Module 2: 密碼學基礎

學習目的

1. 透過密碼學的基礎知識介紹，培養同學對傳統密碼學基本原理、公開金鑰系統演算法與RSA演算法推演及驗證、數位簽章原理、雜湊函式基本原理、亂數產生等議題的完整概念。

2. 本模組利用五個小節包括(1)加密的基本概念 (2)對稱式加解密法:介紹DES、Triple DES、IDEA及AES等演算法 (3)非對稱式加解密法:介紹公開金鑰原理及RSA演算法(4)數位簽章:介紹數位簽章型式及數位簽章標準(DSS及DSA) (5)雜湊函式:介紹雜湊函式需求及知名的雜湊函式演算法(MD5及SHA-1)。

學習目的(續)

3. 各演算法內容包括安全性需求、演算法原理介紹、系統架構、操作程序及安全性分析，透過動畫及原理解說，提供專門研習密碼學或資訊安全同學對密碼學有初步的認識。

4. 使用六個鐘點，建議2-1~2-2使用三個鐘點， 2-3~2-5三個鐘點(其中2-4~2-5本章謹作概念介紹，詳細內容將於第五章介紹)，2-6專題實現作可作為學生的homework。

Module 2: 大綱

Module 2-1: 認識加密的基本概念(*)

Module 2-2: 對稱式加解密法(**)

Module 2-3: 非對稱式加解密法(**)

Module 2-4: 認識數位簽章(**)

Module 2-5: 雜湊函式 (*)

Module 2-6: 專案實作(*)

ul

* 初級(basic):基礎性教材內容

**中級(moderate):教師依據學生的吸收情況，選擇性介紹本節的內容

***高級(advanced):適用於深入研究的內容

u1

* 初級(basic):基礎性教材

**中級(moderate):教師依據學生的吸收情況，選擇性介紹本節的內容

***高級(moderate):適用於深入研究的內容

user, 2007/2/14

Module 2-1: 認識加密的基本概念

2-1 認識加密的基本概念

- 加密就是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，而且允許真正獲得授權的人，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。

2-1 認識加密的基本概念

- 透過加密可以提供下列三種安全服務：
  - **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。
  - **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。
  - **可說明性(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。
- 在安全計畫之中廣泛地使用加密機制，只是因為加密機制可以協助資訊的機密性、完整性和可說明性。

2-1 認識加密的基本概念

- 本節的內容如下：

  Module 2-1-1: 加密專有名詞

  Module 2-1-2: 對稱的加密模式

2-1-1 加密專有名詞

- 加密運作模式專有名詞如下：
  - **原文（plaintext）**：資訊的原始格式。
  - **密文（chipertext）**：透過加密演算法輸出的資訊。
  - **演算法（algorithm）**：將原文處理成密文的運算法則。
  - **金鑰（key）**：一個與原文無關的數值，其可將原文轉成密文或將密文轉成原文的過程中，一種用來協助演算法計算的資料。
  - **加密（encryption）**：將原文經加密運算轉成密文的程序。
  - **解密（decryption）**：為加密的逆運算，將密文轉成原文的程序。

2-1-1 加密專有名詞

- **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這兩個領域。

2-1-2 對稱的加密模式

必要條件

- 使用對稱式加密的兩個必要條件:
  - 一個強固的加密演算法
  - 只有傳送者與接收者才知道的一把秘密金鑰

$$Y = \mathrm{E}_K(X)$$

$$X = \mathrm{D}_K(Y)$$

- 假設加密演算法是已知的，意味著需要一個分送金鑰的安全管道

2-1-2 對稱的加密模式

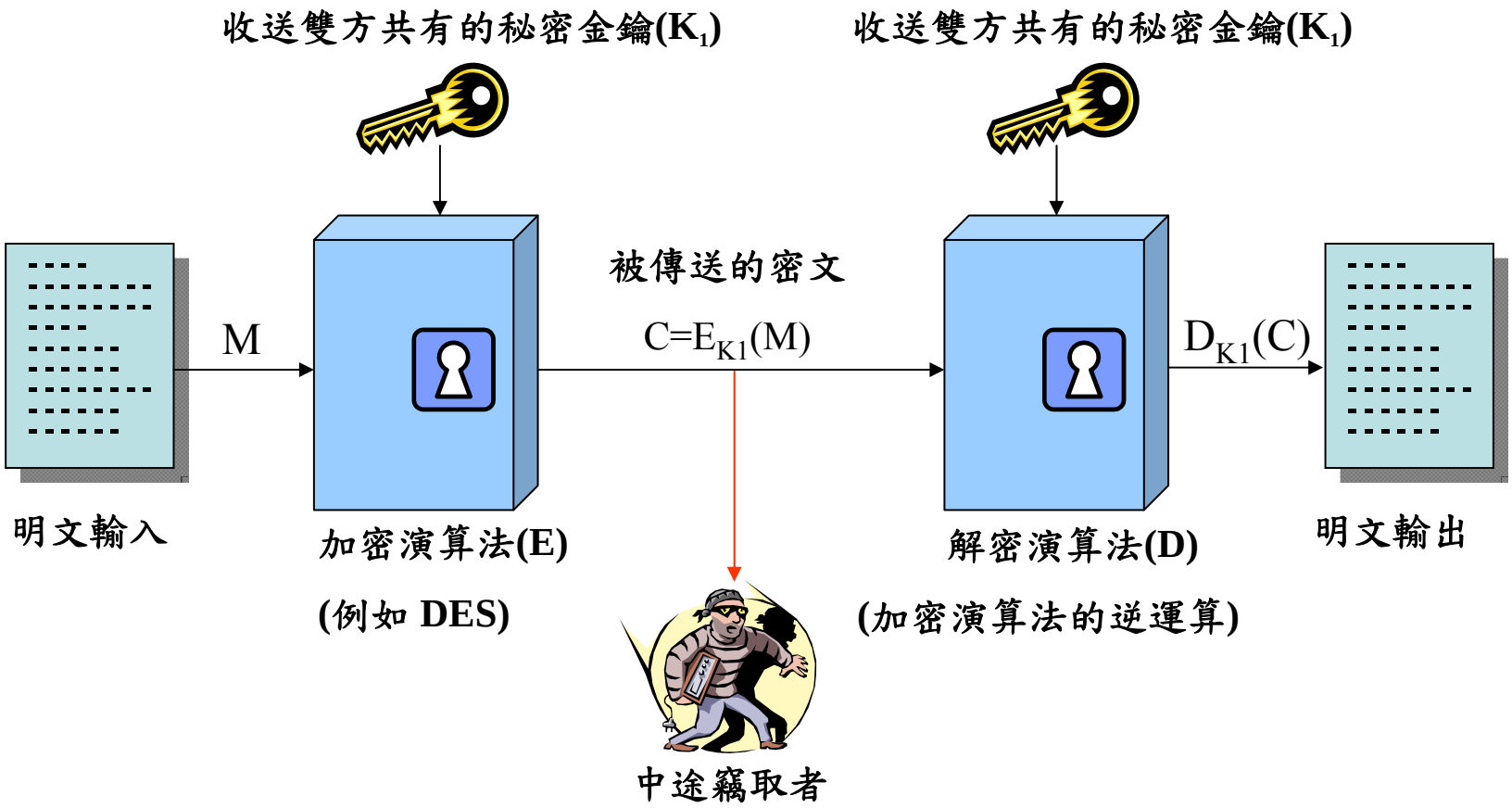


圖2.1 傳統加密的簡化模型

2-1-2 對稱的加密模式

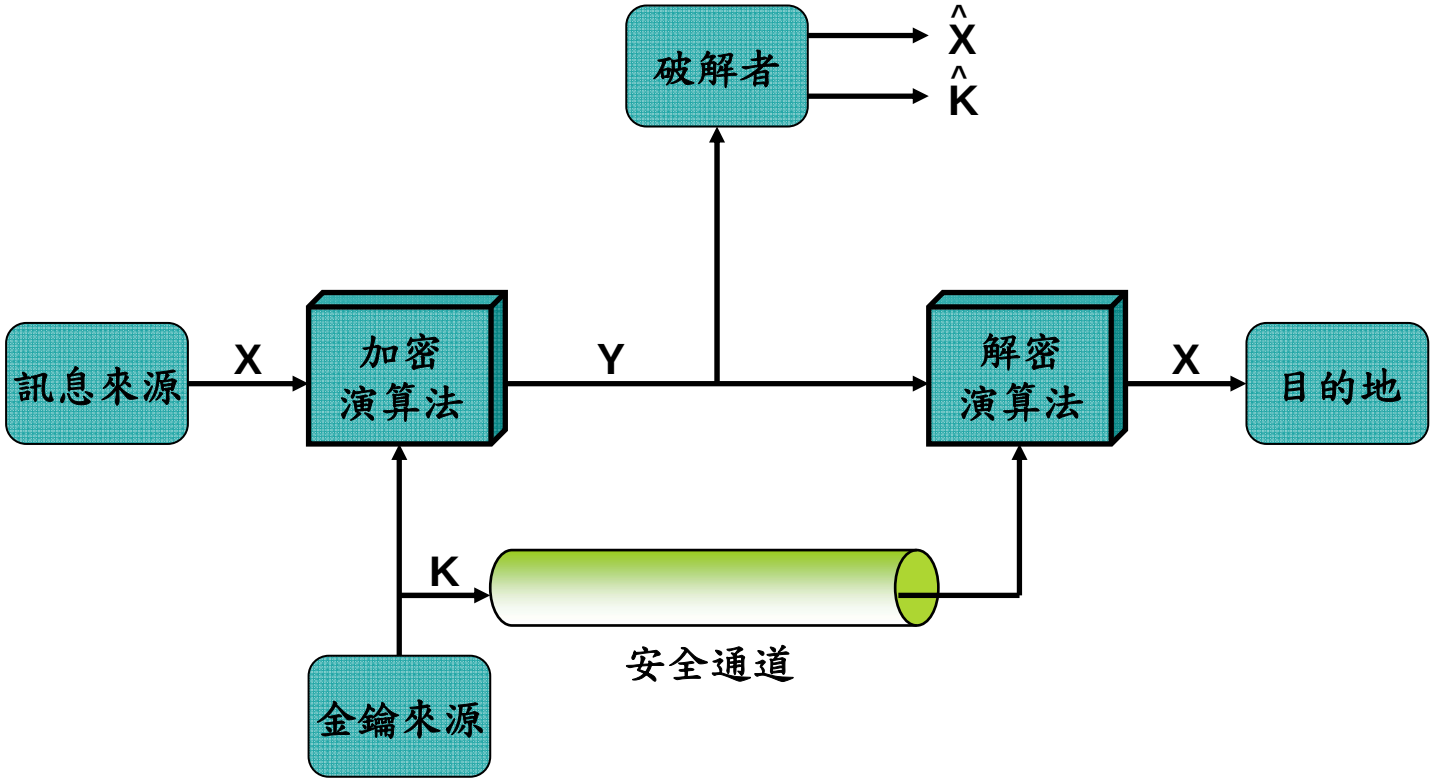


圖2.2 傳統密碼學的模型

2-1-2 對稱的加密模式

密碼學（cryptograph）

- **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
    - 前兩者混用(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
    - 區段加密(block cipher)：一次處理一大段元素，根據輸入的區段產生輸出的區段
    - 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

2-1-2 對稱的加密模式

密碼破解（**cryptanalysis**）的兩種方式：

- 密碼破解法(cryptanalysis)
- 暴力攻擊法(brute-force attack)

密碼破解的方式

- **只知道密文(ciphertext only)**
  - 只知道演算法與密文，利用統計方法來求出明文
- **已知明文(known plaintext)**
  - 藉由已知或推測的明文-密文組合來破解
- **自選明文(chosen plaintext)**
  - 藉由自選的明文與對應的密文來破解
- **自選密文(chosen ciphertext)**
  - 藉由自選的密文與對應明文的來破解
- **自選文字(chosen text)**
  - 藉由加密自選的明文或解開自選的密文來破解

2-1-2 對稱的加密模式

1. 密碼破解法（**cryptanalysis**）
   - 當演算法受到攻擊時，分析師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。
   - 若是具有這類弱點的演算法，也就不能稱為牢靠的演算法，當然也就不能使用。
   - 例如2005~2006年山東大學數學與系統科學學院教授、資訊安全實驗室主任—王小雲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。
   - 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。

2-1-2 對稱的加密模式

2. 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 (每微秒加密一次)	耗時 (每微秒加密10^6次)
32	$2^{32}=4.3\times10^9$	2^{31}微秒=35.8分鐘	2.15毫秒
56	$2^{56}=7.2\times10^{16}$	2^{55}微秒=1142年	10.01小時
128	$2^{128}=3.4\times10^{38}$	2^{127}微秒=5.4×10^{24}年	5.4×10^{18}年
168	$2^{168}=3.7\times10^{50}$	2^{167}微秒=5.9×10^{36}年	5.9×10^{30}年
26個字母 (排列數)	$26!=4\times10^{26}$	2×10^{26}微秒=6.4×10^{12}年	6.4×10^6年

表2-1 全面搜尋金鑰所花費的平均時間

2-1-2 對稱的加密模式

- 其他方式透過系統週遭的弱點
  - 利用資訊系統的弱點(vulnerability)是駭客在網路上一種實際攻擊的方式。
  - 攻擊系統周遭的弱陷，會比攻擊加密演算法來得容易。
  - 在討論加密的內容時，一般都不太會探討這種問題，但是為網路駭客(hacker)攻擊常用的方法。

2-1-2 對稱的加密模式

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，因為密文本身提供的資訊不足以唯一決定其密文。
- **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。

Module 2-2: 對稱式加解密法

2-2 對稱式加解密法

前言

- 加密法可以分為『私密金鑰(pubic key)』與『公眾金鑰(public key)』兩大類。
- 所有的傳統加密演算法都是私密金鑰加密法，在1970年代的公開金鑰加密法發明之前，這是唯一的加密方式。
- 使用私密金鑰加密時，只要經過授權並擁有相同金鑰的人，都可以讀取資訊的內容。
- 資訊的保護可簡化為金鑰管理(key management)問題。

2-2 對稱式加解密法

- 本節的內容如下：
  - 2-2-1 私密金鑰加密
  - 2-2-2 取代技巧(**)
  - 2-2-3 替換技巧(**)
  - 2-2-4 資訊隱藏
  - 2-2-5 區段加密的原理
  - 2-2-6 資料加密標準(DES)
  - 2-2-7 Triple DES
  - 2-2-8 AES(**)
  - 2-2-9 其它私密金鑰演算法

u2

u2

user 2007/1/24
* 選擇性(optional)介紹的章節:教師依據學生的吸收情況，選擇性介紹本節的內容

** 先進(advanced)的章節:適用於深入研究的內容
user, 2007/1/24

2-2-1 私密金鑰加密

- 由於加密／解密使用相同的金鑰，因此私密金鑰加密也稱為對稱式金鑰加密。
- 在經過對稱式加密之後，可以確保資訊的機密性。只有金鑰的擁有人(key owner)，才可以解密訊息。
- 在傳輸的過程中，訊息發生任何變化，都會造成解密失敗，因此可以得知訊息是否遭到修改。

2-2-1 私密金鑰加密

- 私密金鑰加密的缺點為無法確認建立金鑰、加密和傳送有效訊息的人。(相關的人可能有此私密金鑰)
- 建立私密金鑰加密的優點為速度相當快，而且很容易利用軟體或硬體複建置私密金鑰。

Module 2-2-2: 取代技巧

2-2-2 取代技巧

- 取代加密(substitution cipher) 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。
  - 前人已發明數種方法敘述如下:
    - Caesar 加密法
    - Monoalphabetic 加密法
    - Playfair 加密法
    - Polyalphabetic 加密法等

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。
- 方法為將每個字母用其後的第三個字母來取代。
- 範例:

  meet me after the toga party

  PHHW PH DIWHU WKH WRJD SDUWB

- 其缺點是，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:

```
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
0 1 2 3 4 5 6 7 8 9 10 11 12
n  o  p  q  r  s  t  u  v  w  x  y  z
13 14 15 16 17 18 19 20 21 22 23 24 25
```

  - 則可將 Caesar 加密法定義如下:

$C = \mathrm{E}(p) = (p + k) \bmod (26)$

$p = \mathrm{D}(\mathrm{C}) = (\mathrm{C} - k) \bmod (26)$

Caesar 加密法的破解方式

- 只有 26 種可能的加密方式
  - A 可能對應到 A,B,..Z 其中一個字母
- 可以採用暴力攻擊法逐一地測試
- 給定密文之後，只需測試所有可能的移位距離
- 當明文出現時，必須可以辨識出來
- 例如，請破解密文“GCUA VQ DTGCM”
- 例Caesar 加密法

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25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

Monoalphabetic 加密法

- 不只是單純對固定字元取代，允許可以任意地取代字元
- 每個明文字元都會隨機地對應到另一個密文字元
- 因此，金鑰的長度就是 26 個字元

明文：abcdefghijklmnopqrstuvwxyz
密文：DKVQFIBJWPESCXHTMYAUOLRGZN

明文：ifwewishtoreplaceletters
密文：WIRFRWAJUHYFTSDVFSFUUFYA

Monoalphabetic 加密法的安全性

- 前一頁的例子一行共有26字母，共有 26! ＝4 x 10^{26} 把金鑰。
- 有那麼多把金鑰，比DES演算法更安全了，但是這樣想是不完全正確的。
- 問題出在語言本身的特性。

Monoalphabetic 加密法的安全性

語言的贅詞與密碼破解的關係

- 人類的語言是有規則性
- 例如 “th lrd s m shphrd shll nt wnt”
- 字元的使用頻率並不均等(詳如圖 2.3)
  - 由統計得知，在英文中，**e** 的字母使用次數最高
  - 其次是 T,R,N,I,O,A,S
  - 其他字元其實很少用到，例如 Z,J,K,Q,X
- 單字元、雙字元以及三字元的出現頻率，都已經有現成的統計表格可以運用。

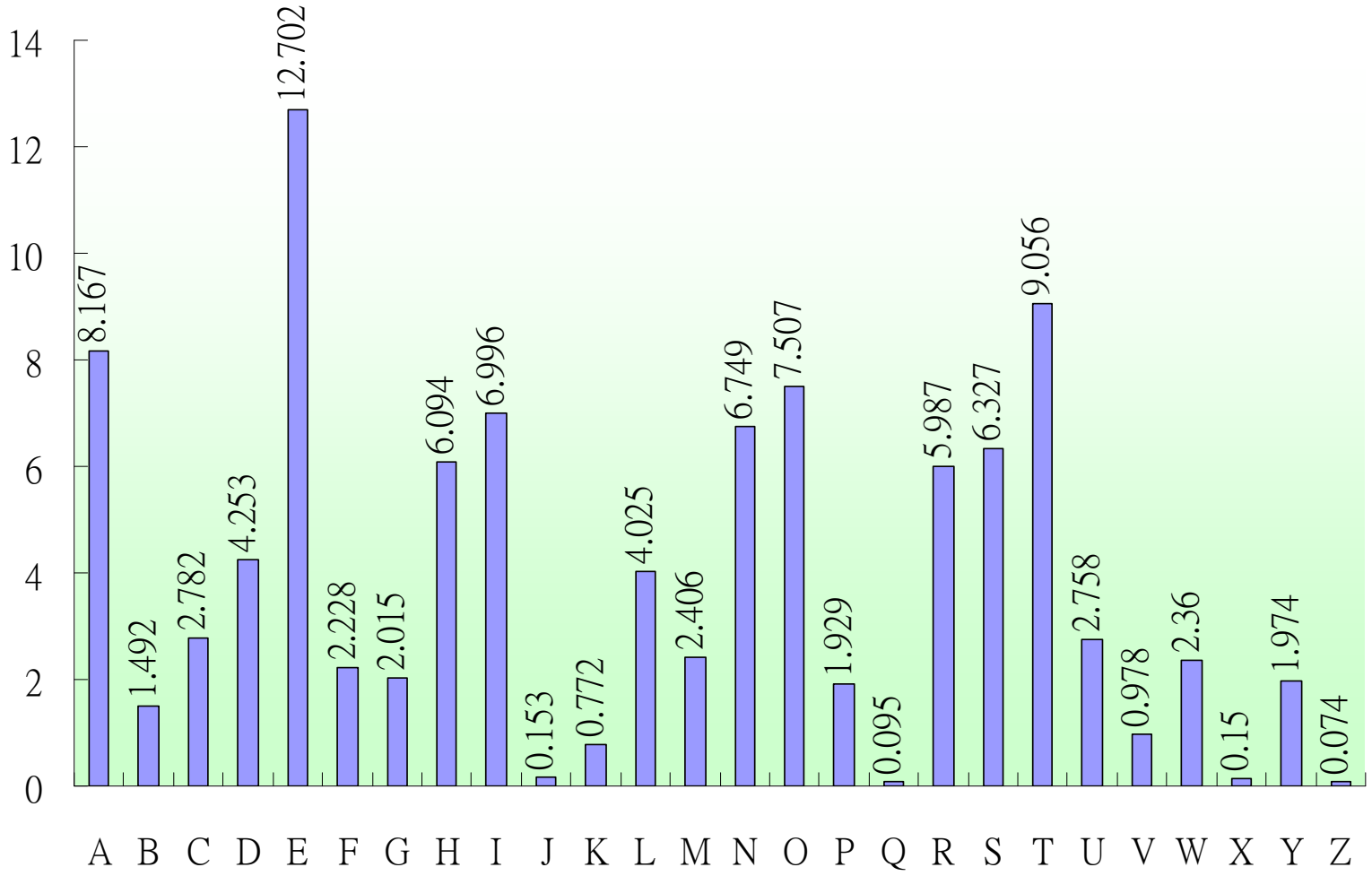


圖 2.3 英文字元的出現頻率統計

Monoalphabetic 加密法的安全性

- 破解的基本概念 - monoalphabetic 取代加密法並不會改變語言字元間的相對出現頻率。
- 阿拉伯科學家在九世紀時就已經發現這個破解法。
- 計算密文的字元出現頻率，將結果與已知的頻率數值互相比對。
- 如果針對 Caesar 加密法的統計圖形來找尋波峰與波谷
  - 波峰出現在 A-E-I 三字元, NO 雙字元, RST 三字元
  - 波谷出現在: JK, X-Z
- 如果是針對 monoalphabetic 加密法，就必須逐一辨識每個字元。
  - 常用的雙字元與三字元統計表格可以幫助我們來破解

Monoalphabetic 加密法的安全性

破解範例

- 給定密文:

  UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
  VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
  EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

- 計算字元相對頻率

P	13.33	H	5.83	F	3.33	B	1.67	C	0.00
Z	11.67	D	5.00	W	3.33	G	1.67	K	0.00
S	8.33	E	5.00	Q	2.50	Y	1.67	L	0.00
U	8.33	V	4.17	T	2.50	I	0.83	N	0.00
O	7.50	X	4.17	A	1.67	J	0.83	R	0.00
M	6.67								

- 經由上表與圖2-3比對，猜測 P 與 Z 就是 e 與 t
- 猜測 ZW 就是 th，而 ZWP 就是 the

Monoalphabetic 加密法的安全性

- 邊試邊改的情況下，最後可得明文如下:

  it was disclosed yesterday that several informal
  but direct contacts have been made with political
  representatives of the viet cong in moscow

Playfair 加密法(**)

- 在monoalphabetic 加密法中那麼多的金鑰，也無法確保安全性。
- 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。
- 將關鍵字的字元依序填入矩陣中 (刪除重複字元)，再將26字母剩餘字，填入矩陣中(I跟J視為同一個字元)。
- 例如，使用關鍵字 MONARCHY

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

加密與解密(**)

每次加密明文中的兩個字元:

1. 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"
2. 如果這兩個字元落在矩陣中的同一列，則用它們右邊的字元來取代它們 (把第一個字元當成是最後一個字元的右邊字元)。例如 "ar" 變成 "RM"
3. 如果這兩個字元落在矩陣中的同一行，則用它們下方的字元來取代它們 (把第一個字元當成是最後一個字元的下方字元)。例如 "mu" 變成 "CM"
4. 在其他的情況下，每個字元都換成與它自己同一列，但是與另一個字元同一行的那個字元。例如，"hs" 變成 "BP"，"ea" 變成 "IM" 或是 "JM" (由加密者自行決定)。

Playfair 加密法的安全性(**)

- 比 monoalphabetic 安全許多，因為有 26 x 26 = 676 那麼多雙字元。
- 表格中需要 676 個項目才足以分析 (monoalphabetic 的表格只需要 26 個項目)，相對地產生更多的密文。
- 已經廣泛地使用許多年 (例如，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。

Polyalphabetic 加密法(**)

- 另一個提升安全性的方法，就是採用多重取代法則稱為 “polyalphabetic加密法”。
- 在具備”多重取代法則”與較”一致的頻率分佈”的情況下，讓破解者更難得逞。
- 加密運作原理
  - 使用一組相關的單一字母(monoalphabetic)取代法則來加密
  - 藉由一把金鑰來決定明文轉密文，該選用那個取代法則
  - 依序使用不同的取代法則，當遇到金鑰結尾時，再重頭開始

One-Time Pad

- 美國陸軍資訊部門Joseph改良Vernam加密法，此方法可達到絕對地安全。
- 如果隨機金鑰的長度真的跟訊息一樣，那麼這個加密法就是安全的，此法稱為 One-Time Pad (OTP)。
- 無法破解的原因是密文與明文間沒有任何統計上的關係。
- 因為對**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。

One-Time Pad

- 理論上來講，One-Time Pad（OTP）是唯一無法破解的加密系統。
- OTP使用一連亂數排列的數字，對一段訊息進行編碼（詳見圖2-4）。
- 如果OTP真的全部都使用亂數、OTP只能使用一次，OTP的長度比訊息長，那麼就沒有找不到密文中的原文的金鑰（OTP本身），因此也就無法解出訊息內容。

One-Time Pad

- OTP還有一點必須特別注意的事項 — 只能使用一次，如果重複使用，就可能進行分析和破解。
- 現今某些加密系統號稱可以模擬OTP。或許這類系統可以提供充分的安全性，但可能也是很容易被破解的系統。
- 一般來說，OTP並不適用於高流量的網路環境。

One-Time Pad

訊息：	S	E	N	D	H	E	L	P
符合字元變化的數值：	19	5	14	4	8	5	12	16
One-time pad：	7	9	5	2	12	1	0	6
加入原文的運算結果：	26	14	19	6	20	6	12	22
密文：	Z	N	S	F	T	F	L	V

圖2-4 One-Time pad運算方式

資料來源:摘自W. Stalling:
Cryptography and Network Security

Module 2-2-3: 置換技巧 (Transposition Technique)

置換加密法

- 置換加密法(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，所以仍可破解。

柵欄加密法(Rail Fence)(**)

- 將明文寫成一連串的深度為2欄對角形式柵欄，然後一列一列地讀出。
- 例如，將訊息寫成

  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

  MEMATRHTGPRYETEFETEOAAT

列置換加密法(**)

- 一個較為複雜的機制，在行數固定的情況下，將訊息一列一列地寫成矩形的結構。
- 然後在一列一列讀出之前，根據某把金鑰來重排行的順序。

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

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並不安全。
- 因此，我們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
- 這就是古典與當代加密法的橋樑。

旋轉機(**)

- 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如圖 2.5 。
- 旋轉機在第二次世界大戰時被廣泛地使用
  - German Enigma, Allied Hagelin, 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 $26^3=17576$ 個取代法。

2-2-3 置換式密碼

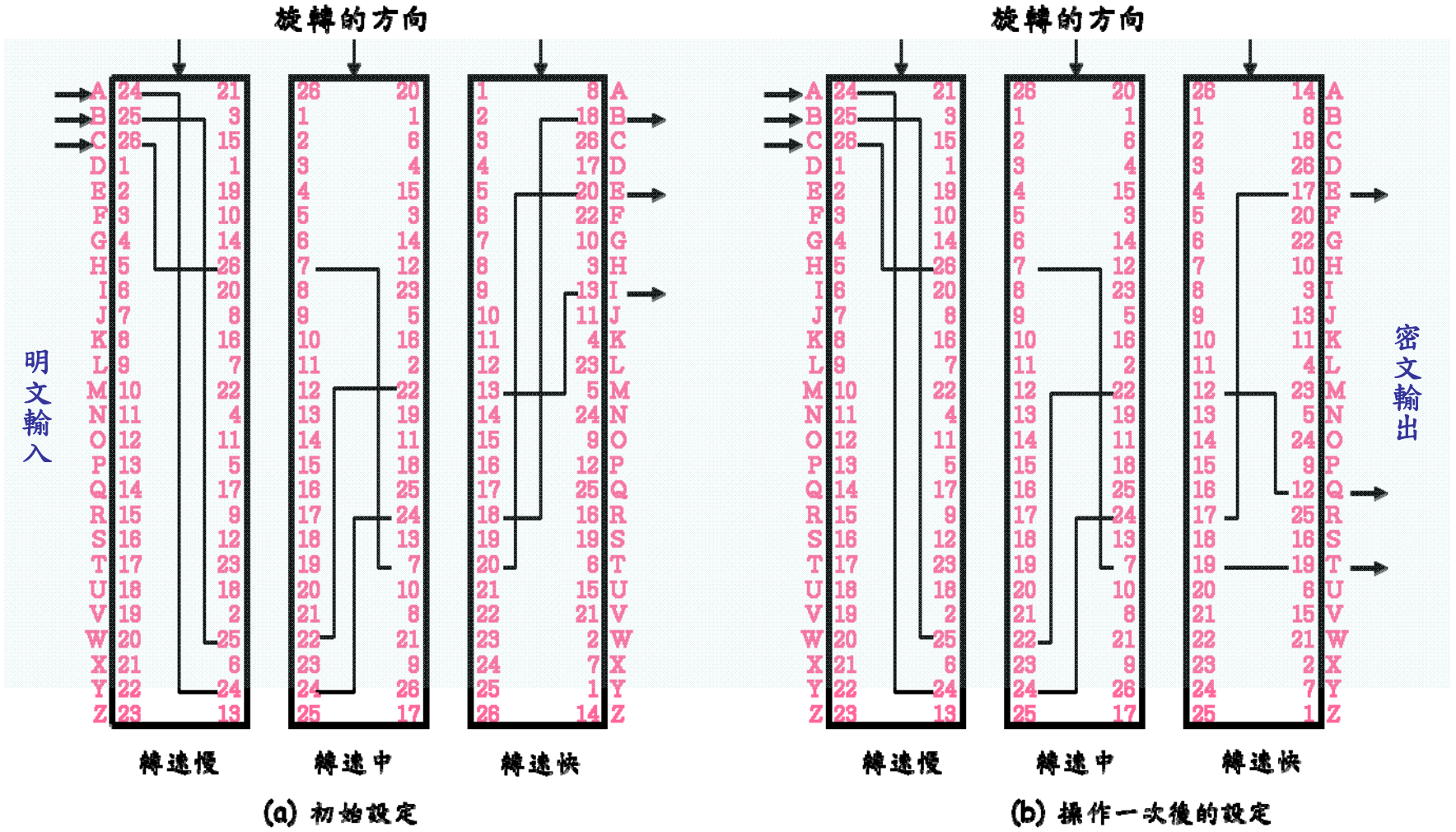


圖 2.5 三個軸向旋轉機:連結數字表示線路的接續方式

Module 2-2-4: 資訊隱藏(Steganography)

資訊隱藏

- 資訊隱藏(Steganography)是將「訊息存在」這件事隱藏起來，常用方式下:
  - 利用某種方式將一個訊息中的部分字元標示起來
  - 利用不可視的墨水
  - 利用影像或音效檔中的最低位元來隱藏訊息
- 缺點
  - 用極高的成本，來隱藏極少量的資訊位元

Module 2-2-5: 區段加密的原理

2-2-5 區段加密的原理

前言

- 檢視當代的區段加密法(block cipher)
- 使用最廣泛的密碼學演算法類型之一

2-2-5 區段加密的原理

區段加密與資料流加密

- 區段加密法(block cipher):
  - 將訊息分為區段來處理，每次對一整個區段加解密，每一次對多個的字元(64 位元或更多)進行取代
- 串流加密(stream cipher):
  - 在加解密時，一次處理一個位元(bit)或位元組(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。
- 現今許多著名的加密法都是區段加密法，所以本教材著重在區段加密。

2-2-5 區段加密的原理

- 許多對稱式區段加密法都是以「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段產生唯一的密文區段，讓加解密程序為可逆(reversible)。
  - 區段加密可視為極大量的取代加密法
- 對 64 位元的區段加密來說，需要 2^{64} 個項目的表格以用來對應加解密的想法，其原理來自於前面介紹的取代及置換的混合式(mixed)加密法。

2-2-5 區段加密的原理

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，成為當代區段式加密法的基礎。
- S-P 網路是以下列兩種基本密碼學運算為基礎
  - 取代 (S-box)
  - 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

2-2-5 區段加密的原理

Feistel 加密結構

- Horst Feistel提出了feistel加密法如圖2-6
  - 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

2-2-5 區段加密的原理

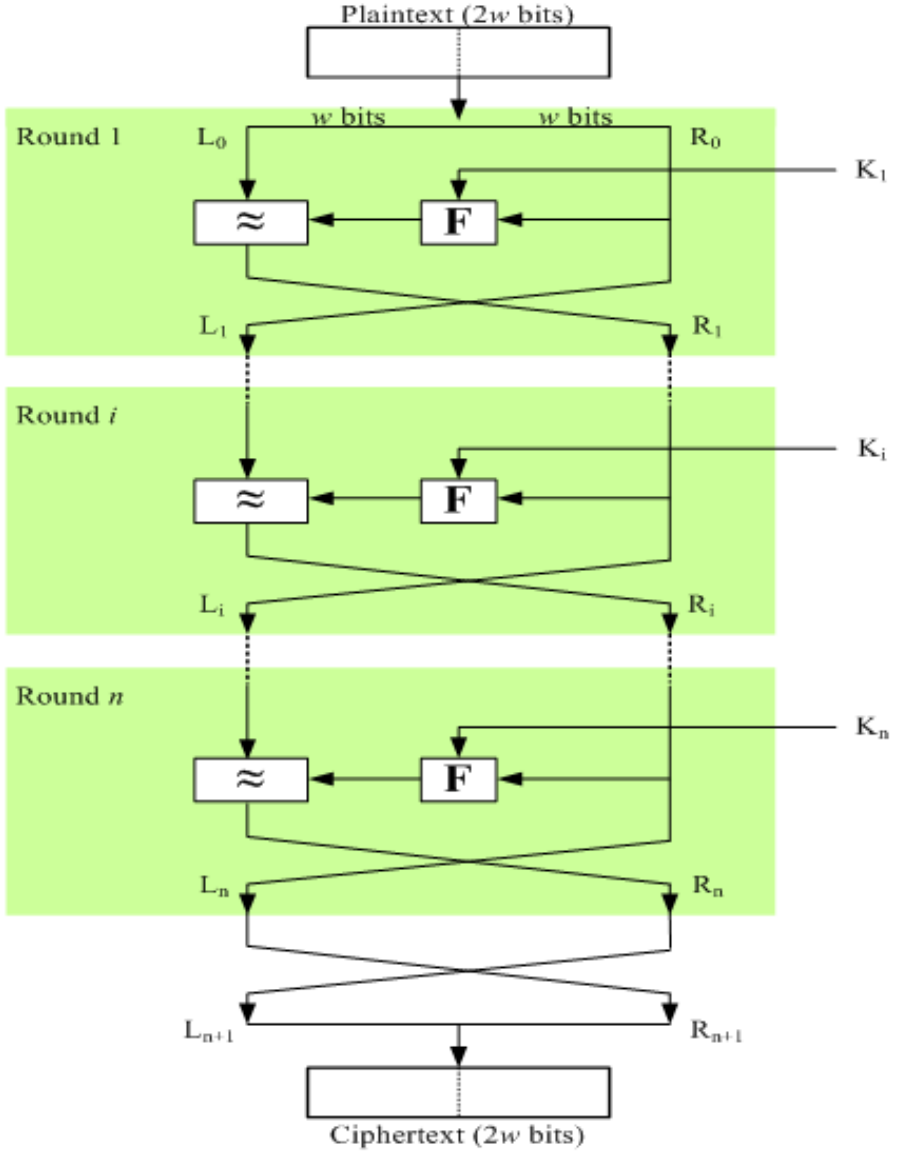


圖2-6標準Feistel網路

2-2-5 區段加密的原理

Feistel 加密法的設計原理如圖2-7

- **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

2-2-5 區段加密的原理

- **用軟體快速加解密(fast S/W encryption/decryption)**
  - 近年來常使用軟體來執行演算法，所以演算法的運算速度就需考量
- **容易分析(ease of analysis)**
  - 若演算法可清楚的表達，分析其被破解的危險性也會變得比較簡單，有助於設計出更安全強固的方法。

2-2-5 區段加密的原理

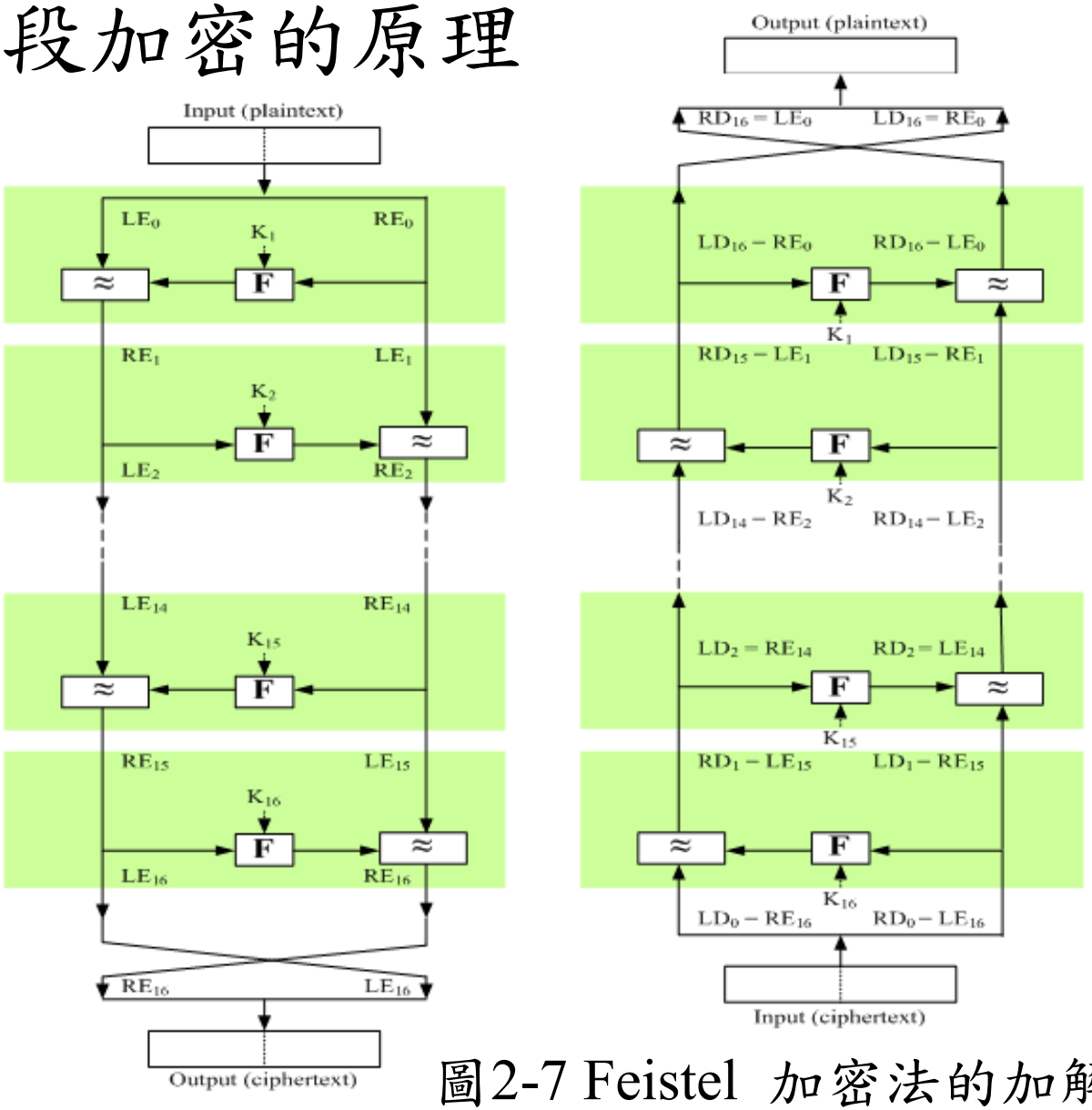


圖2-7 Feistel 加密法的加解密程序

Module 2-2-6: 資料加密標準

2-2-6 資料加密標準

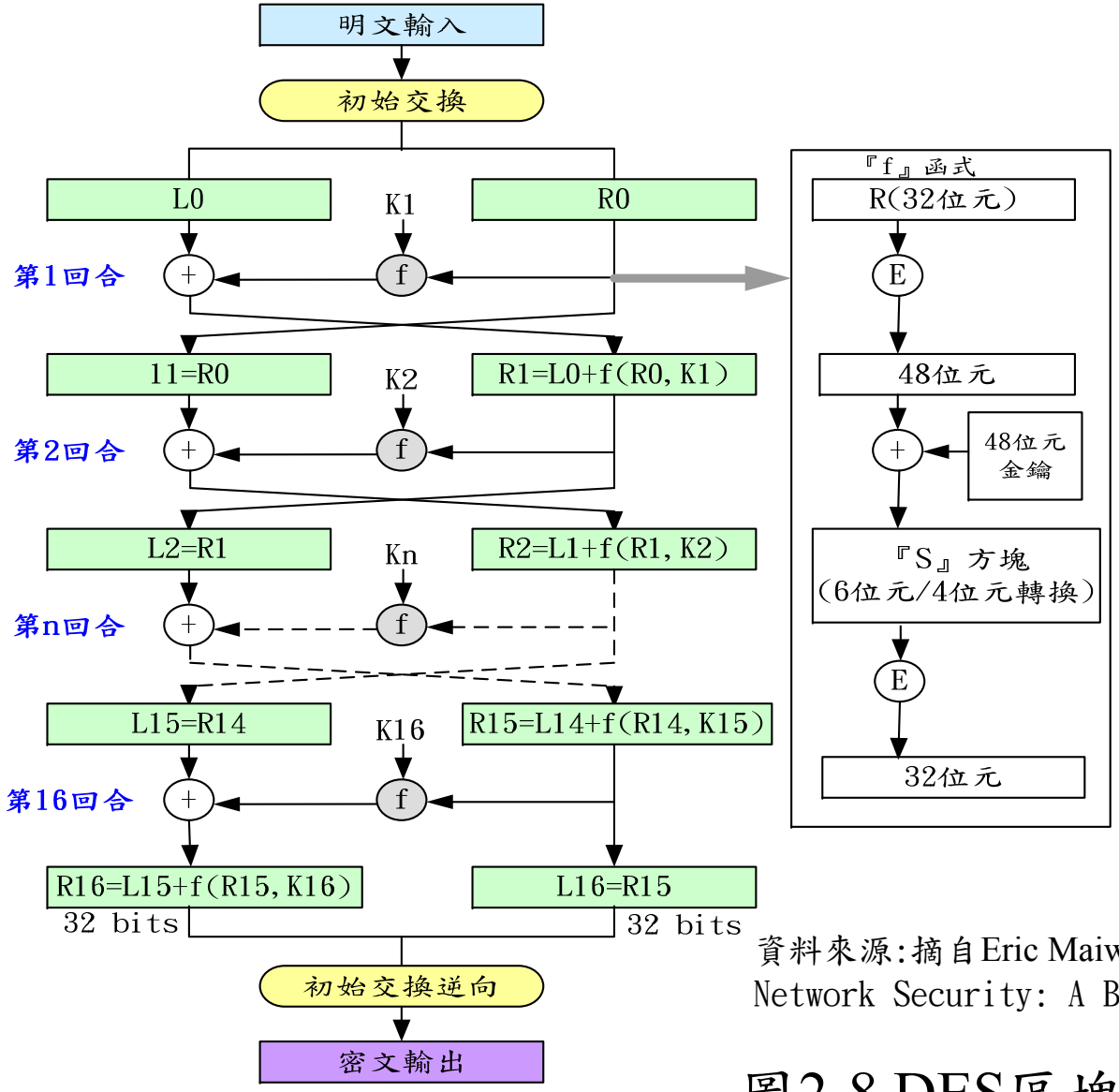
- 資料加密標準（Data Encryption Standard，DES）是由IBM在1970年代初期發展出來的演算法，DES 使用很廣泛，特別在商業領域。
- 在經過NSA審核、修正、認可之後，美國國家標準與技術協會（United States National Institute of Standards and Technology，NIST），在1977年正式採用並做為DES的加密標準 FIPS PUB 46。
- 於1983、1988、1993和1999年再次認定這個標準。

2-2-6 資料加密標準

- 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屬於區塊式密碼（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（詳見圖2-8）。

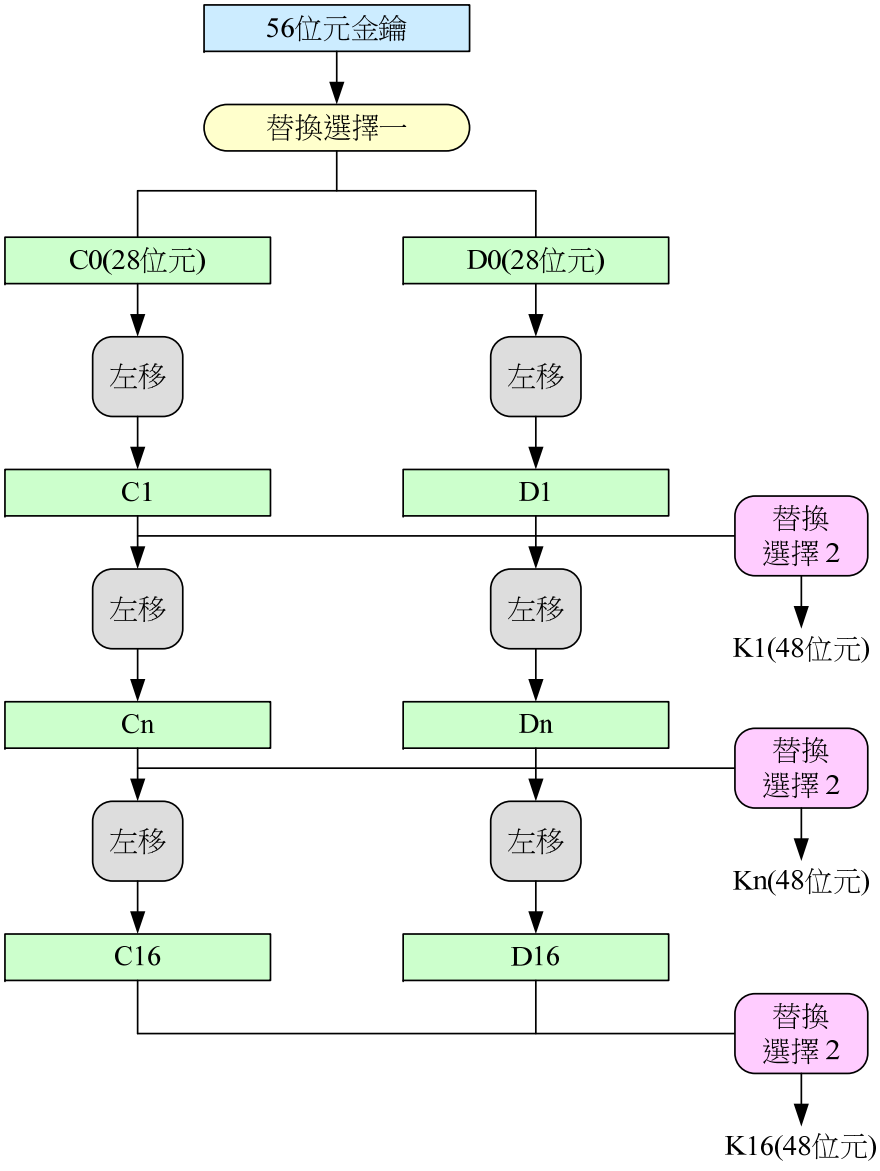
2-2-6 資料加密標準

- 在圖2-8之中可以找到標示為『f』的函式。
- f 函式內含稱為取代(substitution) —『S』區塊如圖2-9 ，這是一個將6位元輸入轉換成4位元輸出的對照表。



資料來源:摘自Eric Maiwald：
Network Security: A Beginner's guide

圖2-8 DES區塊演算法



資料來源:摘自Eric Maiwald：
Network Security: A Beginner's guide

圖2-9 DES產生次金鑰演算法

2-2-6 資料加密標準(**)

啟始重排 IP (Initial Permutation)

- 資料加密的第一步
- 啟始重排會重組輸入資料的位元順序
- 偶數位元移到左半邊 (LH)；奇數位元移到右半邊 (RH)
- 結構上安排非常規律，以利於硬體實作
- 範例:

  IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

2-2-6 資料加密標準(**)

DES 每回合的結構如圖 2-10

- 使用兩半 L 與 R 各 32 位元
- 與任意的 Feistel 加密法一樣，都可以描述成:

  $L_i = R_{i-1}$

  $R_i = L_{i-1}$ xor $F(R_{i-1}, K_i)$
- 利用 32 位元的 R 與 48 位元的子金鑰:
  - 利用重排 E 將 R 擴充成 48 位元
  - 與子金鑰相加
  - 傳給 8 個 S-box 來得到 32 位元的結果
  - 利用 32 位元的重排 P 來重排以上的結果

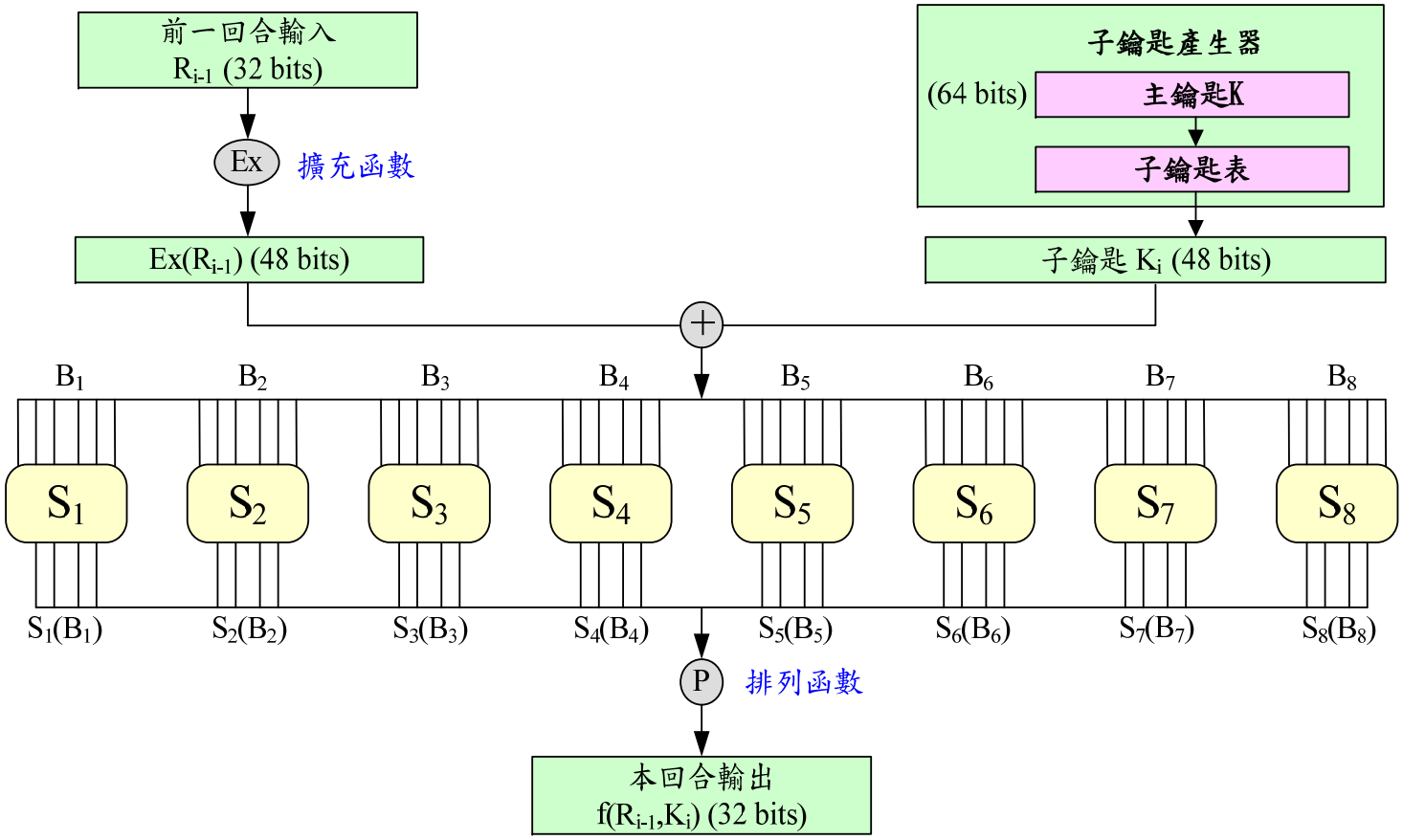


圖 2-10 DES 每回合的結構

2-2-6 資料加密標準(**)

S-boxes

- 共有八個 S-boxes；每個將 6 位元對應到 4 位元。
- 每個 S-box 其實是由 4 個較小的 4 位元方塊所組成
  - 外側的位元 1 與 6 (列 位元) 用來選取一列
  - 內側的位元 2-5 (行 位元) 用來選取一行
  - 產生 8 組 4 位元,也就是 32 位元
- 列的選取受到資料與金鑰的影響，這個性質稱為 autoclaving。
- 範例:

S(18 09 12 3d 11 17 38 39) = 5fd25e03

2-2-6 資料加密標準(**)

DES 的金鑰排程

- 產生每回合所需的子金鑰。
- 由以下程序組成：
  - 對金鑰的啟始重排 (PC1)，其中把 56 位元切成兩半各為 28 位元
  - 接下來的 16 個步驟將會進行：
    - 從每半邊選取 24 位元
    - 依照 PC2 進行重排，並且將結果傳給函式 f
    - 根據金鑰左旋排程 K，將每一半旋轉 1 或 2 個位置

2-2-6 資料加密標準(**)

DES 的解密方式

- 以 Feistel 的設計來說，步驟跟加密一樣，但以相反的順序使用子金鑰 (SK16 … SK1)。
- 以IP還原加密時 FP 的最後一個步驟。
- 第一回合會利用 SK16 來還原加密的第十六回合。
- …
- 第十六回合會利用 SK1 來還原加密的第一回合。
- 最後的 FP 會還原加密時一開始的 IP 。
- 藉此還原成初始值。

2-2-6 資料加密標準(**)

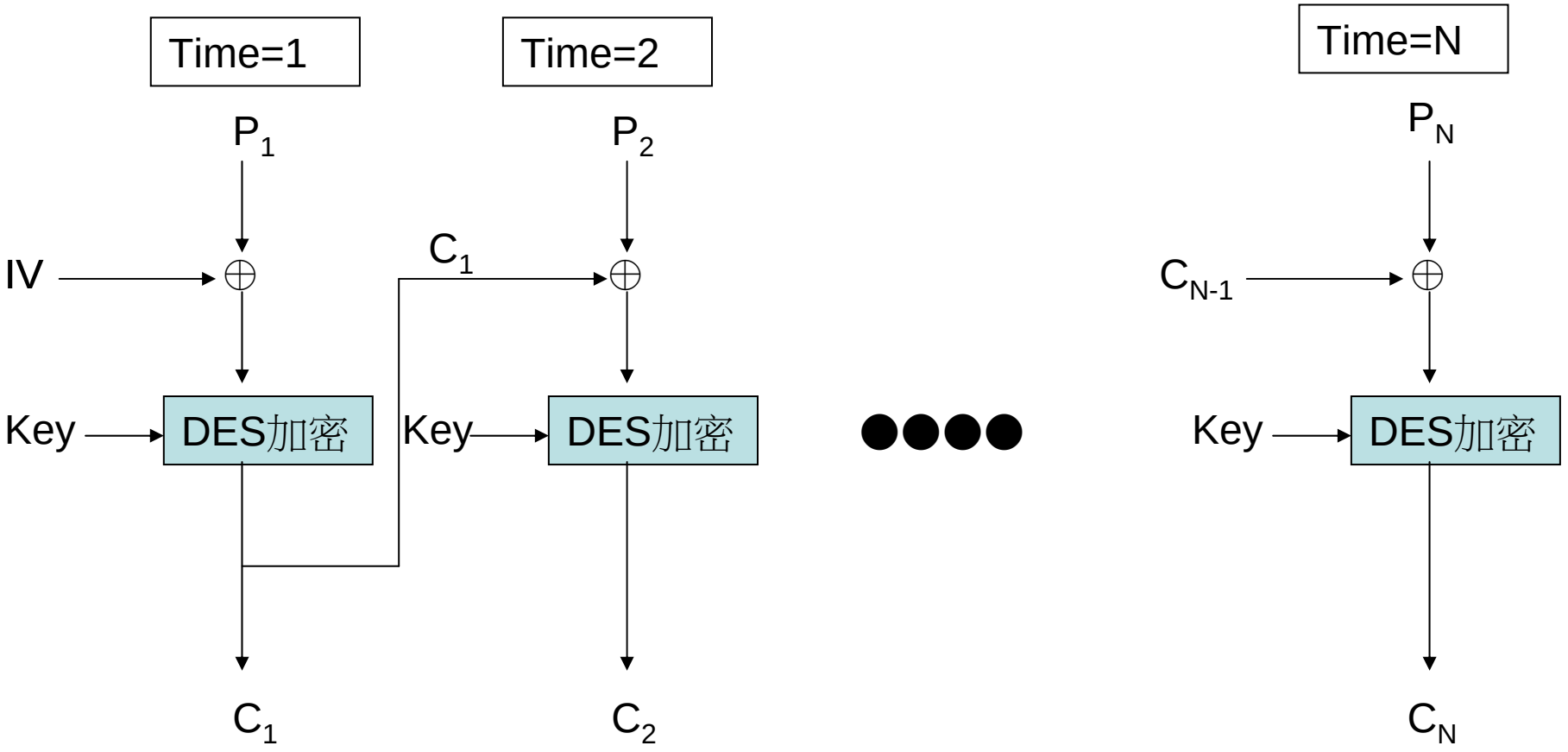
- DES四種運作模式如下：
  - **ECB（Electronic code book）**：這是最基本的區塊加密法，原文和金鑰都視為產生密文的輸入。在這個模式下，相同的輸入會產生相同的輸出。
  - **CBC（Cipher Block Chaining）**：每一個區塊加密之前，必須與前一個區塊的密文做一次XOR運算，之後再進行加密。因此，每一個區塊的加密結果都會受到之前所有區塊內容的影響。如此一來，雖然在資訊中可能出現多次相同的明文，都會因為受到前文的影響而產生不同的密文。

2-2-6 資料加密標準(**)

- 在CBC模式之下，雙方均使用相同的一把金鑰，被破解的可能性增加，故每次通訊前，雙方先協議一個初值（Initialization Vector，IV)
- IV通常加在第一個區塊，每次通訊採用不同的初值，以增加被破解的困難度。
- CBC模式的加密及解密運作程序如下:

2-2-6 資料加密標準(**)

- CBC模式的加密

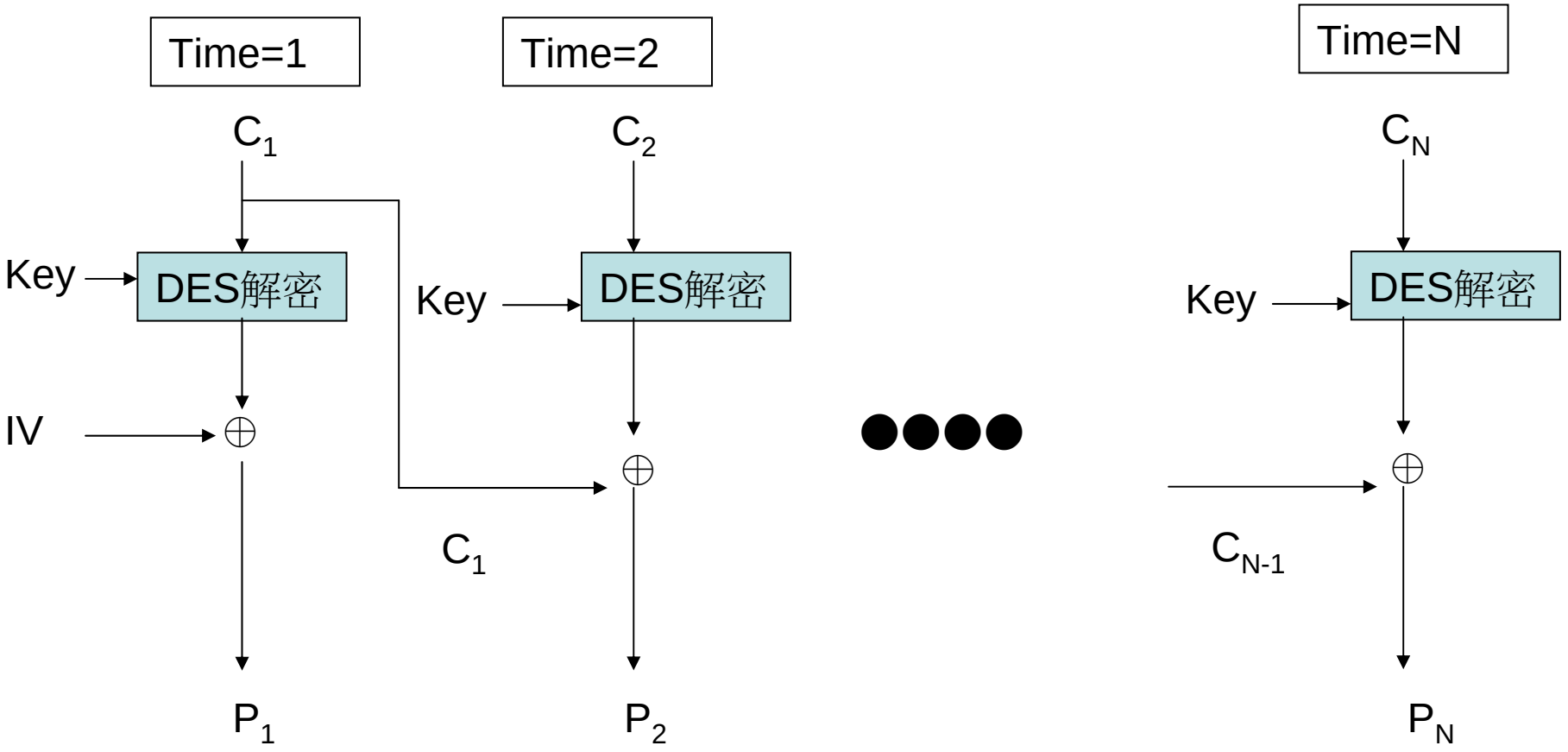


$C_i=E_K(P_i \oplus IV)$, $C_{i+1}=E_K(C_i \oplus P_{i+1})$,

⊕:XOR運算

2-2-6 資料加密標準(**)

- CBC模式的解密



$P_i=D_K(C_i)\oplus IV$, $P_{i+1}=D_K(C_{i+1})\oplus C_i$

⊕:XOR運算

資料來源:摘自W. Stalling: Cryptography and Network Security

2-2-6 資料加密標準(**)

- 可利用密文反饋模式(CFB)或是輸出反饋模式(OFB)將DES轉換成資料流加密法。
  - **CFB（Cipher feedback）**：這個模式會將前一個區塊產生的密文做為DES的輸入，輸出結果是將原文組合成新的密文。
  - **OFB（Output feedback）**：輸出模式類似CFB，好處是傳輸發生位元錯誤不會擴散下去，缺點是比CFB更易遭到訊息修改攻擊。

2-2-6 資料加密標準

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，花幾個月就可破解
  - 1998 年科學家使用特定硬體可在幾天內破解
  - 1999 年科學家結合以上方式可在 22 小時內破解!

2-2-6 資料加密標準(**)

DES 可能遭遇的攻擊--計時破解法(timing attacks)

- 根據我們對加解密的執行時間的瞭解，來推導出部分/全部的子金鑰位元與明文的資訊。
- 破解法是根據以下事實：
  - 不同的輸入值所需加解密的計算時間也不同，Smartcards 尤其會面臨這個問題。
- 目前DES尚可抵擋計時破解法。

2-2-6 資料加密標準(**)

- DES 可能遭遇的攻擊
  - 分析破解法(analysis cryptanalysis)
  - 差異破解法(differential cryptanalysis)
  - 線性破解法(linear cryptanalysis)

2-2-6 資料加密標準(**)

1.分析破解法(analysis cryptanalysis)

- 有許多針對 DES 的分析破解方式，這些方法基本上都算是統計攻擊法，這些破解法取決於細部的加密結構
  - 蒐集與加密法相關的資訊
  - 最終可以解出部分/全部的子金鑰位元
  - 如果需要的話，可以用窮舉法求出剩餘的位元
- 以下我們介紹兩種威力強大且具前景分析破解法
  - 差異破解法
  - 線性破解法

2-2-6 資料加密標準(**)

2.差異破解法(differential cryptanalysis)

- 密碼破解領域近年來公認最大的成就。
- 美國國家安全局(NSA) 在 70 年代就得知此法。
- Murphy, Biham & Shamir 於 1990 提出此破解區段加密法的有效方法。
- 此法對於破解現今大部分的區段加密法來說，已破解部份區段加密法。
- DES 能有效的抵擋此破解法。

2-2-6 資料加密標準(**)

2. 差異破解法特性

- 為一種破解 Feistel 加密法的統計攻擊。
- 運用S-P 網路的設計不良，使得我們可以從函式 *f* 的輸出來計算出輸入與金鑰。

2-2-6 資料加密標準(**)

差異破解法會比較一組加密輸出(m_{i+1}, m_{i+1}')

- 在使用相同金鑰的情況下
- 已知輸入間的差異
- 求出輸出間的差異($\triangle m_{i+1}$)

$$\begin{aligned}\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m_i', K_i)] \\ &= \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m_i', K_i)]\end{aligned}$$

2-2-6 資料加密標準(**)

差異破解法原理如圖 2-11

- 如果某種輸入差異造成某種輸出差異的機率為p。
- 如果某種發生機率很高的輸出入差異組合出現的話，就可以推導出此回合所使用的子金鑰。
- 以此類推進行多回合(機率遞減)，理論上就可以找到f函式使用的子金鑰。

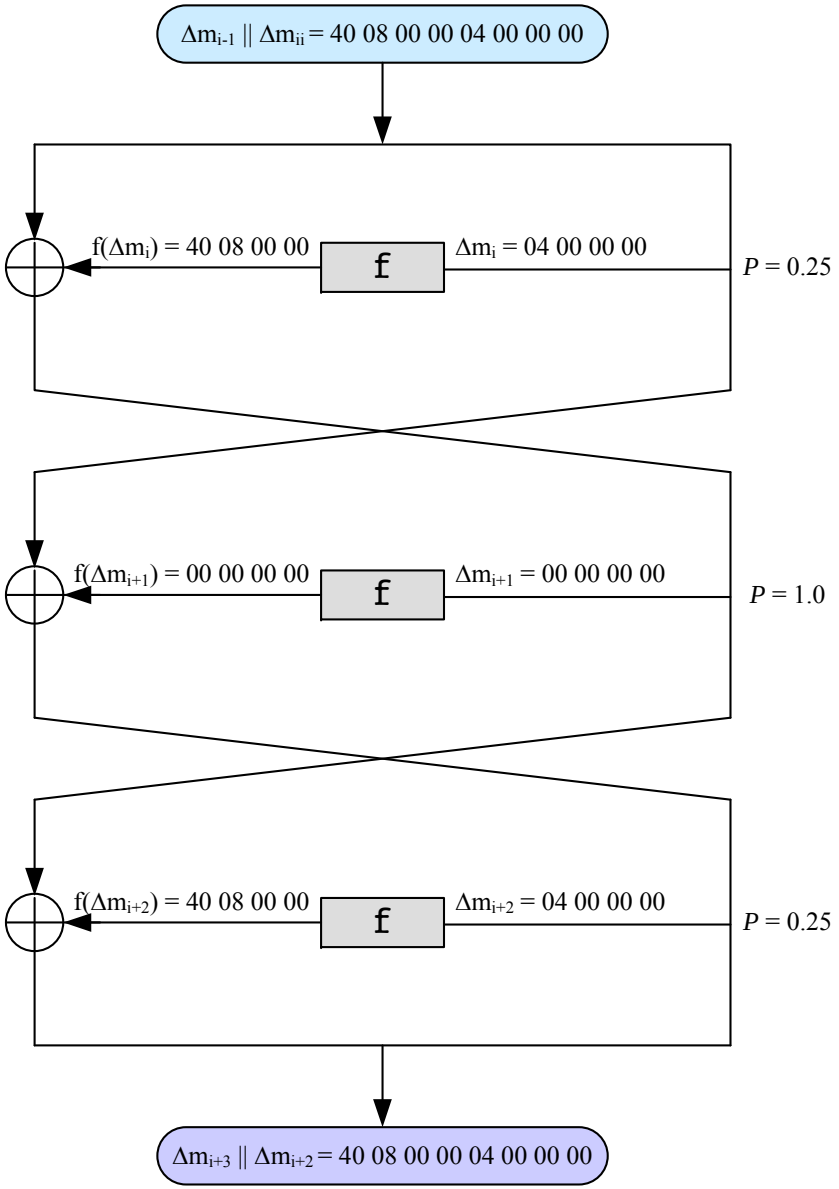

$\Delta m_{i-1} \| \Delta m_{ii} = 40\ 08\ 00\ 00\ 04\ 00\ 00\ 00$
$f(\Delta m_i) = 40\ 08\ 00\ 00$
f
$\Delta m_i = 04\ 00\ 00\ 00$
$P = 0.25$
$f(\Delta m_{i+1}) = 00\ 00\ 00\ 00$
f
$\Delta m_{i+1} = 00\ 00\ 00\ 00$
$P = 1.0$
$f(\Delta m_{i+2}) = 40\ 08\ 00\ 00$
f
$\Delta m_{i+2} = 04\ 00\ 00\ 00$
$P = 0.25$
$\Delta m_{i+3} \| \Delta m_{i+2} = 40\ 08\ 00\ 00\ 04\ 00\ 00\ 00$

圖 2-11 差異破解法

2-2-6 資料加密標準(**)

2. 差異破解法

- 持續地對已知 XOR 結果的輸入組合加密，直到已知 XOR結果的輸出組合出現。
- 當我們看到
  - 如果某回合的中間值符合我們需要的 XOR 結果，就是**正確組合**
  - 如果不符合就是**錯誤組合**，相對比率 S/N 就是可供破解的資訊
- 可以推得此回合使用的金鑰
  - 正確組合表示使用相同的金鑰
  - 錯誤組合提供隨機值

2-2-6 資料加密標準(**)

- 如果回合數眾多的話，機率會變很低，會需要許多的輸入組合，甚至超過 64 位元所能容納的量。
- Biham 與 Shamir 提出了一個如何用13回合的迭代特性來破解完整的16回合DES的方法。

2-2-6 資料加密標準(**)

3. 線性破解法(linear cryptanalysis)

- 為近期的另一項研究成果，必須以機率遞減的方式執行多回合，故也是一種統計方式的破解法。
- 由 Matsui 等人在 90 年代初期發展出來，以求取線性近似值(linear approximation)為基礎來破解DES的金鑰，需要 2^{47} 個已知明文才能破解DES，實際上來說仍舊不實用。

2-2-6 資料加密標準(**)

3. 線性破解法

- 在 機率p 不等於 ½ 的情況下，求出線性近似值

  $P[i_1,i_2,...,i_a](+)C[j_1,j_2,...,j_b] = K[k_1,k_2,...,k_c]$

  此處的 i_a,j_b,k_c 是明文(P),密文(C) ,金鑰(K)中的位元位置

- 列出金鑰位元的線性方程式，利用 max likelihood 演算法來求出一個金鑰位元

- 測試大量的加密結果，破解的機率為: $|p-½|$

Module 2-2-7: Triple DES

2-2-7 Triple DES

- 在1992年研究人員發現重複使用DES ，可建立牢靠的加密法則，因而產生了Triple DES（TDES）。
- TDES可以使用兩組(k1/k3 , k2)或參組金鑰(k1, k2和k3)。如果僅僅使用兩組金鑰，那麼k3和k1是一樣的而只有k2不同。
- 透過混合式加密法，將資訊作取代與重排，這也就是為什麼TDES會比一般DES來得牢靠，成為大家選用的方案。
- 圖2-12 顯示了TDES運作的方式。

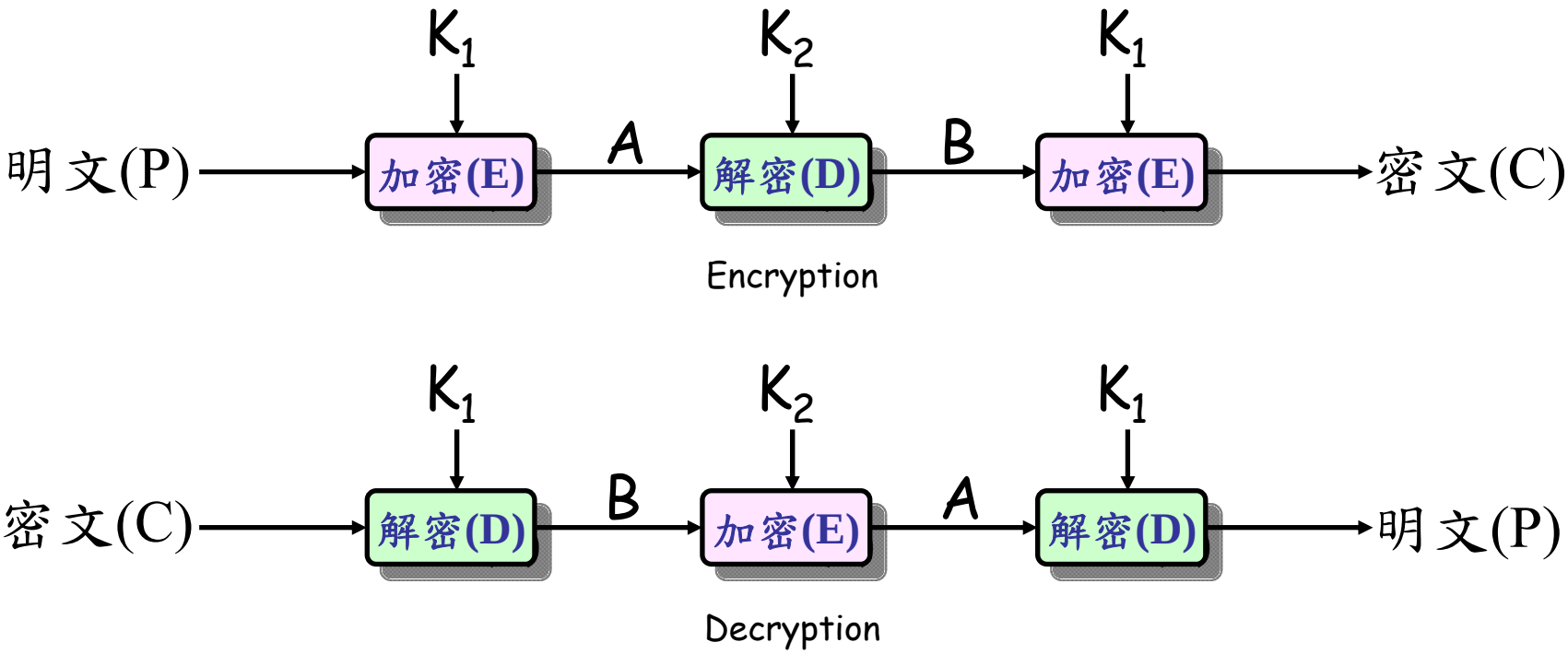


圖2-12 Triple DES方塊圖

資料來源:摘自Eric Maiwald：Network Security: A Beginner's guide

為什麼選用 Triple-DES?

- DES 很明顯地需要一個替代方案
  - 理論上 DES 可破解
  - 金鑰的窮舉搜尋法已被證明可能破解 DES
- 為什麼不用 Double-DES?
  - 雖然與單一的 DES 不同，但是會遭遇中點交會攻擊法
- 中點交會攻擊法(Meet-in-the-Middle Attack)
  - 對「重複使用兩次」的加密法可有效攻擊
  - 因為 $X = E_{K1}[P] = D_{K2}[C]$
  - 攻擊目標為儲存所有金鑰對 P 加密之後的結果
  - 比對所有金鑰對 C的解密結果與 X 是否相符
  - 密碼的執行需要 $O(2^{56})$ 個步驟

使用兩把金鑰的 Triple-DES

- 整個程序需要加密三次
  - 看起來似乎需要三把金鑰
- 但是，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
  - 如果 *K1*=*K2*，則傳統 DES 也可以解讀
- 已標準化為 ANSI X9.17 & ISO 8732。
- 目前尚未見到可行的破解方式。

使用三把金鑰的 Triple-DES

- 雖然沒什麼實際的破解方式，但是使用兩把金鑰的 Triple-DES 安全度較低。
- 如果採用三把金鑰的 Triple-DES 就可以更安心
  - $\mathbf{C = E_{K3}[D_{K2}[E_{K1}[P]]]}$
- 已經被某些網路應用程式採用，例如PGP, S/MIME。

2-2-7 Triple DES

- 和DES相較之下，TDES屬於相對快速的演算法，這是因為可以利用硬體建置TDES。
- 在運算的時候，所需的時間約略是DES的三倍，這是因為含有三次DES運算。
- 目前大多數的應用程式應該選用TDES取代DES。

Module 2-2-8: AES

2-2-8 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，高等加密標準)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，他們提出的Rijmen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、可在硬體設備建置等因素，都是這個演算法獲選的原因。

AES 的特色

- 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 Triple-DES 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。

2-2-8 AES的特色

- 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TDES的最佳選擇。

AES 加密法

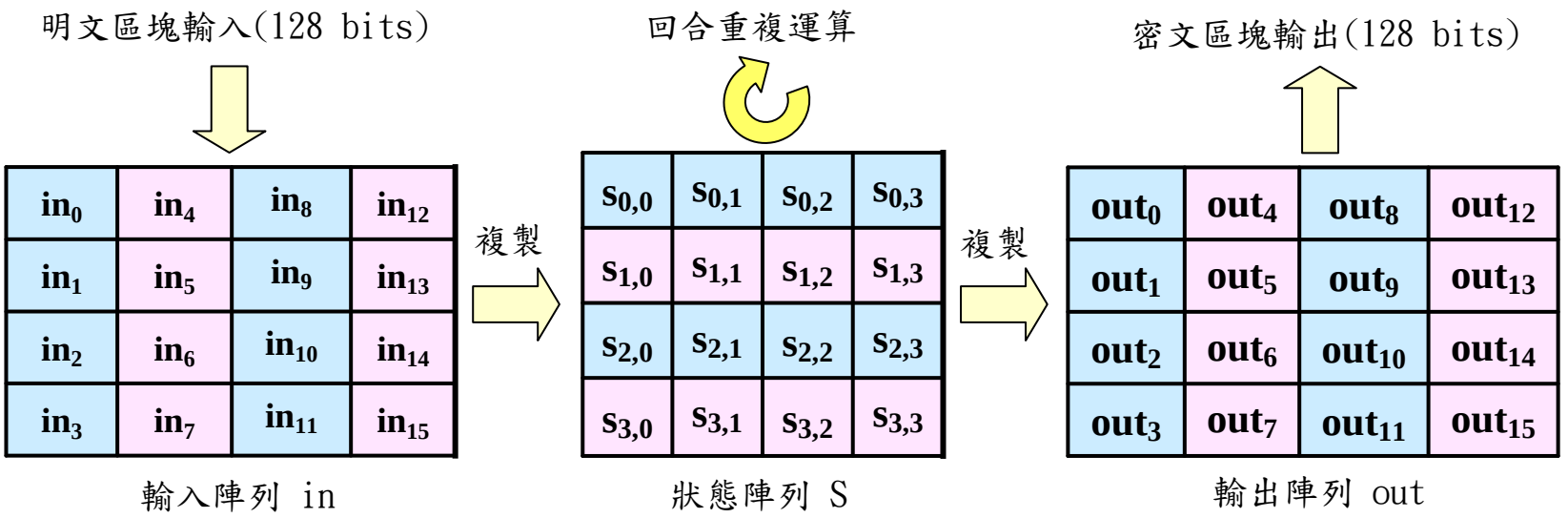
- 為迭代式(**iterative**)加密法、而非 **feistel** 加密法
  - 資料區段為 4 段 4 位元組資料
  - 每回合處理整段資料區段
- 設計目標:
  - 可以抵禦已知的攻擊法
  - 在許多 CPU 上都可以快速執行並且程式碼很小
  - 設計很簡單

AES 加密法(**)

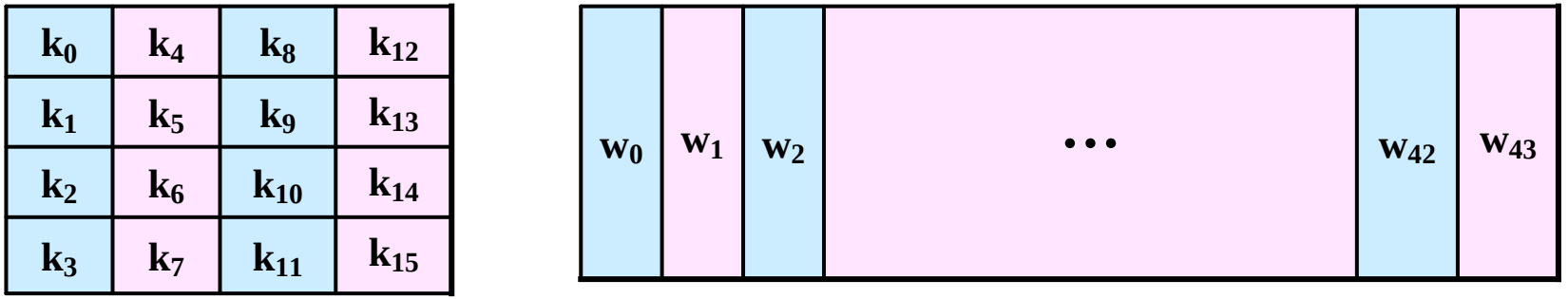
加密程序

- 資料區段為 4 段 4 位元組方形矩陣，亦稱為狀態(state)矩陣如圖2-14。
- 每個狀態矩陣必須經歷 9/11/13 個回合(如圖2-15):
  - 位元組的取代 (每個位元組根據一個 S-box 來運作)
  - 列的移位 (在不同段/行之間重排位元組)
  - 把行內數值混合 (對區段進行矩陣乘法來取代)
  - 採用回合金鑰 (對狀態與金鑰做 XOR 運算)

- 針對金鑰與前一回合的結果作 XOR 運算。
- 所有的運算都以 XOR 或查表形式運作，故快速又有效率。



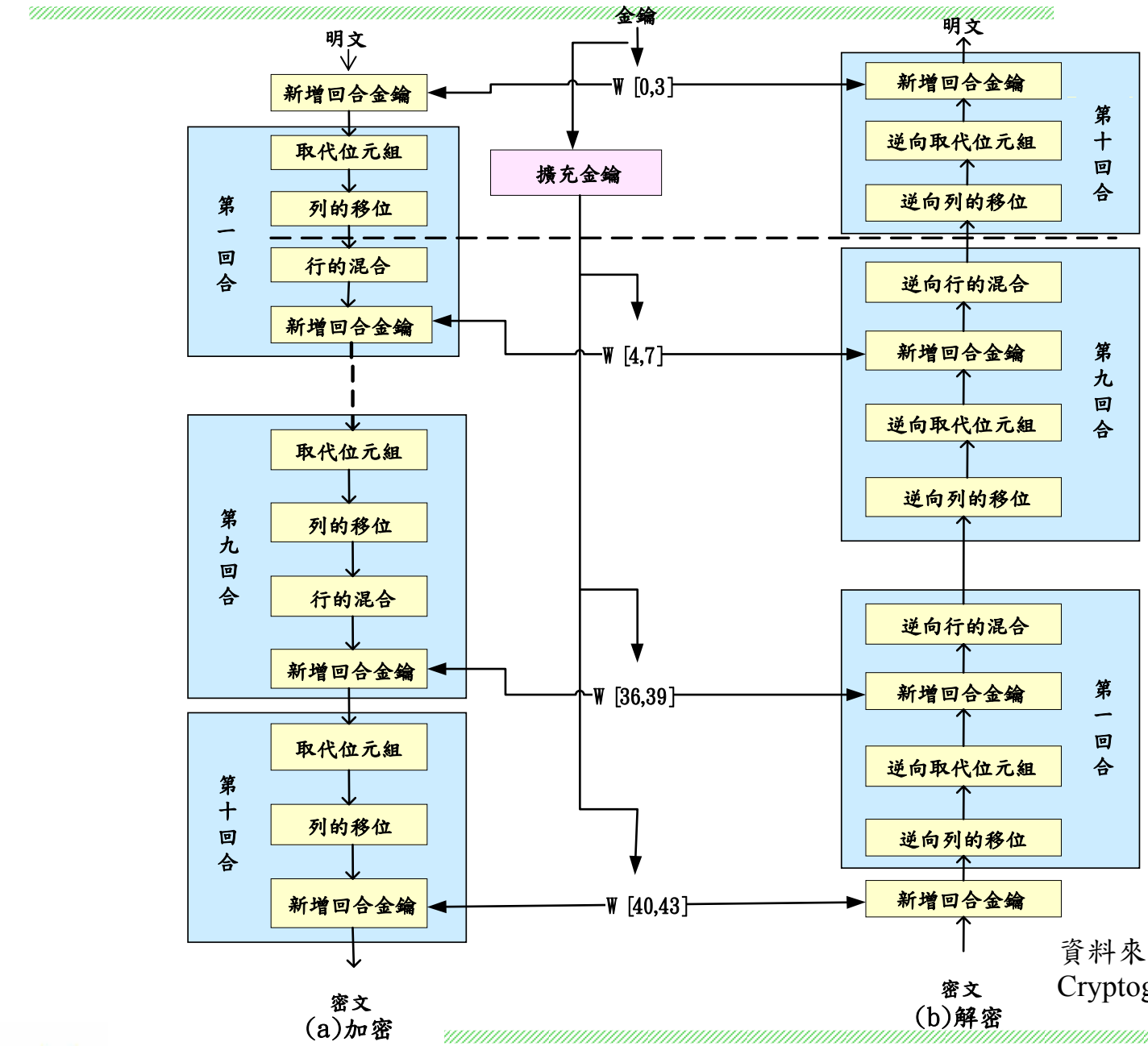
(a)輸入、 State陣列以及輸出



(b) 金鑰與擴充金鑰

資料來源:摘自W. Stalling: Cryptography and Network Security

圖2-14 AES 資料結構



資料來源:摘自W. Stalling:
Cryptography and Network Security

圖 2-15 AES 加解密程序

AES 加密法(**)

AES 加密過程包括(1)位元組的取代 (2)列的移位 (3)行混合 (4)新增回合金鑰 (5)金鑰擴充等步驟，程序如圖 2-16

(1) 加密程序中的位元組的取代(SubsBytes)

- 對每一位元組作簡單的取代
- 使用一個 16x16 位元組的表格，其中包含所有 256 個 8 位元值的重排方式
- 狀態矩陣中的每個位元組都被某列 (由最左邊四個位元來決定) 與某行 (由最右邊四個位元來決定) 的元素來取代
- S-box 是根據 GF(2^8) 中值的運算來定義的
- 設計目標需可以抵擋已知的攻擊方式

AES 加密法(**)

(2) 加密程序中列的移位(ShiftRows)

- 每一列對位元組作循環移位
  - 第一列不更動
  - 第二列循環左移一位元組
  - 第三列循環左移二位元組
  - 第四列循環左移三位元組
- 解密時向右移位。
- 因為是一行一行來處理，這個步驟會重排行之間的位元組。

AES 加密法(**)

(3) 加密程序中的行混合(MixColumns)方式

- 每一行獨立處理
- 每一位元組都被另一數值取代
- 有效地執行有限體$GF(2^8)$上的矩陣乘法，採用質式 $m(x) = x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

AES 加密法(**)

(4) 加密程序中的新增回合金鑰(Add round key)方式

- 對狀態矩陣與 128 位元的回合金鑰作 XOR 運算
- 同樣地，逐行來處理 (一系列有效的位元組計算)
- 解密程序完全一樣，因為 XOR 就是自身的逆運算，可以正確地還原回合金鑰
- 新增回合金鑰設計得愈簡單愈好

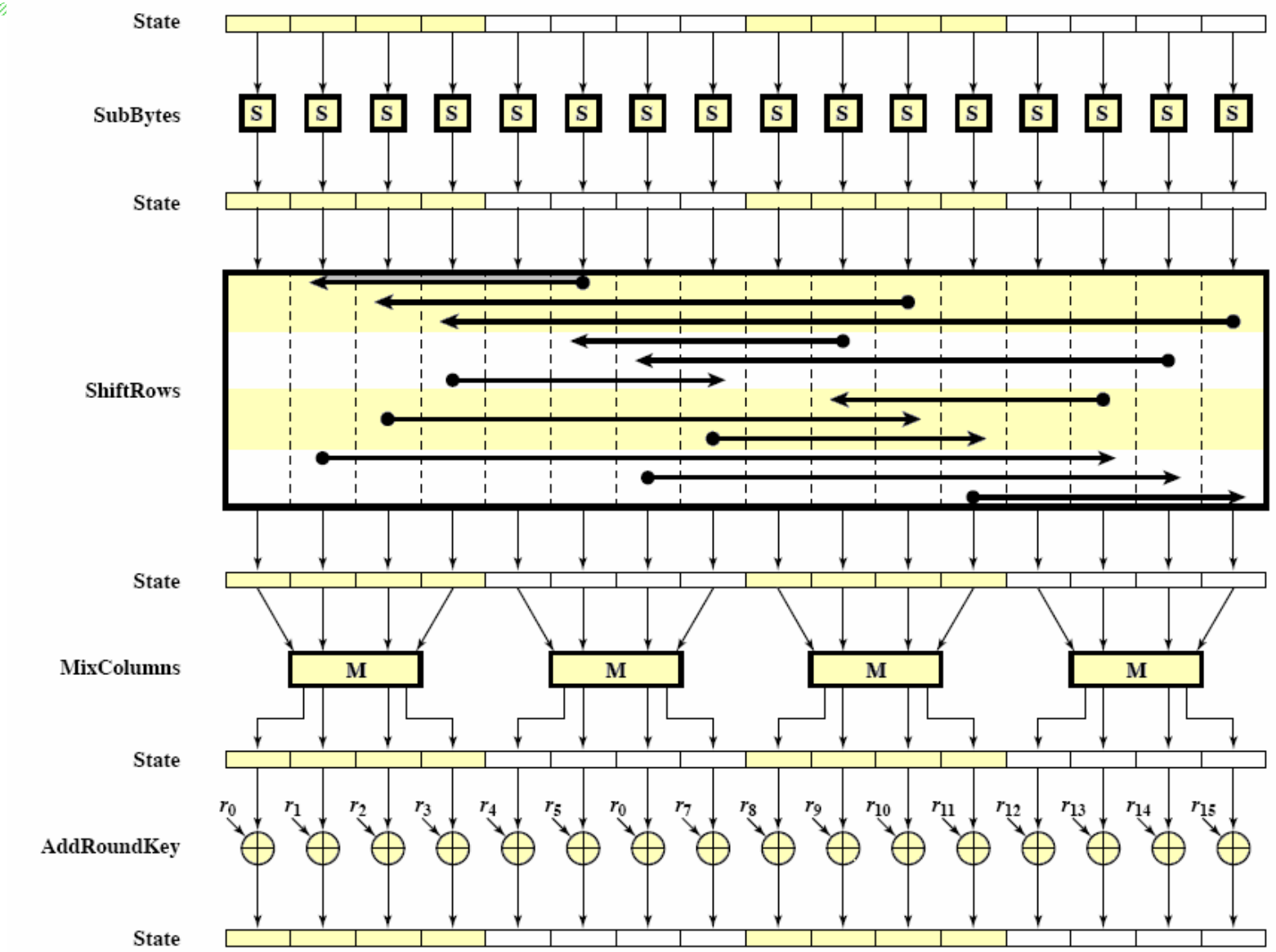


圖 2-16 AES 的加密回合架構

AES 加密法(**)

(5) AES 的金鑰擴充(Key expansion)方式

- 將 128 位元 (16 位元組)的金鑰擴充至44/52/60 個 32 位元字元的陣列中
- 首先將輸入的金鑰複製至擴充金鑰的前四個字元
- 然後根據前四個與目前的四個字元依序將字元填入
  - 在 4 個情況的其中三個，直接對他們做 XOR 運算
  - 每逢四的倍數，就將目前的四個字元與前四個字元的 S-box + 旋轉+某常數 XOR 的結果 XOR 在一起
- 設計成可以抵擋已知的攻擊方式

AES 加密法(**)

AES 的解密(圖 2-15)

- AES 的加解密流程並不同，缺點是同時需要加解密，則需要使用不同的模組。
- 但是有一軟體版本，可以定義一個與加密步驟相同的等價解密運算，但須滿足下列需求
  - 必須使用每個步驟的逆運算
  - 金鑰的產生與使用順序必須改變
- 經驗證相同的等價解密運算是可行的，因為符合下列條件時結果不變
  - 將位元組的取代與列的移位互相交換
  - 將行的混合與回合金鑰的使用互相交換

AES 加密法(**)

實作上的考量

- 可以在 8 位元的 CPU 上有效地實作
  - 使用一個 256 元素的表格，進行位元組取代
  - 列的移位就等於位元組的移位
  - 可用 XOR 運算來加入回合金鑰
  - 行的混合需要 GF(2^8)上的矩陣乘法運算，其需要針對位元組來計算，這只要用查表方式就可以很簡單地解決

AES 加密法(**)

實作的觀點

- 可以在 32 位元的 CPU 上有效地實作
  - 重新針對 32 位元字元來定義每個步驟
  - 可以預先算好四個 256 字元的表格
  - 這樣一來，每個回合的每個行的計算，就可以用查表 4 次與 4 個 XOR 計算來解決
  - 需要 16Kb 來儲存表格
- 設計者認為，其有效率的實作方式是Rijndael雀屏中選為AES的關鍵。

Module 2-2-9: 其它私密金鑰演算法

2-2-9 其它私密金鑰演算法

- 除前述DES、TDES 、 AES外，在安全系統上可以選用的私密金鑰演算法如下：
  - **IDEA（International Data Encryption Algorithm，國際資料加密演算法）**：是由瑞士發展出來的。IDEA使用128位元的金鑰，所以也可以應用在PGP（Pretty Good Privacy）。
  - **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。

2-2-9 其它私密金鑰演算法

- **Blowfish**：Blowfish允許使用的金鑰長度最長可達448位元，且在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，且應用在較為新版的PGP之中。
- **GOST**：GOST用來回應DES的蘇聯標準，使用256位元金鑰。

Module 2-3: 非對稱式加解密法

Module 2-3-1: 公開金鑰加密法

Module 2-3-2: RSA演算法

Module 2-3-1: 公開金鑰加密法

私密金鑰加密法的缺點

- 傳統 私密/秘密/單一金鑰 加密法只用一把金鑰，傳送者與接收者共用這把金鑰，也稱為對稱式加密。
- 如果這把金鑰被破解，那麼通訊安全就岌岌可危。
- 對稱式加密法的缺點：
  - 無法確認資訊傳送者
  - 接收者可以偽造訊息並宣稱是由傳送者所送出的

公開金鑰加密法

- 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。
- 因為兩造不對等，所以稱為非對稱式(asymmetric)加密法。
- 運用數論(number theory)的觀念應用在函式上。
- 與其說是取代對稱式加密法，不如說是與對稱式加密法互補，兩種方法的比較如表 2-2。

公開金鑰加密法

- 公開金鑰/雙金鑰/非對稱式加密法會使用兩把金鑰：
  - 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如圖 2-17
  - 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）簽章如圖 2-18
- 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。

傳統加密法	公開金鑰加密法
運作的條件： 1. 加解密都採用相同的演算法及鑰匙。 2. 傳送者與接收者必須共用演算法及金鑰。 **如何保證安全性：** 1. 必須保護金鑰。 2. 如果沒有可用的資訊的話，無法(或是很難)解開密文。 3. 即使知道演算法與某些密文，也無法求得金鑰。	**運作的條件：** 1. 使用同一個演算法與**一組金鑰**來進行加解密，其中一把金鑰用來加密，別一把用來解密。 2. 傳送者與接收者必須各自持有同組金鑰中的某一把(持有的金鑰不同)。 **如何保證安全性：** 1. 必須保護一組金鑰的某一把。 2. 如果沒有可用資訊的話，無法(或是很難)解開密文。 3. 即使知道演算法、某些密文、與某一把金鑰，也無法求得另一把金鑰。

表 2-2 傳統加密法與公開金鑰加密法

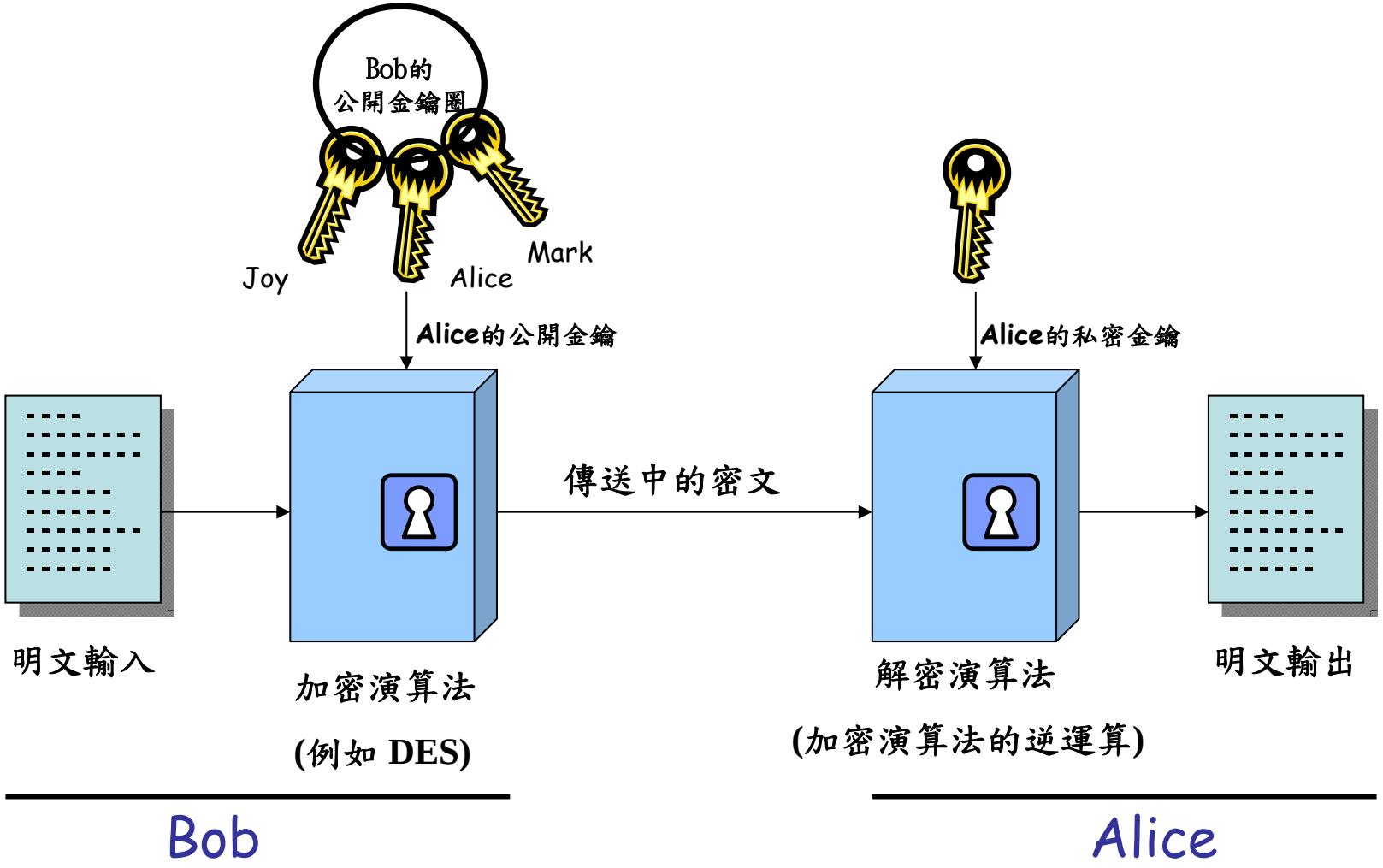


圖 2-17公開金鑰加密法—加密與解密

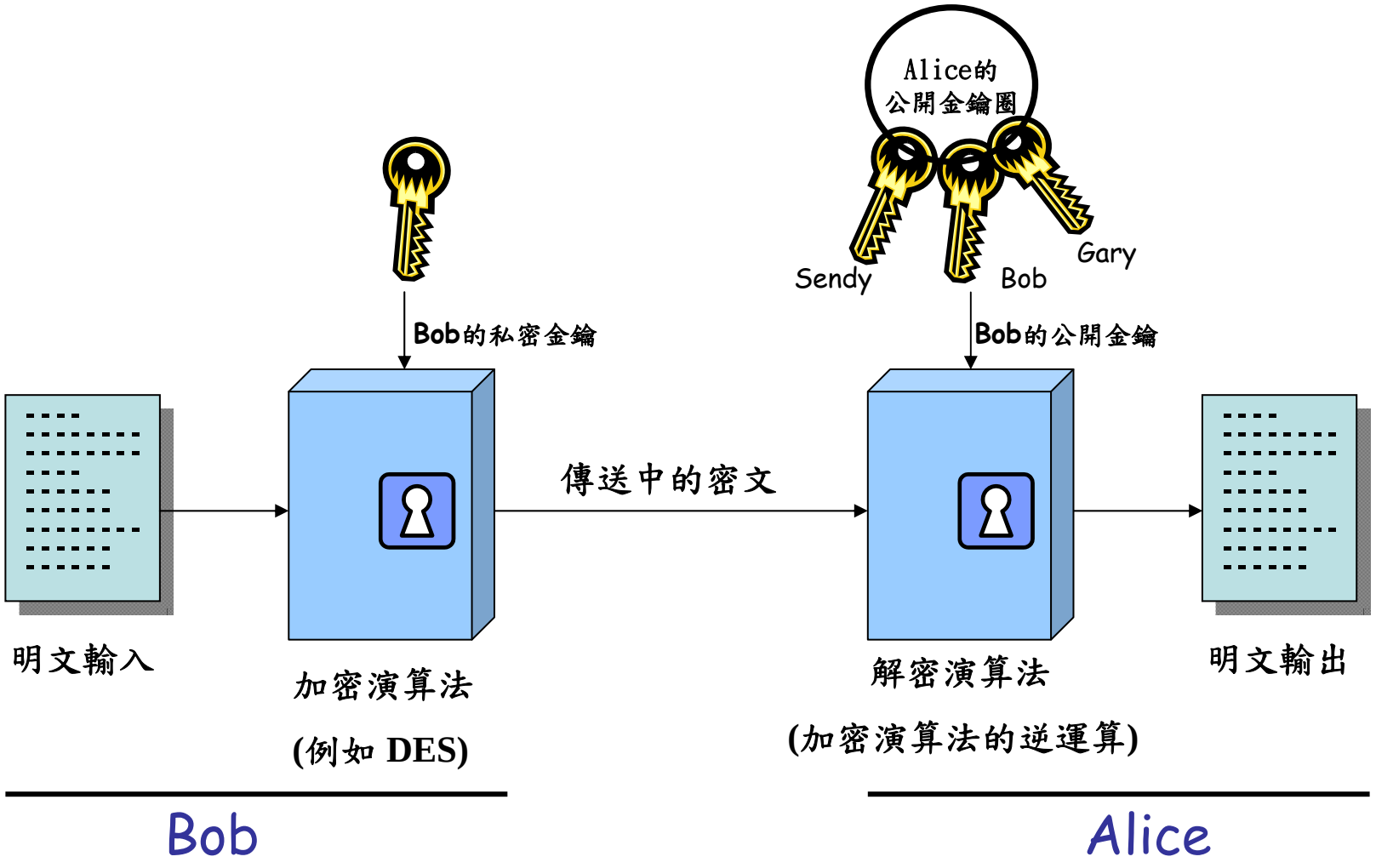


圖 2-18 公開金鑰加密法—簽章與驗章

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
- 公開金鑰加密法是 Whitfield Diffie 與 Martin Hellman 於 1976 年在 Stanford 大學所發明的
  - 其實在這之前，某些社群內已發現了這個方法

公開金鑰加密法的特性

- 公開金鑰加密法需要兩把金鑰，並且具備以下特性：
  - 如果只知道演算法與加密金鑰，要求出解密金鑰這件事在計算上是不可能的
  - 在已知相關的加解密金鑰時，訊息的加解密是很容易計算的
  - 同一組內的任何一把金鑰都可以拿來加密，而用另一把金鑰來解密（在某些機制下）
- 公開金鑰加密、解密及簽章、驗章如圖2-19

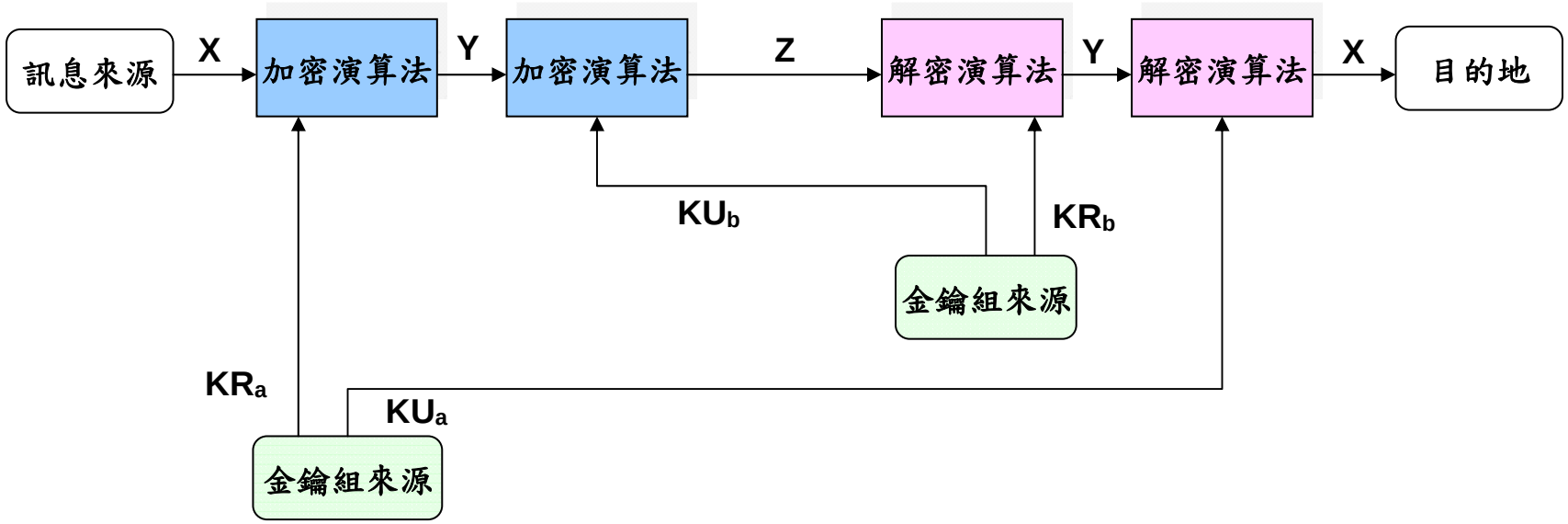


圖2-19 公開金鑰加密系統:安全性與確認性

公開金鑰加密法的應用

- 公開金鑰加密法的應用可分為三類：
  - **加密/解密** (提供安全性)
  - **數位簽章** (提供確認性)
  - **金鑰交換** (針對通訊金鑰)
- 有些演算法適用於三種的應用，某些方法則只適用三種中的一種或兩種。
- 表2-3說明RSA及其他演算法的應用。

公開金鑰加密法的應用

演算法	加解密	數位簽章	金鑰交換
RSA	是	是	是
橢圓曲線法	是	是	是
Diffie-Hellman	否	否	是
DSS	否	是	否

表2-3 公開金鑰加密系統的應用領域

公開金鑰機制的安全性

- 與私密金鑰法一樣，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 (> 512 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，執行速度會較私密金鑰機制慢。

Module 2-3-2: RSA演算法

RSA演算法

- Rivest、Shamir及 Adleman 於 1977 年在 MIT 所發明的。
- RSA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
  - 指數運算需要 $O((\log n)^3)$ 個運算 (簡單問題)
- 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
  - 因數分解需要 $O(e^{\log n \log \log n})$ 個運算 (困難問題)

RSA 的金鑰設定

- 使用者以下列方式來產生公開與私密金鑰(圖2-20)
- 任意選取兩個大質數 p與q，計算其模數 N=p.q
  - 請注意 ø(N)=(p-1)(q-1)
- 任意選取加密金鑰 e
    - 此處 1<e<ø(N), gcd(e,ø(N))=1
- 解下列等式來求其解密金鑰 d
  - e.d=1 mod ø(N) 且 0≤d≤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} ，保存秘密解密金鑰：KR={d,N}

2-3 RSA加解密運算

- 針對機密性的基本加密演算法如下(圖2-21)：
  - 密文C = (原文M)^e mod N
  - 原文M = (密文C)^d mod N
  - 公眾金鑰 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。

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

圖2-20 RSA金鑰產生

加密	
明文	$M<N$
密文	$C=M^e \pmod N$

解密	
密文	C
明文	$M=C^d \pmod N$

圖2-21 RSA加解密

RSA加解密運算

- 傳送者以下列方式加密訊息 M：
  - 取得**接收者**的公開金鑰 KU={e,N}
  - 計算 $C=M^e \pmod N$, 其中 $0 < M < N$
- 接收者以下列方式解開密文 C：
  - 使用其私密金鑰 KR={d,N}
  - 計算 $M=C^d \pmod N$

RSA 運算原理(**)

- 因為 RSA 運算使用Euler 定理：
  - $a^{ø(n)} \bmod N = 1$
  - 此處 gcd(a,N)=1
- RSA 具有下列特性：
  - N=p.q
  - ø(N)=(p-1)(q-1)
  - 謹慎選取互為反元素（ e.d=1 mod.ø(N)）的 e 與 d
  - 對某個k來說，e.d=1+k.ø(N)
- 下列的關係恆成立

∵$C=M^e \bmod N \rightarrow M=C^d \bmod N = (M^e)^d \bmod N = M^{1+k.ø(N)} \bmod N$

依據Euler 定理的引理 $M^{1+k.ø(N)} = M \bmod N$

故$M=C^d \bmod N = M \bmod N$ 故得證！

RSA 範例

1. 選取質數： p=17 與 q=11
2. 計算 n = pq =17×11=187
3. 計算 ø(n)=(p–1)(q-1)=16×10=160
4. 選取 e 使得 gcd(e,160)=1; 選取 *e*=7
5. 求取 d 使得 *de*=1 mod 160 且 *d* < 160
   其值為 d=23 因為 23×7=161= 10×160+1
6. 發佈公開金鑰 KU={7,187}
7. 保存秘密私密金鑰 KR={23, 187}
   （參考圖2-22）

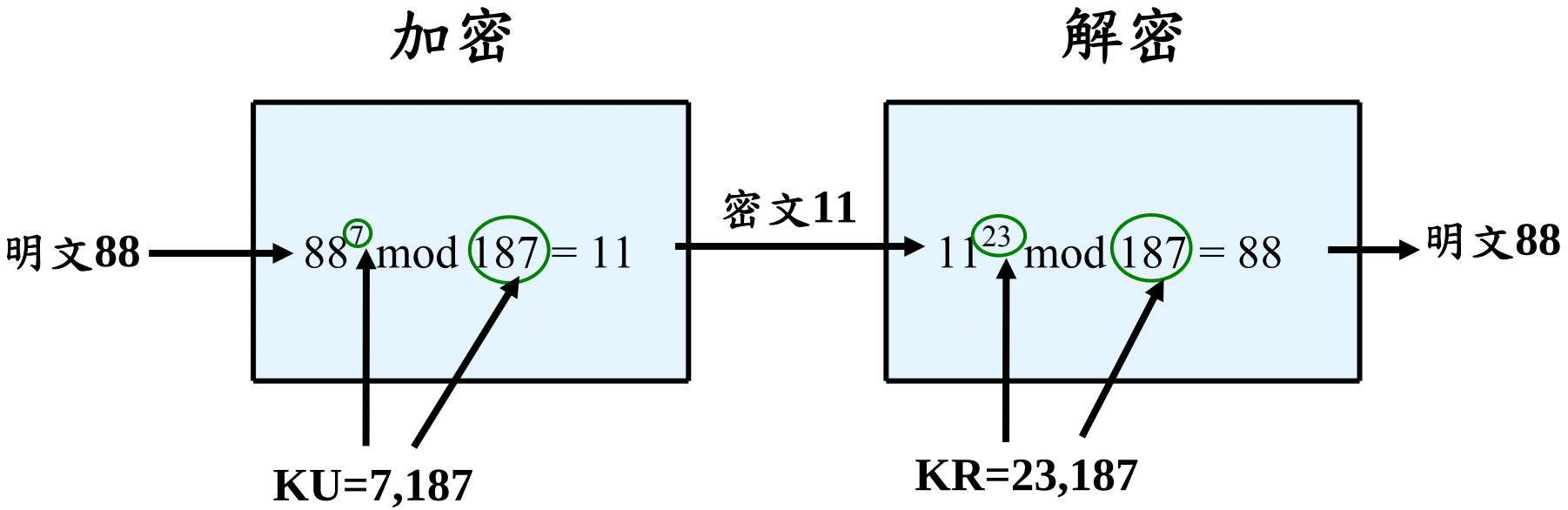

加密
解密
明文88
88^7 mod 187 = 11
密文11
11^23 mod 187 = 88
明文88
KU=7,187
KR=23,187

圖2-22 RSA範例

RSA 範例

- RSA 的加解密範例：
- 已知訊息 M = 88 (88<187)
- 加密：

  $C = 88^7 \bmod 187 = 11$
- 解密：

  $M = 11^{23} \bmod 187 = 88$

RSA計算效率(**)

指數運算

- 加解密計算均利用到整數的次方，再取n的同餘，若次方數很大，可利用「平方與乘法演算法」。
- 使用「平方與乘法演算法」，這是一個快速有效的指數計算法。
- 基本觀念是持續地計算平方，必要時再計算乘法來產生結果。
- 以二進位表示法來看待指數，對 n 來說，只需要 $O(\log_2 n)$ 次乘法
  - 例如，$7^5 = 7^4.7^1 = 3.7 = 10 \bmod 11$
  - 例如，$3^{129} = 3^{128}.3^1 = 5.3 = 4 \bmod 11$

2-3 RSA與確認資訊來源(**)

- 值得一提的是RSA也可以逆向操作，即可確認資訊的來源。
- 每個人都可以用對方的公眾金鑰將資訊解密，如果驗證無誤，就可以確定這個資訊是由金鑰對的主人所發出的，可確認資訊來源及不可否認性。

RSA 的安全性(**)

- 有三種攻擊 RSA 的方法
  - 暴力搜尋金鑰法(brute-force attacks)：嘗試所有可能的私鑰，除非有重大突破，私鑰長度大於1024位元的 RSA應該是安全的。
  - 數學攻擊法(mathematical attacks):取決於計算 Ø(N)的困難度。因為在求 N 同餘的情況下，因數分解很困難。
  - 計時攻擊法(timing attack):根據解密的運算時間推算key。

計時攻擊法(**)

- Paul Kocher證明運用不同運算間的時間差找出私密金鑰，它是一種「只知道密文」的攻擊方式。
- RSA 會透露指數運算時所耗用的時間。
- 反制計時攻擊法的方法
  - 固定的指數計算時間(constant exponentiation time):在回傳計算結果前，確定所有指數計算使用相同的時間。
  - 隨機的延遲(random delay):於指數計算中，加入隨機的延遲。
  - 使用障眼法(blinding):於指數運算前，將密文乘以一個隨機的數字， 避免遭受到一位元一位元的攻擊方法；RSA Data Security於部份的產品加入此功能。

Module 2-3-3: 其他公眾金鑰演算法

2-3-3 其他公眾金鑰演算法

- 還有其它幾種具有Diffie-Hellman交鑰交換法(第三章介紹)和RSA相同效果的演算法。
- 三種較為流行的演算法如下：
  - Elgamal公眾金鑰演算法
  - 數位簽章演算法(DSA)
  - 橢圓曲線加密(ECC)

Elgamal公眾金鑰演算法(**)

- Taher Elgamal屬於Diffie-Hellman的變形。這個演算法強化了Diffie-Hellman系統，最後也具有加密的功能，且演算法之一還具有確認來源的能力。
- Elgmal並沒有專利權（RSA有專利權），因此是一個較為省錢的選擇（不需要支付專利費用）。
- 這個演算法是以Diffie-Hellman為基礎，因此也是屬於離散對數的計算問題。

數位簽章演算法(**)

- 數位簽章演算法（Digital Signature Algorithm，DSA）是美國政府發展的數位簽章標準（詳見下一節數位簽章）。
- 這個演算法是Elgamal的變形，不過只能做為身份確認用途，且不能用於機密性資訊的保護用途。

橢圓曲線加密(**)

- Neal Koblitz和Victor Miller在1985年分別獨立提出橢圓曲線加密（Elliptic curves Encryption）構想。
- 一般認為，Elliptic Curve Cryptosystems（ECC）有別於因式分解和離散對數的數學計算。
- 計算問題如下：假設橢圓曲線的兩個點A和B，產生的計算式為A＝kB，若給定A跟B，很難找到整數k。
- 使用ECC的效益會比RSA或Diffie-Hellman來得好。

橢圓曲線加密(**)

- 在相同的安全等級情況下，ECC的金鑰長度更短（這就是ECC問題的困難點），計算速度也比較快。
- 在普遍接受ECC之前，還必須深入研究和調查，而且ECC也已具有許多專利。

Module 2-4: 認識數位簽章

2-4 認識數位簽章

- 數位簽章（Digital Signature）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。
- 數位簽章逐漸流行，且是邁向無紙化環境的重要關鍵。
- 前美國總統柯林頓曾經簽署過，讓數位簽章具有法律效力的法案。

2-4 數位簽章

- 數位簽章是一種利用電子資訊加密的身份確認法則。
- 如果使用某人的私密金鑰將資訊加密，可以使用某人的公開金鑰將資訊解密，也就可以確認資訊來源的私密金鑰擁有人。
- 如果資訊可以正確地解密，那也代表資訊的內容沒有修改過，因此也可提供完整性的保護。
- 有了數位簽章之後，我們會希望在接收資訊和解密之後，並利用數位簽章進一步保護資訊不會遭到竄改。

2-4 數位簽章

- 圖2-23顯示如何達成數位簽章的可能方式，首要的步驟是雜湊（hash）訊息的功能。
- 經過雜湊運算之後，會產生訊息的檢查碼(MAC)，再使用MAC做為私密金鑰加密的資料來源。
- 最後將訊息和加密之後的檢查碼一起傳送給資訊的接收端。
- 在接收端經解密取得資訊之後，可將資訊輸入到相同的雜湊函式之中，計算出檢查碼。

2-4 數位簽章

- 接著利用公開金鑰將加密之後的檢查碼解密，比對兩者的檢查碼是否相同。
- 如果檢查的結果相同，表示資訊沒有遭到修改。
- 數位簽章的關鍵因素如下：
  - 使用者私密金鑰的保護
  - 安全的雜湊函式
- 有關數位簽章標準(DSS)及數位簽章演算法(DSA)的詳細內容，將於第五章訊息鑑別作詳細介紹。

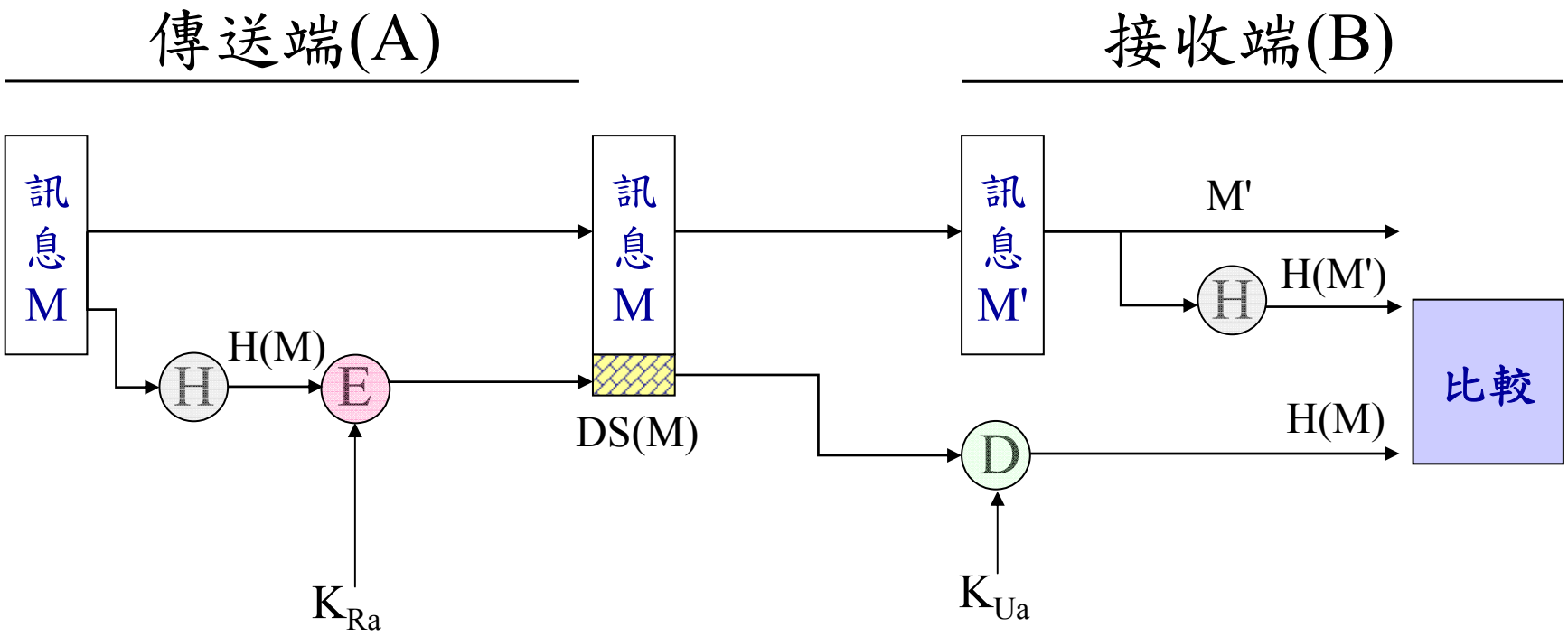


圖2-23 數位簽章運算

資料來源:摘自W. Stalling:
Cryptography and Network Security

Module 2-5: 雜湊函式

雜湊函式

- 雜湊函式：訊息鑑別碼變形就是單向的雜湊函式(Hashing Function)，雜湊函式將一個可變長度的訊息產生固定長度的雜湊碼，功能具備錯誤偵測。
- 與訊息鑑別碼不同處，雜湊函式不使用私密金鑰加密，而是使用輸入訊息函式。
- 雜湊碼亦稱訊息摘要(message digest)或雜湊值(hashing value)。

雜湊函式

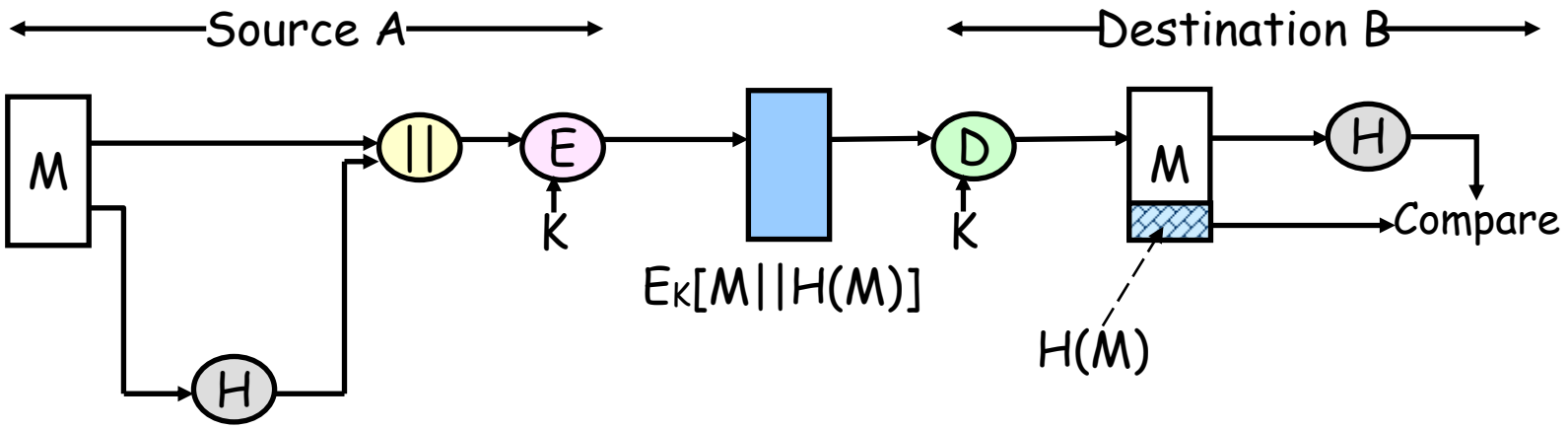
- 通常會假設雜湊函式是已知的、且不需要金鑰。
- 雜湊常用來偵測訊息的變動。
- 具有各種處理訊息的方式，最常用來產生數位簽章。

雜湊函式的特性

- 雜湊函式用來產生檔案/訊息/資料的「指紋(fingerprint)」。

  $$h = H(M)$$

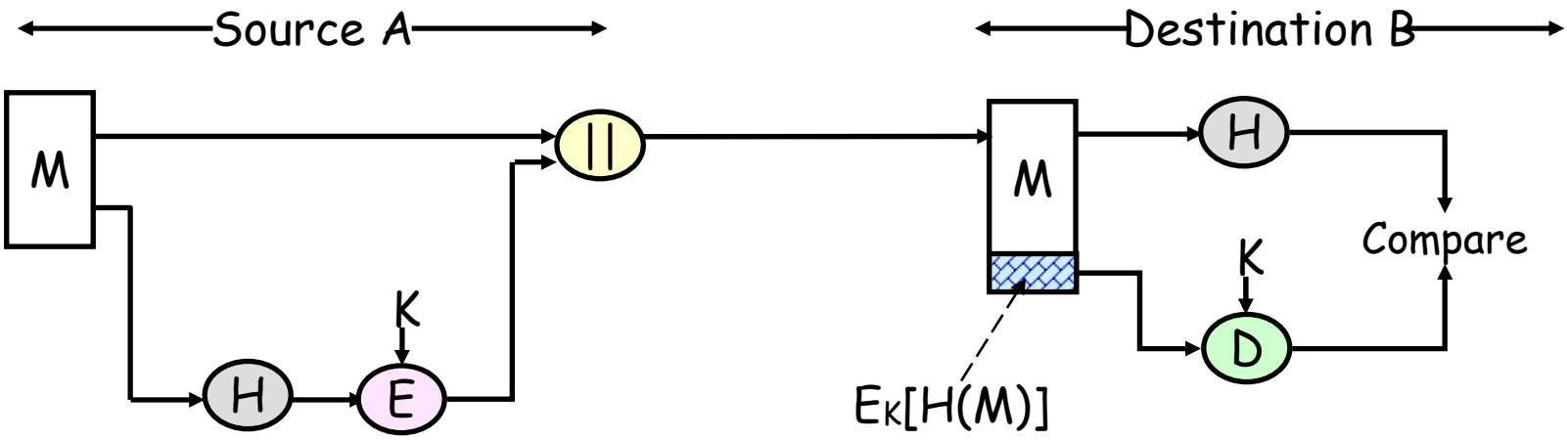
  - 壓縮不定長度的訊息 M
  - 產生固定長度的指紋 h
- 雜湊函式是公開的，故須保護資訊的雜湊值。
- 圖2-24(a)~(c)說明雜湊函式的基本運作。



資料來源:摘自W. Stalling:
Cryptography and Network Security

1. 明文經由雜湊函式運算得出雜湊碼
2. 將雜湊碼及明文結合，並以對稱式加密法加密後傳送
3. B以雙方共享密鑰解密取得明文及雜湊碼
4. B將明文代入相同的雜湊函式運算出雜湊碼
5. 將算出的雜湊碼與接收而來的雜湊碼進行比對

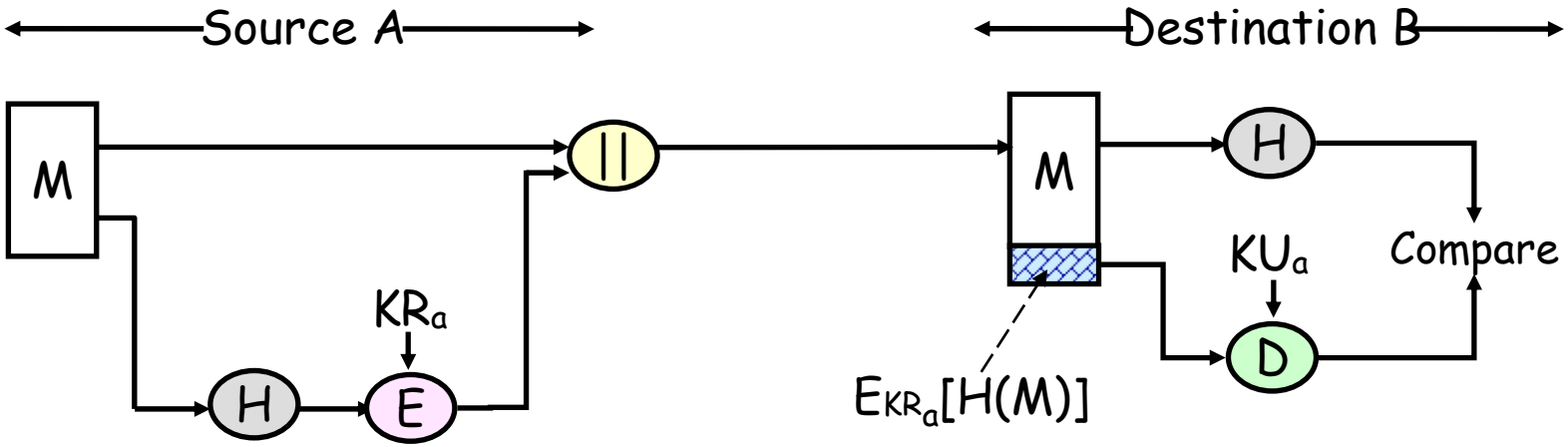
圖2-24(a)雜湊函式的基本運作



資料來源:摘自W. Stalling:
Cryptography and Network Security

1. 明文經由雜湊函式運算得出雜湊碼
2. 將雜湊碼以對稱式加密法加密，並與明文結合後傳送
3. B以雙方共享密鑰對加密的部分解密，取得雜湊碼
4. B將明文代入相同的雜湊函式運算出雜湊碼
5. 將算出的雜湊碼與接收而來的雜湊碼進行比對

圖2-24(b)雜湊函式的基本運作



資料來源:摘自W. Stalling: Cryptography and Network Security

1. 明文經由雜湊函式運算得出雜湊碼
2. 將雜湊碼以**非對稱式加密法**(A的私密金鑰)加密，並與明文結合後傳送
3. B以A的公開金鑰對加密的部分解密，取得雜湊碼
4. B將明文代入相同的雜湊函式運算出雜湊碼
5. 將算出的雜湊碼與接收而來的雜湊碼進行比對

圖2-24(c)雜湊函式的基本運作

雜湊函式的要求

1. 可用在任意長度的訊息 M 上
2. 產生固定長度的輸出 h
3. 對任意的訊息 M 來說，很容易計算出 h=H(M)
4. 給定 h，我們很難求得 x，使得 H(x)=h，稱為單向特性
5. 給定 x，很難求得 y，使得 H(y)=H(x)，稱為弱碰撞抵抗力(weak collision resistance)
6. 很難求出一組 x,y，使得 H(y)=H(x)，稱為強碰撞抵抗力(strong collision resistance)

(三) 簡單的雜湊函式(**)

簡單函式的方案

- 主要是對訊息區段的每一位元作 XOR 運算，此法可表示如下:

  $c_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$，

  其中c_i=雜湊碼的第i的位元

  m ＝輸入的n位元區段總數

  b_{ij} ＝ 第j區段中的第i位元

  $\oplus$ ＝XOR運算

- 以密碼學的安全考量來說，簡單雜湊函式並不安全，可能遭受生日攻擊法，而需要更具威力的密碼學函式 (如SHA-1函式)。

(三) 知名雜湊演算法

- 目前知名且常用的三個雜湊函式為
  - MD5
  - SHA-1
  - RIPEEMD-160
- MD5、SHA-1簡介及雜湊函式的安全性，將於第五章訊息鑑別作詳細介紹。

※未來發展趨勢與研發議題

- 具備數位簽章之數位浮水印技術（Digital Watermarking ）
- 量子密碼學（Quantum Cryptography）

習 題

習題一

- 請說明秘密金匙密碼系統的優點與缺點(相對公開金匙密碼系統)，目前著名的秘密金匙密碼系統為何？

習題二

- 請說明公開金鑰密碼系統的優點與缺點(相對秘密金鑰密碼系統)，目前著名的公開金鑰密碼系統為何？

習題三

- 請簡要的說明DES加密系統的架構。

習題四

- 請簡明介紹DES加密系統中的f函式。

習題五

- 請簡介DES演算法中，S-Box的架構與運作原理。

習題六

- 請說明DES的加密模式標準的CBC模式（Cipher Block Chaining Mode）運作原理。

習題七

- 請詳述DES的三重加密模式(TDES)。

Module 2-6:專案實作(**)

專案目的

- 本專案將以DES、TDES、AES及RSA加密法為主題。
- 建置openssl應用環境。
- 利用實際操作的方式讓同學了解其加密法的概念。

Openssl軟體簡介-1

- Openssl為一開放原始碼的套裝軟體。
- 具有密碼學技術的公開原始碼函式庫。
- 實作出SSL及TLS。
- 支援多種平台(Unix/Linux/Windows 98/ME/2000 /NT/XP等)。
- 利用此套裝軟體以實作了解DES、TDES、AES……等的加密技術概念。

Openssl軟體簡介-2

- 一般來說Openssl都安裝於Unix平台上比較多。
- 若要安裝於Windows平台上，須先安裝ActivePerl套件及C的編譯器。
- 為省略上述煩瑣的安裝步驟，可直接下載編譯過可於Windows執行的Openssl套件。
- 至http://crypto.nknu.edu.tw/textbook/CH01/ 下載Openssl套件，予以解壓縮後即可於命令列模式中執行。

專案摘要

- 各加密法的執行檔放於openssl目錄中的OUT32資料夾底下
- 執行下述指令，以實際操作了解各項加密法的概念
  - openssl des-ecb –e –in xxx.txt –out yyy.out –k password (DES加密)
  - openssl des-ecb –d –in yyy.out –out xxx.txt –k password (DES解密)
  - openssl des-ede3 –e –in xxx.txt –out yyy.out –k password (TDES加密)
  - openssl des-ede3 –d –in yyy.out –out xxx.txt –k password (TDES解密)
  - openssl aes-128-ecb –e –in xxx.txt –out yyy.out –k password (AES加密)
  - openssl aes-128-ecb –d –in yyy.out –out xxx.txt –k password (AES解密)
  - openssl genrsa –out rsa_privatekey.pem –passout pass:password –des3 1024 (產生RSA私密金鑰)
  - openssl rsa –in rsa_privatekey.pem –passin pass:password –pubout –out rsa_publickey.pem (產生對應的RSA公開金鑰)
  - openssl rsautl –encrypt –pubin –inkey rsa_publickey.pem –in xxx.txt –out yyy.txt (利用公開金鑰加密)
  - openssl rsautl –decrypt –inkey rsa_privatekey.pem –in yyy.txt –out xxx.txt (利用私密金鑰解密)

參考文獻

1. Cryptography and Network Security， Third Edition by William Stallings，Pentice Hall.
2. Network Security Essential — Applications and Standards by William Stallings ，Pentice Hall.
3. Network Security: A Beginner's guide ， Second Edition by Eric Maiwald，McGraw-Hill.
4. RSA Laboratories, http://www.rsasecurity.com/rsalabs/
5. Gardner, M. Codes, Ciphers, and Secret Writing. New York: Dover, 1972.
6. Garrett, P. Making, Breaking Codes: An Introduction to Cryptology. Up-per Saddle River, NJ: Prentice Hall, 2001.
7. Kahn, D. The Codebreakers: The Story of Secret Writing. New York:Scribner, 1996.
8. Katzenbeisser, S., ed. Information Hiding Techniques for Steganography and Digital Watermarking. Boston: Artech House, 2000.

參考文獻

9. Konheim, A. Cryptography: A Primer. New York: Wiley, 1981.
10. Korner, T. The Pleasures of Counting. Cambridge, England: Cambridge University Press, 1996.
11. Kumar, I. Cryptology. Laguna Hills, CA: Aegean Park Press, 1997.
12. Nichols, R. Classical Cryptography Course. Laguna Hills, CA: Aegean Park Press, 1996.
13. Nichols, R. ed. ICSA Guide to Cryptography. New York: McGraw-Hill,1999.
14. Singh, S. The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. New York: Anchor Books, 1999.
15. Sinkov, A. Elementary Cryptanalysis: A Mathematical Approach. Washington, DC: The Mathematical Association of America, 1966.
16. Wayner, P. Disappearing Cryptography. Boston: AP Professional Books,1996.

參考文獻

17. Barker, W. Introduction to the Analysis of the Data Encryption Standard (DES). Laguna Hills, CA: Aegean Park Press, 1991.
18. Coppersmith, D. "The Data Encryption Standard (DES) and Its Strength Against Attacks." IBM Journal of Research and Development, May 1994.
19. Electronic Frontier Foundation. Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design. Sebastopol, CA: O'Reilly, 1998
20. Menezes, A.; Oorshcot, P.; and Vanstone, S. Handbook of Applied Cryptography. Boca Raton, FL: CRC Press, 1997.
21. Schneier, B. Applied Cryptography. New York: Wiley, 1996.
22. Simovits, M. The DES: An Extensive Documentation and Evaluation. Laguna Hills, CA Aegean Park Press, 1995.
23. Stinson, D. Cryptography: Theory and Practice. Boca Raton, FL: CRC Press, 2002.
24. Daemen, J., and Rijmen,V. “Rijndael: The Advance Encryption Standard,”Dr.Dobbs Journal,March 2001.

參考文獻

25. Salomaa, A. Public-Key Cryptography. New York: Springer-Veriag, 1996
26. Simmons, G., ed. Contemporary Cryptology: The Science of Information Integrity. Piscataway, NJ: IEEE Press, 1992.
27. Enge, A. Elliptic Curves and Their Applications to Cryptography. Norwell, MA: Kluwer Academic Publishers, 1999.
28. Fernandes, A. "Elliptic Curve Cryptography." Dr. Dobb's Journal, December 1999.
29. Jurisic, A., and Menezes, A. "Elliptic Curves and Cryptography." Dr.Dobb's Journal, April 1997.
30. Koblitz, N. A Course in Number Theory and Cryptography. New York:Springer-Verlag, 1994.
31. Kumanduri, R., and Romero, C. Number Theory with Computer Applications. Upper Saddle River, NJ: Prentice Hall, 1998.
32. M. Robshaw, MD2, MD4, MD5, SHA and Other Hash Functions. RSA Laboratories Technical Report TR-101, July 1995. http://www.rsasecurity.com/rsalabs/index.html
33. Touch, J. "Performance Analysis of MD5." Proceedings, SIGCOMM '95,October 1995.

參考文獻

34. 賴溪松、韓亮、張真誠,近代密碼學及其應用, 松崗,民國八十五年。
35. 黎添壽,吳順裕,資訊與網路安全技術,旗標,民國九十三年十二月。
36 張真誠,林祝興,資訊安全技術與應用, 全華科技,民國九十五年。