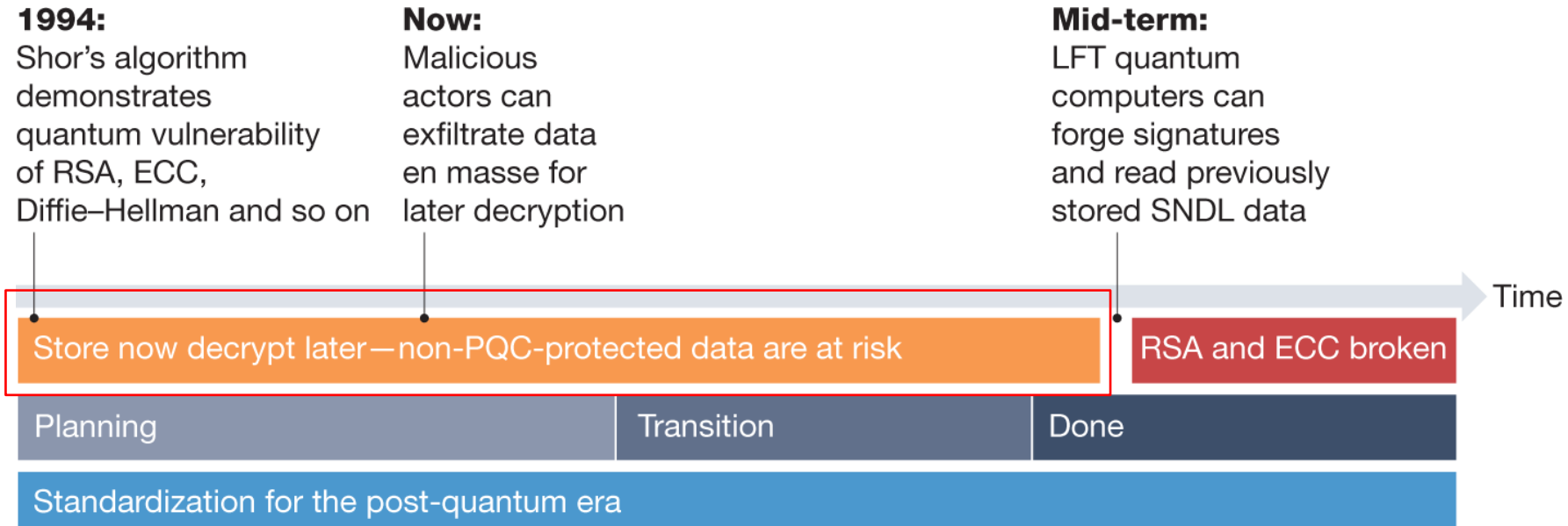
RSA範例(2)

- Alice 建立自己的金鑰對：
 - 她選擇 $p=397$ 及 $q=401$
 - 她計算 $n=397 \times 401=159197$
 - 然後計算 $\varphi(n)=396 \times 400=158400$
 - 接著選擇 $e=343$ 及 $d=12007$
- 如果Bob知道 e 及 n ，他如何傳送一份訊息給 Alice？
- 假設Bob想要送訊息「NO」給Alice，他將每一個字元轉成數字(從 00到25)，每一個字元被編碼成兩位數
- 於是，Bob連結這兩個兩位數的數字而得到一個四位數，明文為1314
- Bob 接著使用 e 及 n 來加密訊息，密文為 $1314^{343} = 33677 \bmod 159197$
- Alice 收到訊息33677後，使用解密金鑰 d 來解開密文 $33677^{12007} = 1314 \bmod 159197$
- 最後Alice獲得明文 1314，得到訊息：NO

公開金鑰的安全性

- 與私鑰加密法一樣，公開金鑰也可能受暴力攻擊法破解，但在實際理論上是不可行的
 - 一般金鑰長度都很長(≥ 1024 位元)，故不易被破解
 - 安全性取決於加解密破解密碼之間的難易差距要夠大
 - 目前已知破解RSA密碼法是非常困難
- 公開金鑰缺點：執行速度會較私鑰加密機制慢
- Peter Williston Shor於1994年證明量子電腦可以在多項式時間內進行因數分解
 - 假如量子電腦成熟，則Shor演算法將淘汰RSA和相關依賴於分解大整數困難性的衍生加密演算法

Store Now Decrypt Later



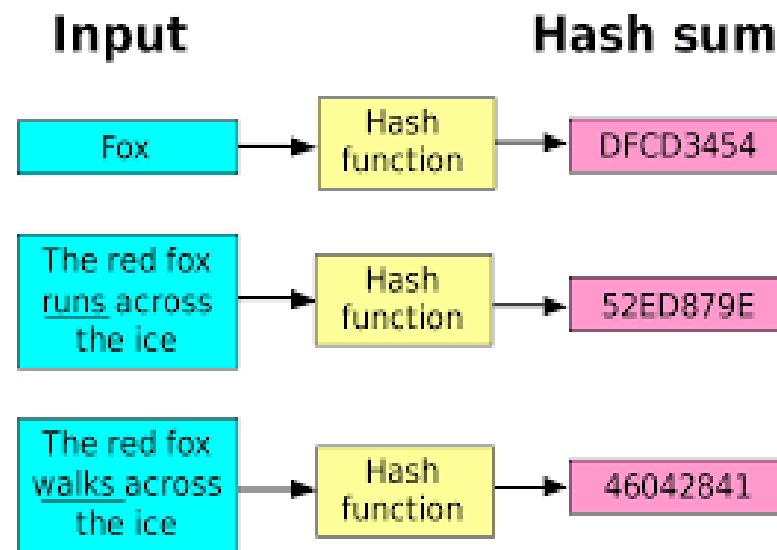
私鑰加密與公鑰加密

- 實務上常搭配使用：
 - 例如用DES將文章加密(快)，用RSA將DES的密鑰加密
- 如果RSA的N大於100位數，找到PQ可能要花數十年，代表難以破解
- 使用硬體設計DES比RSA 快1000 倍

	私鑰加密	公鑰加密
其他名稱	對稱式加密	非對稱式加密
加解密的Key是否相同	相同	不同
Key是否可公開	不可公開	公鑰可公開 私鑰不可公開
Key保管	若與N個人交換訊息，需要保管好N把Keys	無論與多少人交換訊息，只保管自己的私鑰
加解密速度	快	慢
應用	常用於加密長度較長的資料，如email	常用於加密長度較短的資料、及數位簽章

雜湊函式 Hash Function

- 又稱單向加密函數，能將任何長度的文件和資訊轉為特定長度的代碼，稱為雜湊值
 - Hash Value or Message Digest
- 雜湊函式用來產生檔案/訊息/資料的指紋 (fingerprint)
 - $h = H(M)$
 - 壓縮不定長度的訊息 M
 - 產生固定長度的指紋 h
- 常見：MD4, MD5, SHA-0, SHA-1

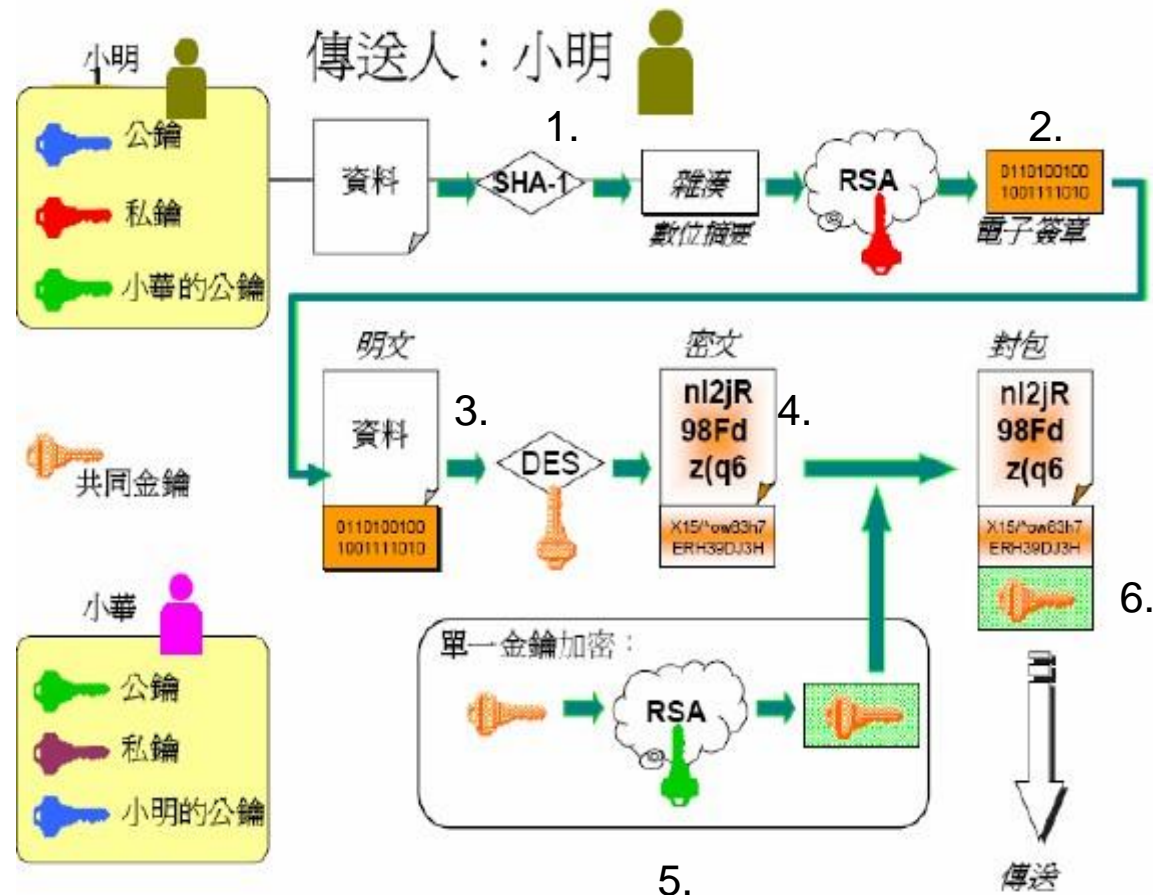


雜湊函式 Hash Function

- 雜湊函式是公開的，故須保護資訊的雜湊值
- 特性：
 - 不可反逆：無法由輸出反推其原輸入值
 - 抗碰撞性(collision resistance)：兩個訊息的雜湊值一樣機率非常低
 - 擴張性(Diffusion)：明文一個小改變會影響一個範圍的雜湊值
- 應用：雜湊常用來偵測訊息的變動以確保資料完整性、產生數位簽章

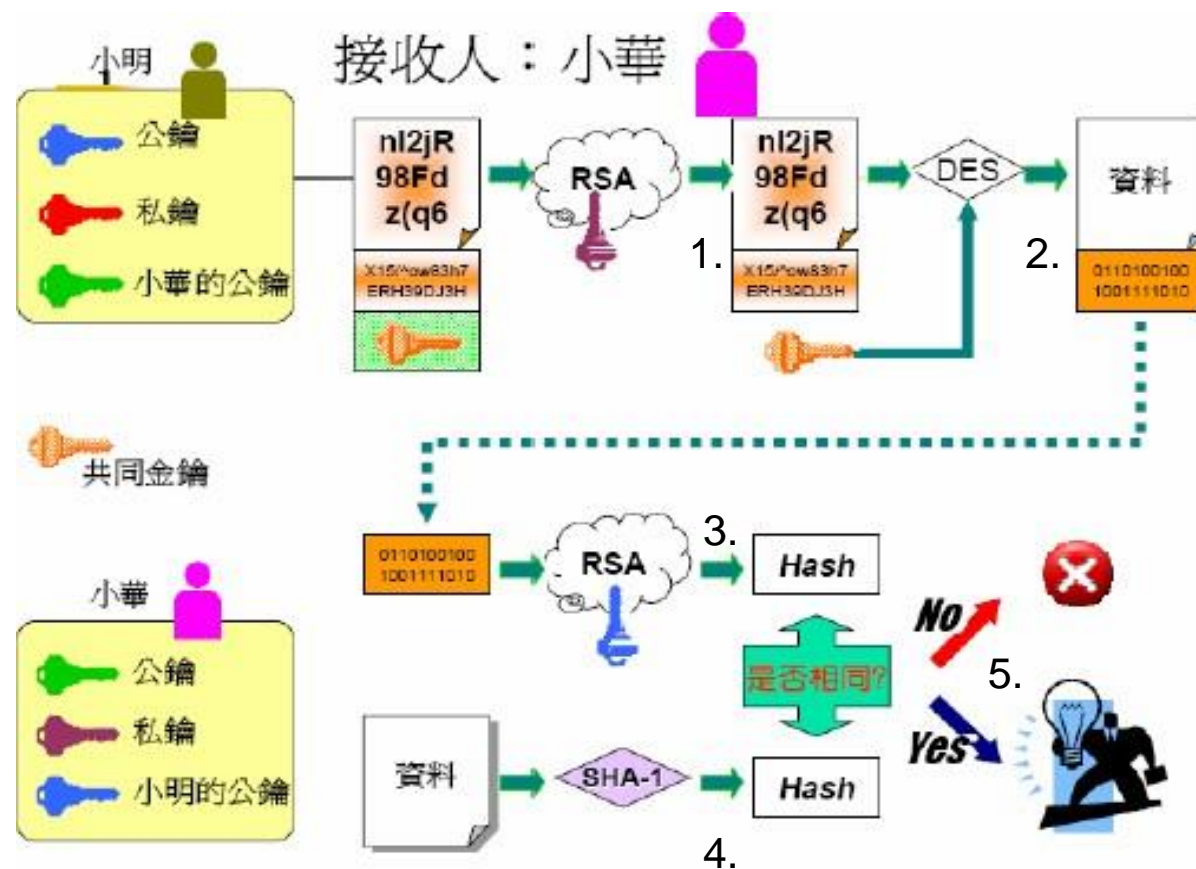
雜湊加密流程

1. 將資料經由Hash Function轉成雜湊值
2. 將雜湊值以傳送者小明的RSA私鑰做成數位簽章
3. 將數位簽章加於資料後端
4. 利用DES共同金鑰將新資料加密為密文
5. 利用接收者小華的公開金鑰，用RSA將DES共同金鑰加密
6. 將5. 的資料附在4.之後，傳送



雜湊解密流程

1. 利用接收者小華的私鑰，將加密後的DES共同金鑰解密
2. 用共同金鑰將密文轉為資料
3. 用傳送者小明的公開金鑰，將數位簽章轉為原始雜湊值
4. 將2.的資料送進雜湊函數
5. 比對3.與4.的雜湊值，若一樣代表資料無誤可信



數位簽章

- 數位簽章(Digital Signature，又稱電子簽章)並不是手寫簽名的影像檔，數位簽章是一種提供身份確認的加密形式，現階段已於多個國家通過，是邁向無紙化環境的重要關鍵
- 前美國總統柯林頓已在2000年簽署過讓數位簽章具法律效力的法案，Electronic Signatures in Global and National Commerce Act，簡稱 ESIGN
 - 聯邦和州層級都有電子簽章法，根據 2000 年的聯邦 ESIGN 法，電子簽章一般都與手寫簽名具有同等法律效力
- 行政院已於2024/02/29通過電子簽章法修正草案，並於2024/04/30經立法院三讀通過
 - 未來數位簽章等同本人親簽，若條件符合可與國外相互承認數位簽章

密碼破解

- 密碼破解(cryptanalysis)的兩種方式：
 - 密碼破解法(cryptanalysis)
 - 暴力攻擊法(brute-force attack)
- 幾類密碼破解流程
 - 只有密文(ciphertext only)
 - 密碼攻擊者透過監聽等手段只能取得密文，大量的密文也許會透露一些統計學上的蛛絲馬跡，但只靠密文要破解加密演算法並不容易
 - 已知原文(known plaintext)
 - 攻擊者找到一些原文與密文的對照，以破解加密演算法，例如大部份電子郵件內容前幾個字都是Dear。此方法幫助破解二戰時的德國加密機器Enigma
 - 選擇原文(chosen plaintext)
 - 攻擊者發出原文，隨即取得密文。例如發一封電子郵件給對方，內容誘使收信人將信件加密後轉發，再攔截到整篇轉發的密文
 - 選擇密文(chosen ciphertext)
 - 與選擇原文相反的方式，收件人收到加密電子郵件，在回覆時將原信件作為附件卻未加密，因此有機會從密文取得原文

密碼破解法(cryptanalysis)

- 當演算法受到攻擊時，分析者會找尋將原文轉成密文的演算法缺點，且在沒有金鑰的情況下還原資訊的原文
- 若演算法具有這類弱點，也就不能稱為牢靠的演算法，也就不能被信賴使用
- 2004~2005年山東大學數學與系統科學學院教授、密碼技術與資訊安全實驗室主任-王小雲所領導的研究團隊成功地破解了國際上廣泛應用的兩大密碼演算法MD5和SHA-1
 - 利用鴿籠原理，無長度限制的訊息透過雜湊函式計算成為有長度限制的hash值，必有碰撞產生
 - 王小雲及她的團隊在2005年將該成果發表為4篇論文，被公認為近年來國際密碼學最出色的成果之一
 - 提出SHA-1的雜湊碰撞演算法，只需少於 2^{69} 的計算複雜度就能找到一組碰撞，改進了Rijmen和Oswald所提的 2^{80} 計算複雜度的碰撞算法。當年8月進一步改良至 2^{63} 計算複雜度

暴力攻擊法(brute-force attack)

- 最基本的破解法，前提是可以看懂或辨識明文
- 理論上可以嘗試所有可能的金鑰，直到找到真正的金鑰
- 一般來說所嘗試的金鑰數大約是可能金鑰總數的一半
- 其破解所需成本與金鑰大小成正比
- 金鑰長度愈長愈不易經由暴力攻擊法破解

金鑰長度 (位元)	可能的金鑰總數	耗時 (每微秒 μs 加密一次)	耗時 (每微秒 μs 加密 10^6 次)
32	$2^{32}=4.3\times 10^9$	2^{31} 微秒=35.8分鐘	2.15毫秒
56	$2^{56}=7.2\times 10^{16}$	2^{55} 微秒=1142年	10.01小時
128	$2^{128}=3.4\times 10^{38}$	2^{127} 微秒= 5.4×10^{24} 年	5.4×10^{18} 年
168	$2^{168}=3.7\times 10^{50}$	2^{167} 微秒= 5.9×10^{36} 年	5.9×10^{30} 年
26個字母 (排列數)	$26!=4\times 10^{26}$	2×10^{26} 微秒= 6.4×10^{12} 年	6.4×10^6 年

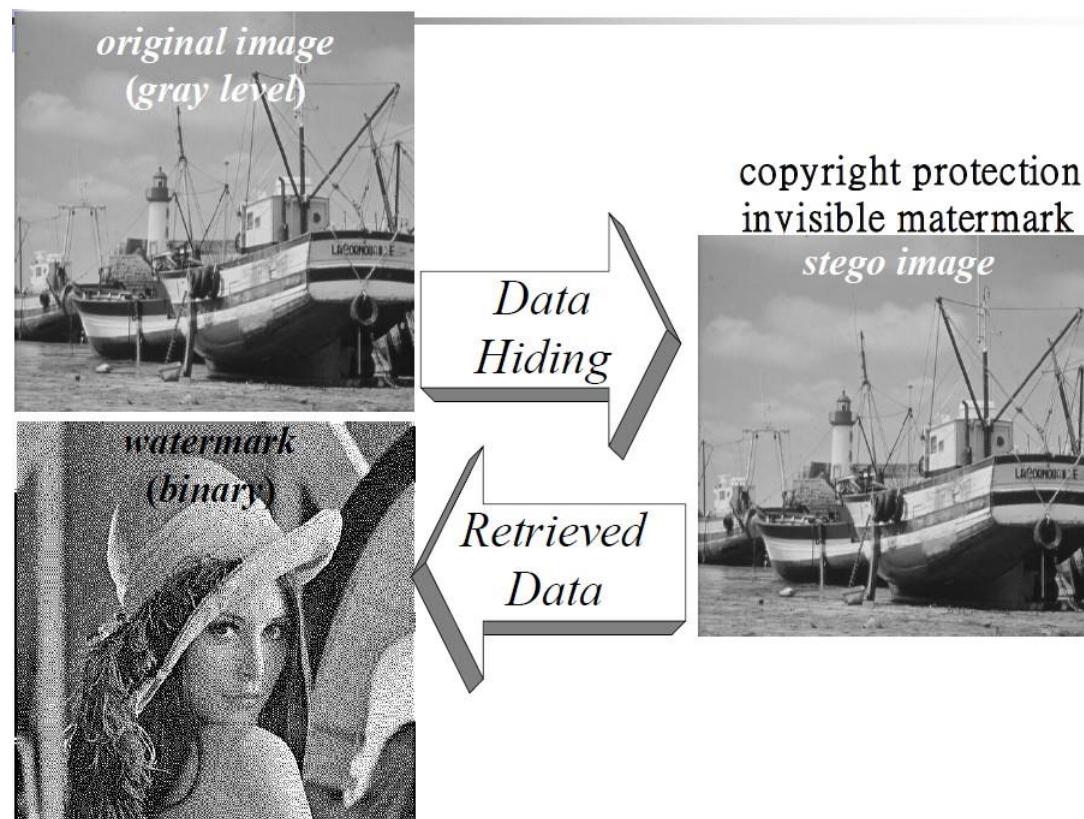
其他破解方式

- 其他方式透過系統週遭的弱點
 - 單純討論加密的內容時，一般都不太會探討這種破解手法。但是這是hacker攻擊常用的方法
 - 利用資訊系統的弱點(vulnerability)是網路上一種實際攻擊的方式
 - 攻擊系統週遭的弱點缺陷，會比攻擊加密演算法來得容易

密碼學日常應用

- 密碼學與資料隱藏






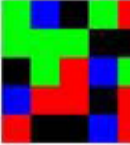
- 網路安全
- 展頻通訊
- 浮水印
 - 影像浮水印
 - 聲音浮水印
- 條碼
 - 一維條碼
 - 二維條碼



二維條碼

- 一維條碼用寬度記錄資訊
- 二維條碼的長度與寬度都記錄資訊
 - Ex: QR Code (Quick Response)
 - 三個角用來定位，不易受旋轉影響
 - 依照圖的大小最多能存數千個字



Name	Figure	Manual
QR Code		Developed by TOYOTA subsidiary Denso Wave in 1994.
VS Code		Develop by Veritec in America.
SemaCode (Data Matrix)		Developed by a software company, Semacode based in Waterloo, Ontario, Canada.
Visual Code		Developed to enable HCI using camera phones.
Shot Code		Created by High Energy Magic of Cambridge University when developing TRIPCode.
Color Code		Developed by Colorzip and mainly used in Korea

二維條碼-QR Code

- 起源於日本，為了追蹤汽車零件
- 2000年國際QR碼標準認可
- 目前應用非常廣泛
 - 文字、網頁、商品資訊、店家資訊
 - 電子票券、車票, ex: ibon買的高鐵票
 - 公車站牌上QR碼，掃一下可知班車時刻與路線並存於手機
 - 課堂問卷與互動



zh.wikipedia.org



課後問券

- 密碼學與密碼分析 課後問券
- <https://forms.gle/RCXZQYokWXnXm7KR9>



Reference Materials

- Thanks to 「教育部資訊安全人才培育計畫」 & 「國網中心雲端資安攻防平臺（Cyber Defense eXercise，CDX）」



Q & A