

# 訊息鑑別 (Message Authentication)



# 本章內容

- 8.1 前言
- 8.2 單向雜湊函數
- 8.3 文件訊息的完整性驗證
- 8.4 MD5單向雜湊函數
- 8.5 SHA單向雜湊函數
- 8.6 文件訊息的來源驗證
- 8.7 訊息鑑別碼

## 8.1 前言

- 在網路上公開的傳送文件訊息(Document or Message)很容易遭到駭客攔截篡改、新增或刪除等攻擊。
  - 需對文件訊息作**完整性(Integrity)**驗證。
- 該文件訊息是否確實為某人所送過來的文件訊息，而非由他人假冒。
  - 需驗證訊息的**來源是否正確**。

## 8.2 單向雜湊函數

### ◎ 單向雜湊函數二個主要功能

- 單向函數：將文件訊息打散及重組，使其不能再還原為原始文件訊息。
- 雜湊函數：將任意長度的文件訊息壓縮成固定長度的訊息摘要。

### ◎ 數學式子

$$MD=H(M)$$

$H(\cdot)$ ：一單向雜湊函數

$M$ ：表一任意長度文件訊息

$MD$ ：訊息摘要

# 單向雜湊函數特性

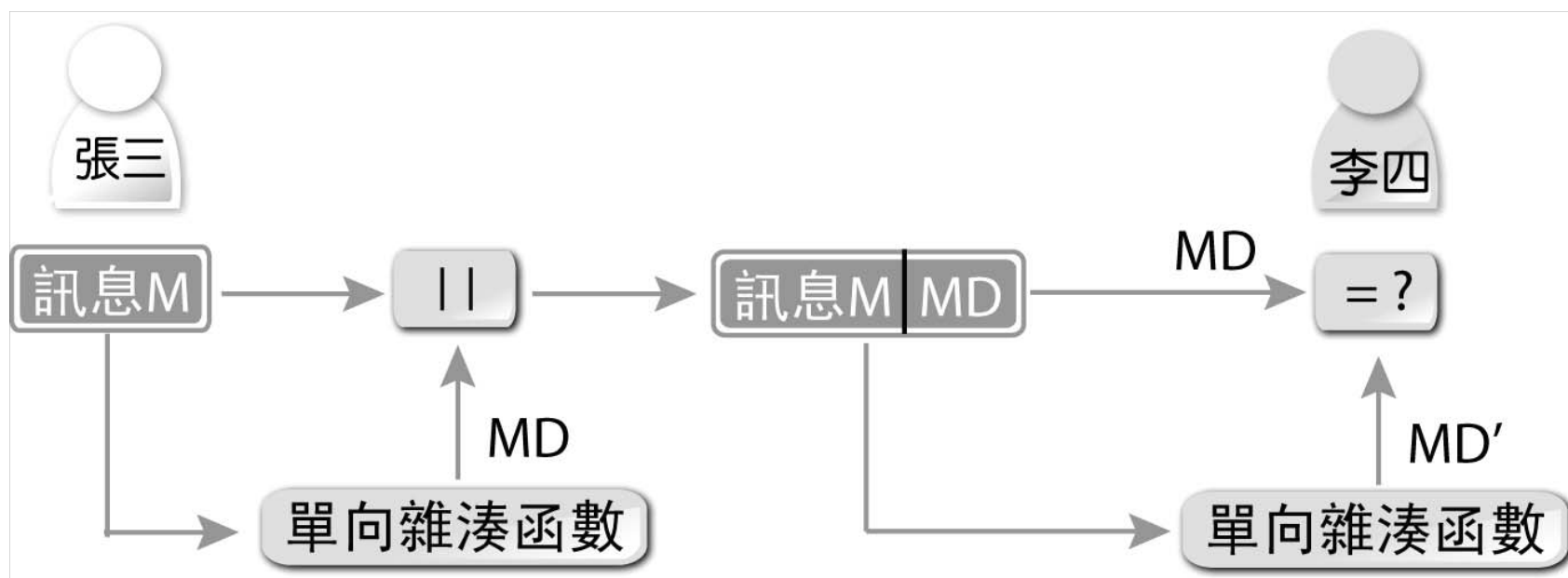
## ◎ 單向雜湊函數三種特性

1. 給定文件訊息 $M$ ，可很容易算出其對應的訊息摘要 $MD$ 。
2. 給定一訊息摘要 $MD$ ，很難從 $MD$ 去找到一個文件訊息 $M'$ ，使 $H(M')=MD$ 。
3. 給定一文件訊息 $M$ ，很難再找到另一文件訊息 $M'$ ，使 $H(M)=H(M')$ 。

避免碰撞的情況發生 (Collision-resistance)

## 8.3 文件訊息的完整性驗證

◎ 以單向雜湊函數做文件訊息的完整驗證，其驗證方式如下：



# 文件訊息完整性驗證

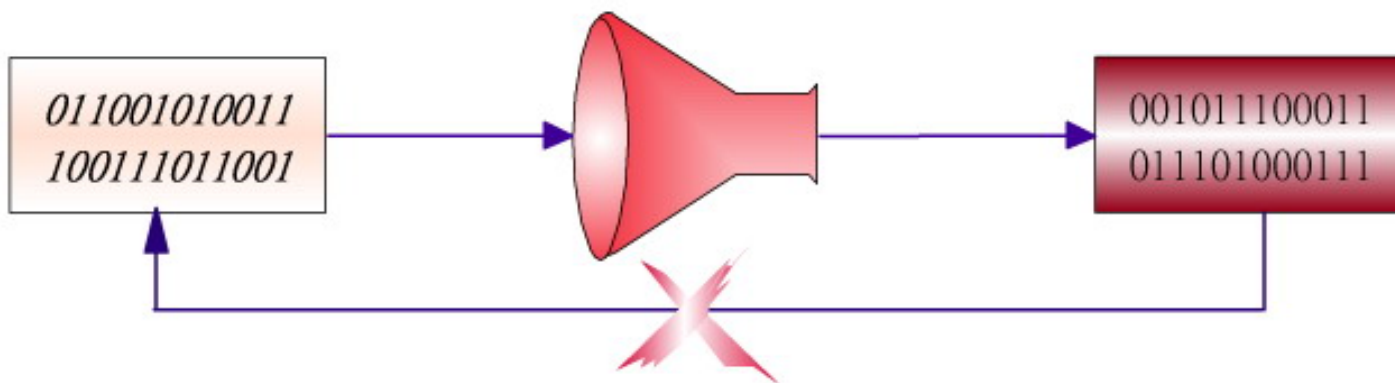
- 首先張三先將要傳送給李四的文件訊息 $M$ ，經單向雜湊函數運算後得到一訊息摘要 $MD$ ，再將文件訊息 $M$ 與訊息摘要 $MD$ 一起送給李四。
- 李四收到後先將文件訊息 $M$ 用同樣的單向雜湊函數運算，假設得到一訊息摘要 $MD'$ ，李四再比較 $MD'$ 是否與收到的 $MD$ 相同，若相同，則李四則可確認此文件之完整性。



## 8.4 MD5單向雜湊函數

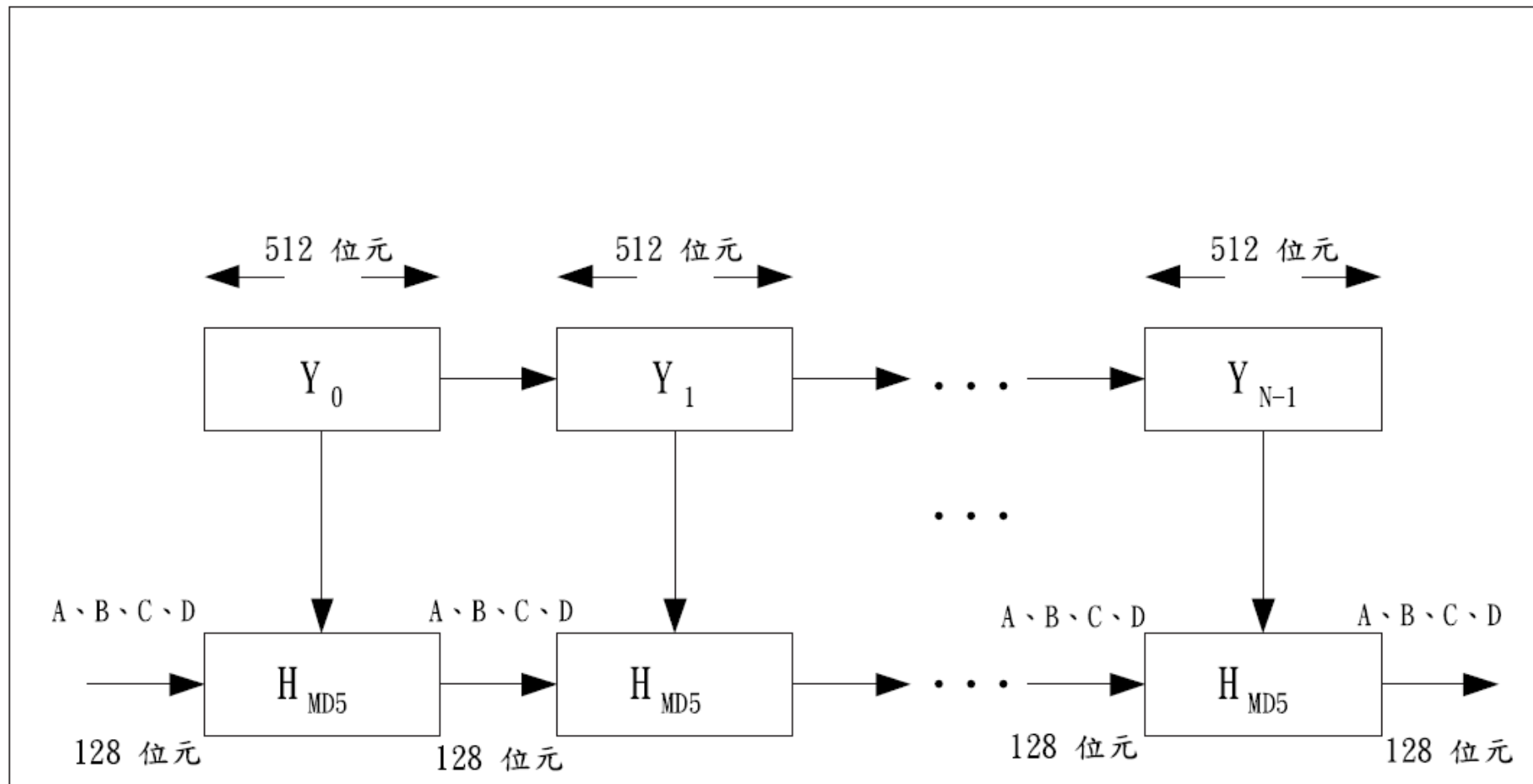
### 簡介

- MD5 was developed by Ron Rivest at MIT
- Arbitrary Length Message  $\rightarrow$  MD5  $\rightarrow$  128-bit Message

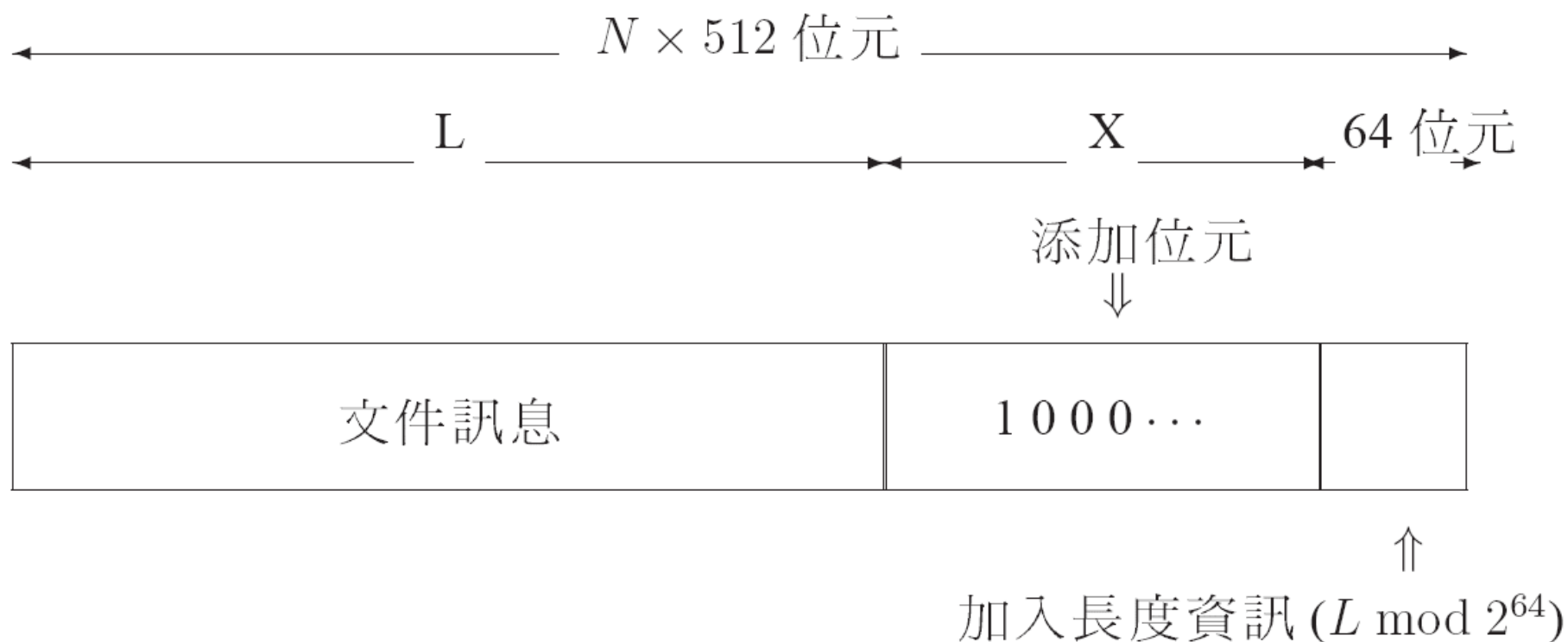




# MD5演算法



# 添加位元及長度資訊



# 文件分割與初始化暫存器

將訊息分割成 $N$ 個512位元的小區塊， $Y_0$ 、 $Y_1$ 、...、 $Y_{N-1}$ ，其總長度等於 $N \times 512$ 位元。將分割後的512位元訊息區塊，再分割為十六個32位元的字元，以 $M[0]$ 、 $M[1]$ 、...、 $M[15]$ 來表示這16個32位元的字元。

## Initial Vector, IV

Note: hexadecimal values

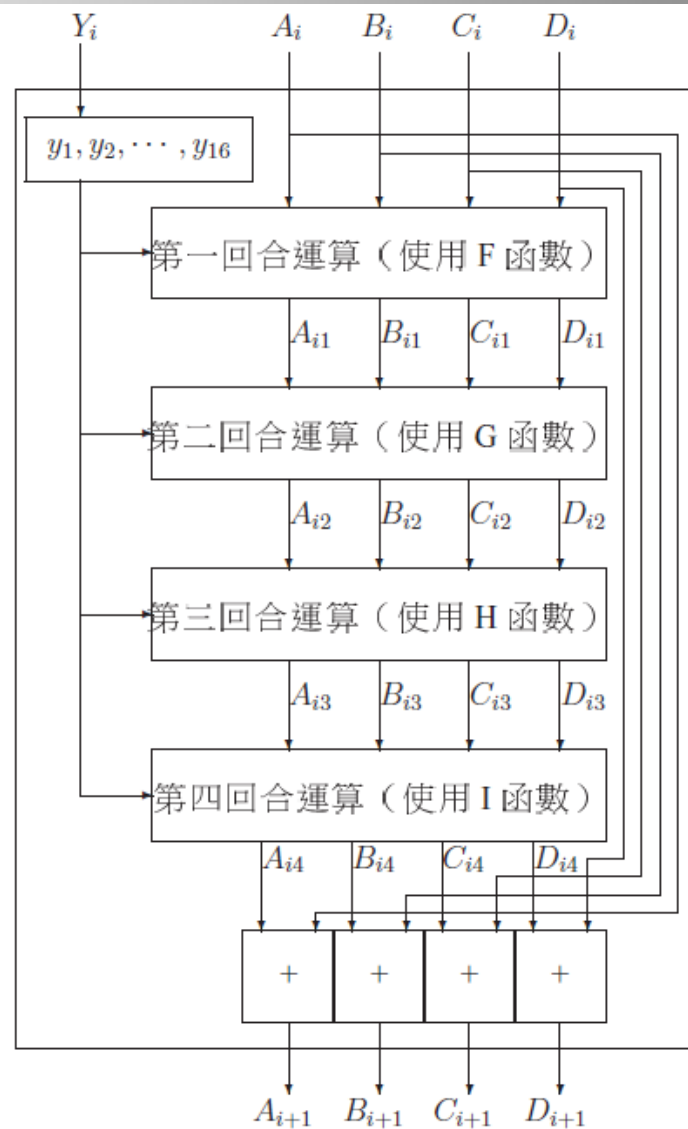
*A*: 01 23 45 67

*B*: 89 AB CD EF

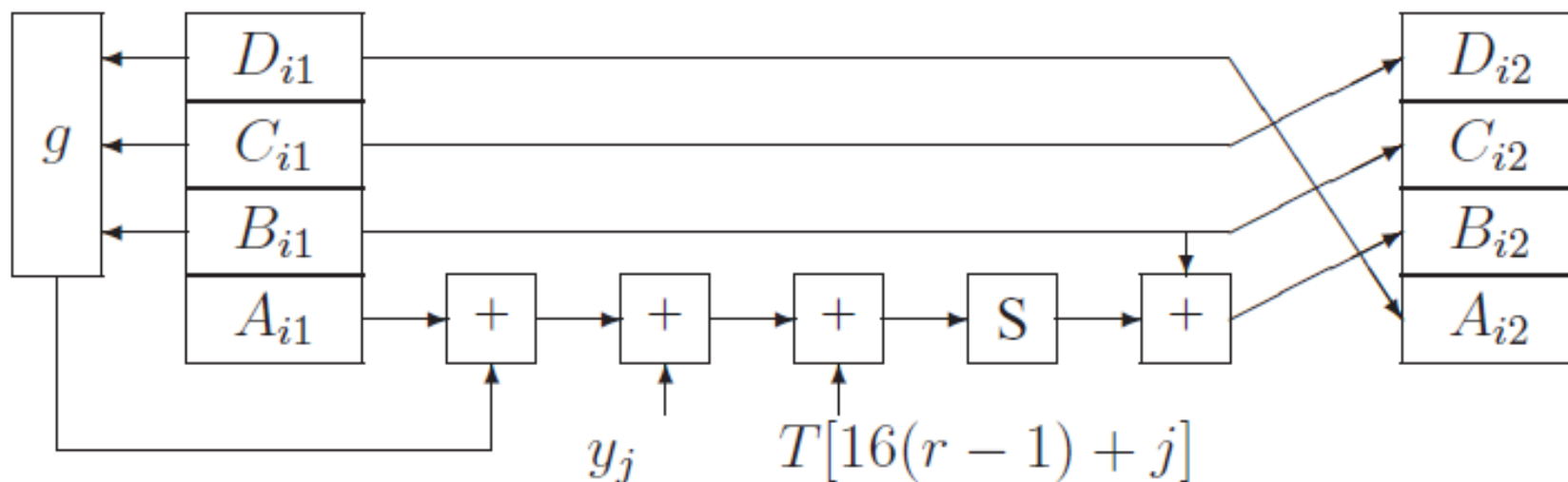
*C*: FE DC BA 98

*D*: 76 54 32 10

# 單一訊息區塊的運算過程



# $H_{MD5}$ 之每一回合運算過程



$$A \leftarrow d$$

$$B \leftarrow ((a + g(b, c, d) + M[K] + T[16(r-1) + K]) \lll S) + b$$

$$C \leftarrow b$$

$$D \leftarrow c$$

# MD5的非線性計算方式及其真值表

| 回合 | 基本函數 ( $g$ ) | 定義                                    |
|----|--------------|---------------------------------------|
| 1  | $F(b, c, d)$ | $(b \wedge c) \vee (\neg b \wedge d)$ |
| 2  | $G(b, c, d)$ | $(b \wedge d) \vee (c \wedge \neg d)$ |
| 3  | $H(b, c, d)$ | $b \oplus c \oplus d$                 |
| 4  | $I(b, c, d)$ | $c \oplus (b \vee \neg d)$            |

| b | c | d | F | G | H | I |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

# MD5的 $T[i]$ 參數值

|                |                |                |                |
|----------------|----------------|----------------|----------------|
| T[1]=d76aa478  | T[2]=e8c7b756  | T[3]=242070db  | T[4]=c1bdceee  |
| T[5]=f57c0faf  | T[6]=4787c62a  | T[7]=a8304613  | T[8]=fd469501  |
| T[9]=698098d8  | T[10]=8b44f7af | T[11]=ffff5bb1 | T[12]=895cd7be |
| T[13]=6b901122 | T[14]=fd987193 | T[15]=a679438e | T[16]=49b40821 |
| T[17]=f61e2562 | T[18]=c040b340 | T[19]=265e5a51 | T[20]=e9b6c7aa |
| T[21]=d62f105d | T[22]=02441453 | T[23]=d8a1e681 | T[24]=e7d3fbc8 |
| T[25]=21e1cde6 | T[26]=c33707d6 | T[27]=f4d50d87 | T[28]=455a14ed |
| T[29]=a9e3e905 | T[30]=fcefa3f8 | T[31]=676f02d9 | T[32]=8d2a4c8a |
| T[33]=fffa3942 | T[34]=8771f681 | T[35]=6d9d6122 | T[36]=fde5380c |
| T[37]=a4beea44 | T[38]=4bdecfa9 | T[39]=f6bb4b60 | T[40]=bebfbc70 |
| T[41]=289b7ec6 | T[42]=eaa127fa | T[43]=d4ef3085 | T[44]=04881d05 |
| T[45]=d9d4d039 | T[46]=e6db99e5 | T[47]=1fa27cf8 | T[48]=c4ac5665 |
| T[49]=f4292244 | T[50]=432aff97 | T[51]=ab9423a7 | T[52]=fc93a039 |
| T[53]=655b59c3 | T[54]=8f0ccc92 | T[55]=ffeff47d | T[56]=85845dd1 |
| T[57]=6fa87e4f | T[58]=fe2ce6e0 | T[59]=a3014314 | T[60]=4e0811a1 |
| T[61]=f7537e82 | T[62]=bd3af235 | T[63]=2ad7d2bb | T[64]=eb86d391 |

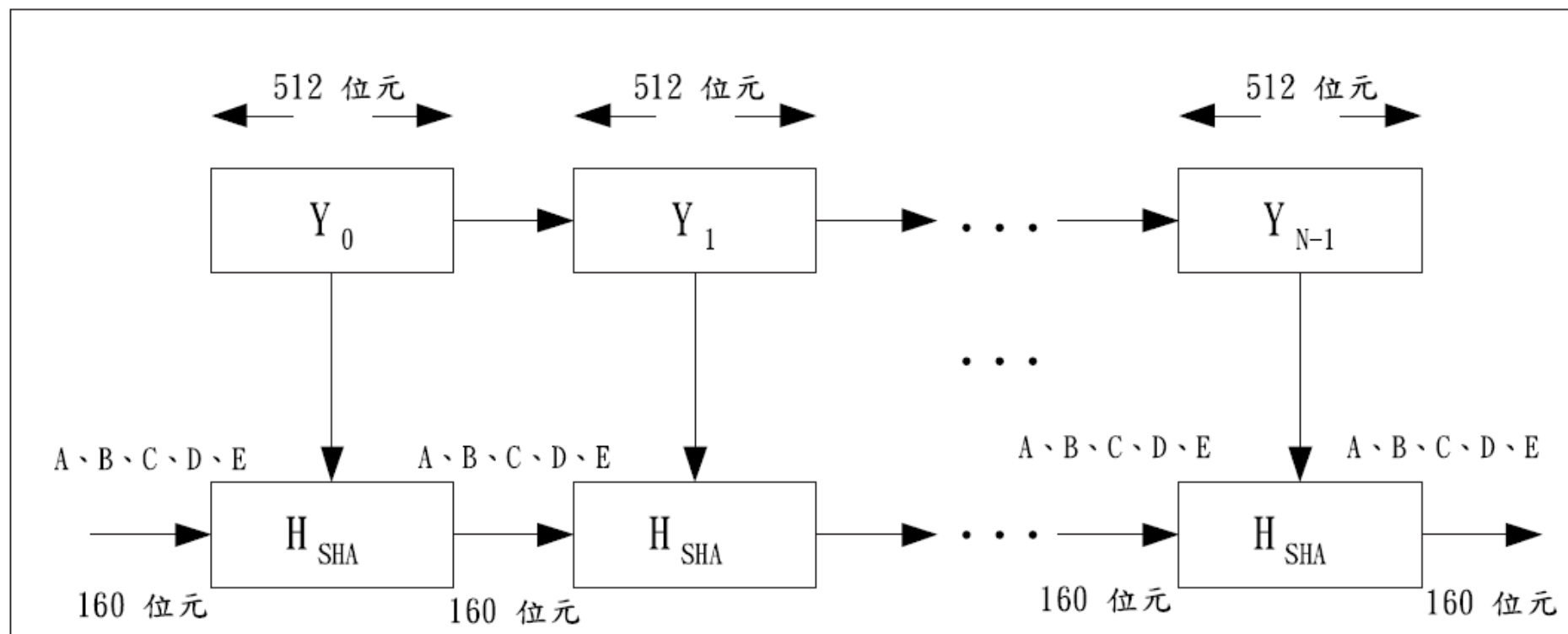


## 8.5 SHA單向雜湊函數

### 簡 介

- SHA 為美國國家標準局(NIST)為了確保數位簽章演算法的安全性所提出。
- SHA 可以將一個任意長度的文件訊息資料，壓縮成只有160位元的固定長度。
- Maximum length of less than  $2^{64}$  Message → SHA → 160-bit Message

# SHA的運算架構



# 初始化MD暫存器

## 五個暫存器的初始向量

*A*: 67 45 23 01

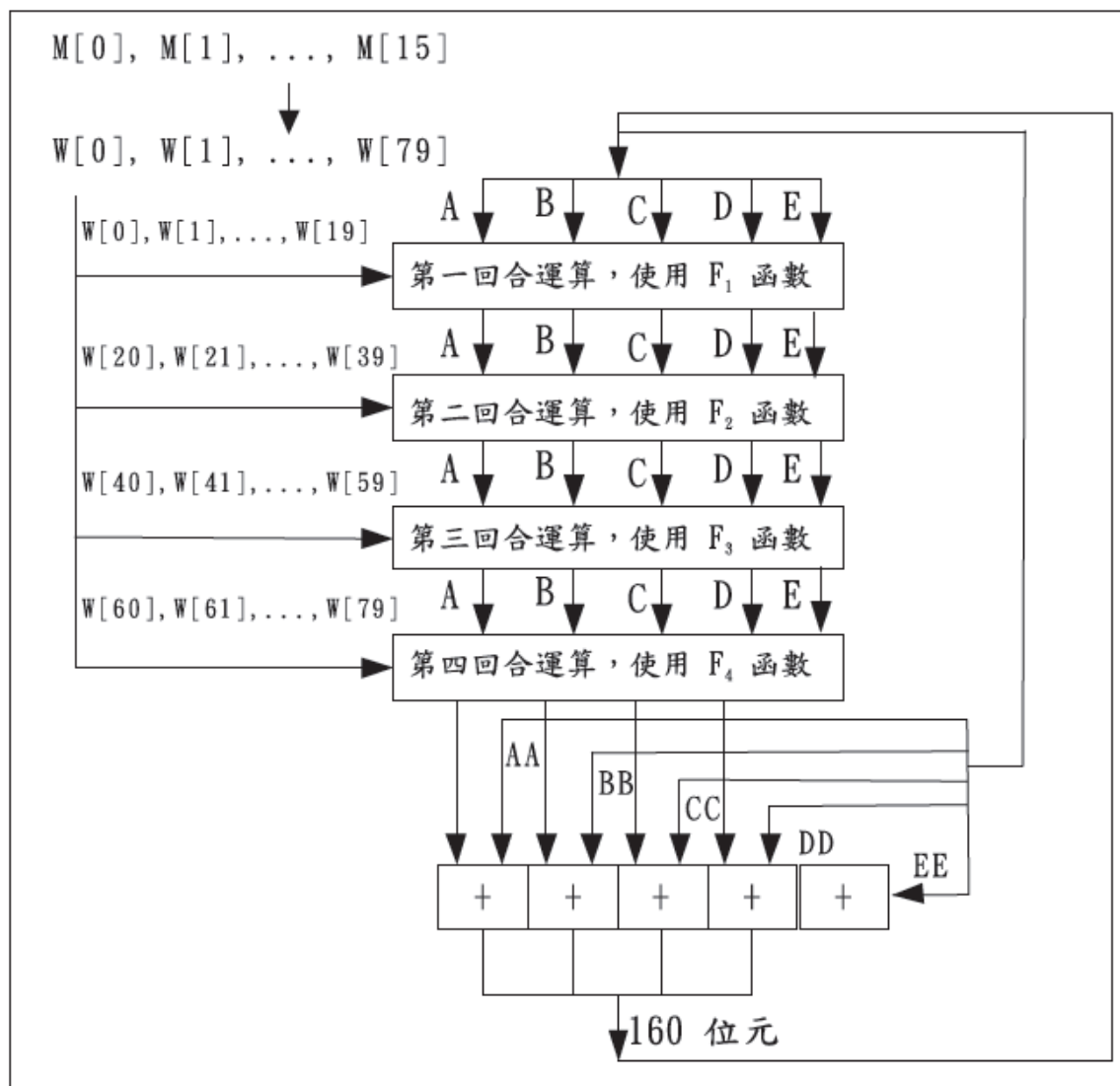
*B*: EF CD AB 89

*C*: 98 BA DC FE

*D*: C3 D2 E1 F0

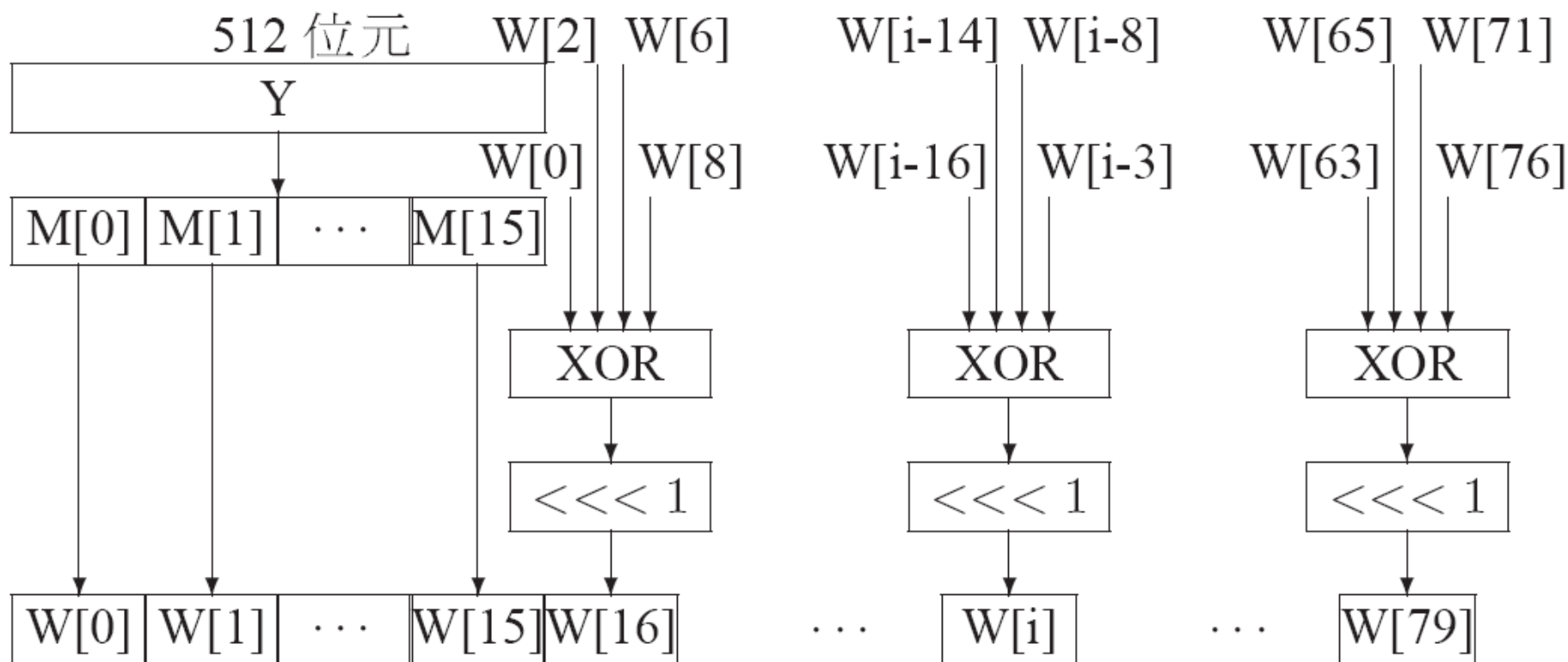
*E*: C3 D2 E1 F0

# SHA訊息區塊的運算過程

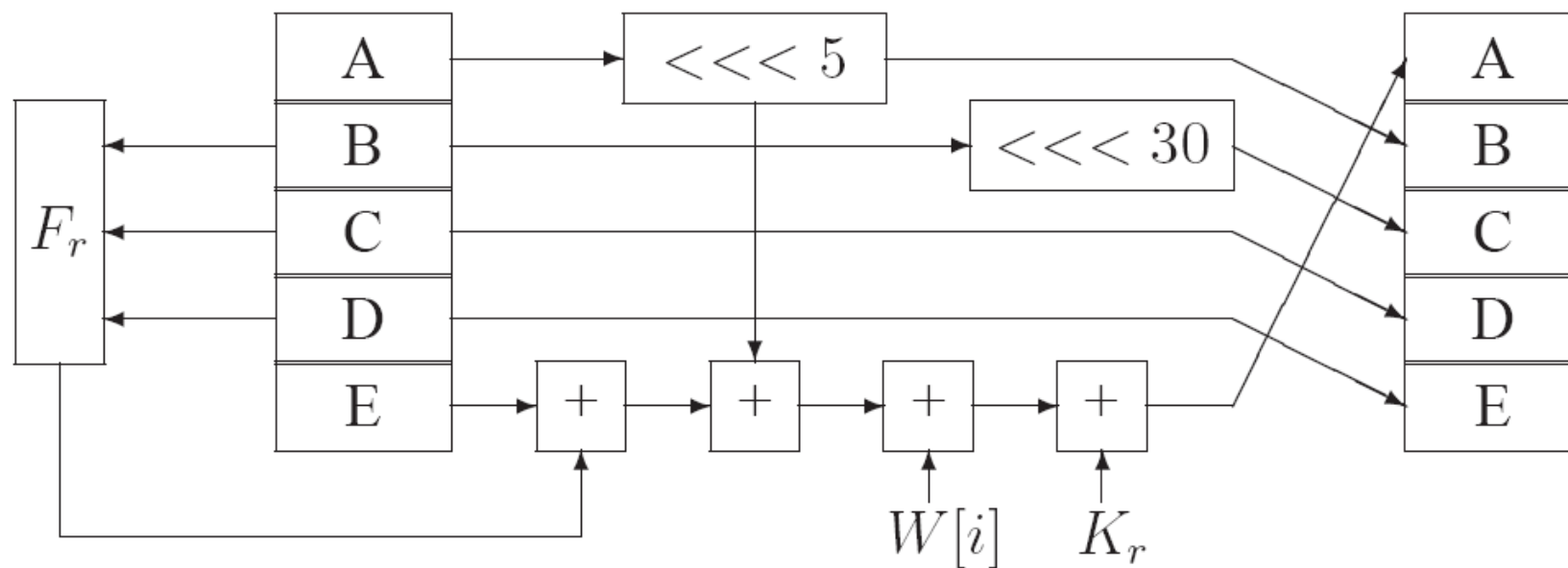


# SHA 中 W 字元的產生過程

$$\begin{cases} W[i] = M[i], & \text{當 } 0 \leq i \leq 15 \\ W[i] = (W[i-3] \oplus W[i-8] \oplus W[i-14] \oplus W[i-16]) \lll 1 & \text{當 } 16 \leq i \leq 79 \end{cases}$$



# SHA的回合運算過程



$$A \leftarrow e + F_r(b, c, d) + (A \lll 5) + W[i] + K_r$$

$$B \leftarrow (A \lll 5)$$

$$C \leftarrow (B \lll 30)$$

$$D \leftarrow c$$

$$E \leftarrow d$$

# SHA的非線性計算方式

| 回合 | 函數 $F_r$       | 定義   |
|----|----------------|--|
| 1  | $F_1(b, c, d)$ | $(b \wedge c) \vee (\neg b \wedge d)$              |
| 2  | $F_2(b, c, d)$ | $b \oplus c \oplus d$                              |
| 3  | $F_3(b, c, d)$ | $(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$ |
| 4  | $F_4(b, c, d)$ | $b \oplus c \oplus d$                              |



SHA的  $F_1$ 、 $F_2$ 、 $F_3$ 、 $F_4$  函數真值表

| $b$ | $c$ | $d$ | $F_1$ | $F_2$ | $F_3$ | $F_4$ |
|-----|-----|-----|-------|-------|-------|-------|
| 0   | 0   | 0   | 0     | 0     | 0     | 0     |
| 0   | 0   | 1   | 1     | 1     | 0     | 1     |
| 0   | 1   | 0   | 0     | 1     | 0     | 1     |
| 0   | 1   | 1   | 1     | 0     | 1     | 0     |
| 1   | 0   | 0   | 0     | 1     | 0     | 1     |
| 1   | 0   | 1   | 0     | 0     | 1     | 0     |
| 1   | 1   | 0   | 1     | 0     | 1     | 0     |
| 1   | 1   | 1   | 1     | 1     | 1     | 1     |

# SHA的常數值 $K_r$

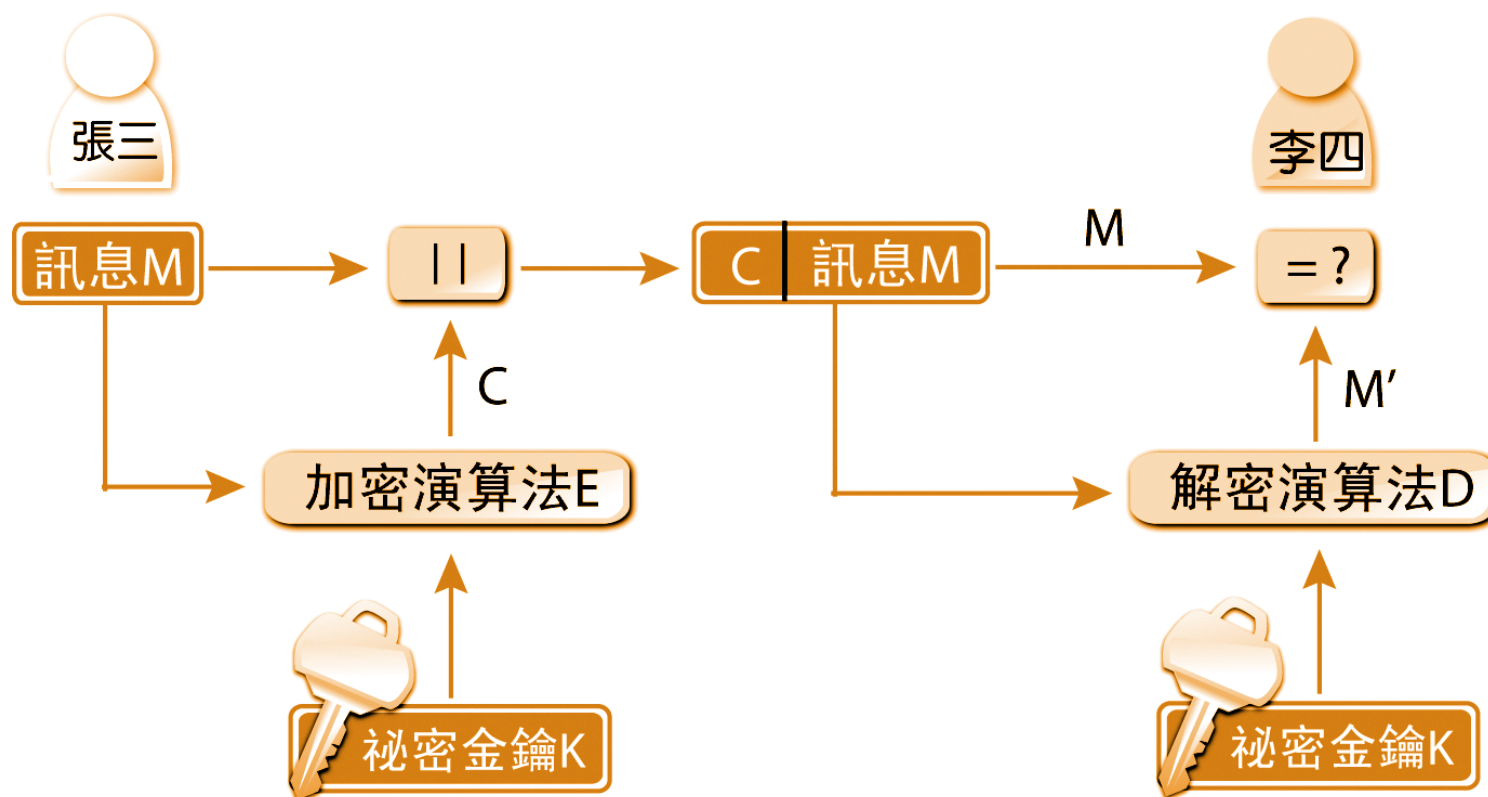
| 回合 | 常數值              |
|----|------------------|
| 1  | $K_1 = 5A827999$ |
| 2  | $K_2 = 6ED9EBA1$ |
| 3  | $K_3 = 8F1BBCDC$ |
| 4  | $K_4 = CA62C1D6$ |

## 8.6 文件訊息的來源驗證

- 單向雜湊函數雖然可用來作訊息的完整性驗證但實際運用上卻是行不通的。
- 問題在於沒有對訊息的來源作鑑別(Authentication)。

## 8.6.1 秘密金鑰密碼系統的訊息鑑別機制

◎ 對稱式密碼系統文件訊息鑑別機制的鑑別式如下：

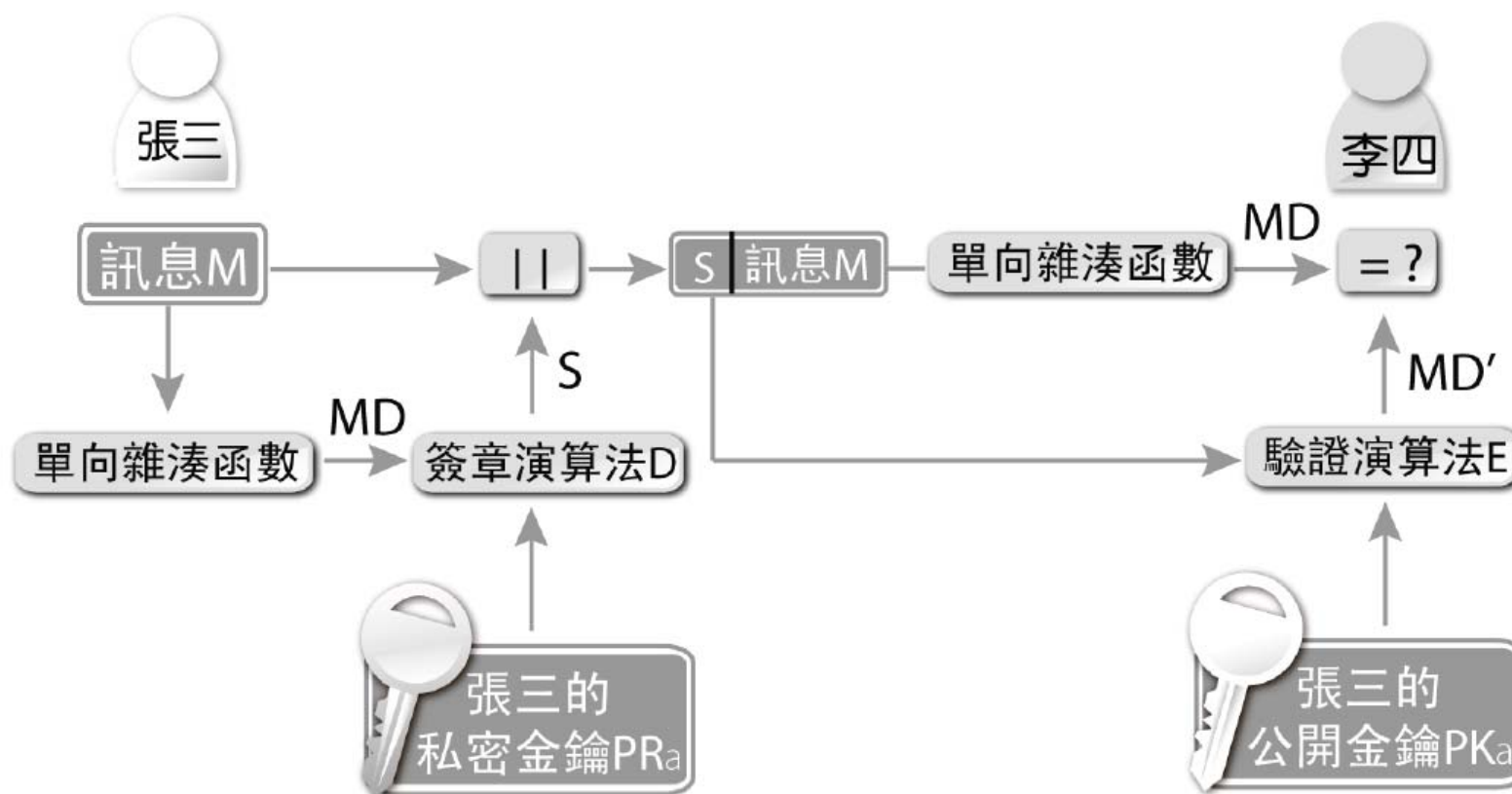


# 秘密金鑰密碼系統的訊息鑑別機制

- 當張三要送訊息 $M$ 給李四，張三就用與李四先協議好的秘密金鑰 $K$ ，來對訊息 $M$ 作加密(如 $C=E_k(M)$ )， $E(.)$ 為一對稱式的加密演算法， $C$ 為加密後所得之密文。
- 張三將訊息 $M$ 及 $C$ 一同寄給李四，李四收到後就驗證 $D_k(C) \stackrel{?}{=} M$ ，其中 $D(.)$ 為一對稱式的解密演算法。若 $D_k(C)$ 與 $M$ 相等，則李四就可確認此訊息為張三所送。

## 8.6.2 公開金鑰密碼系統的訊息鑑別機制

◎ 以公開金鑰密碼系統來做數位簽章，其文件訊息鑑別方式如下：



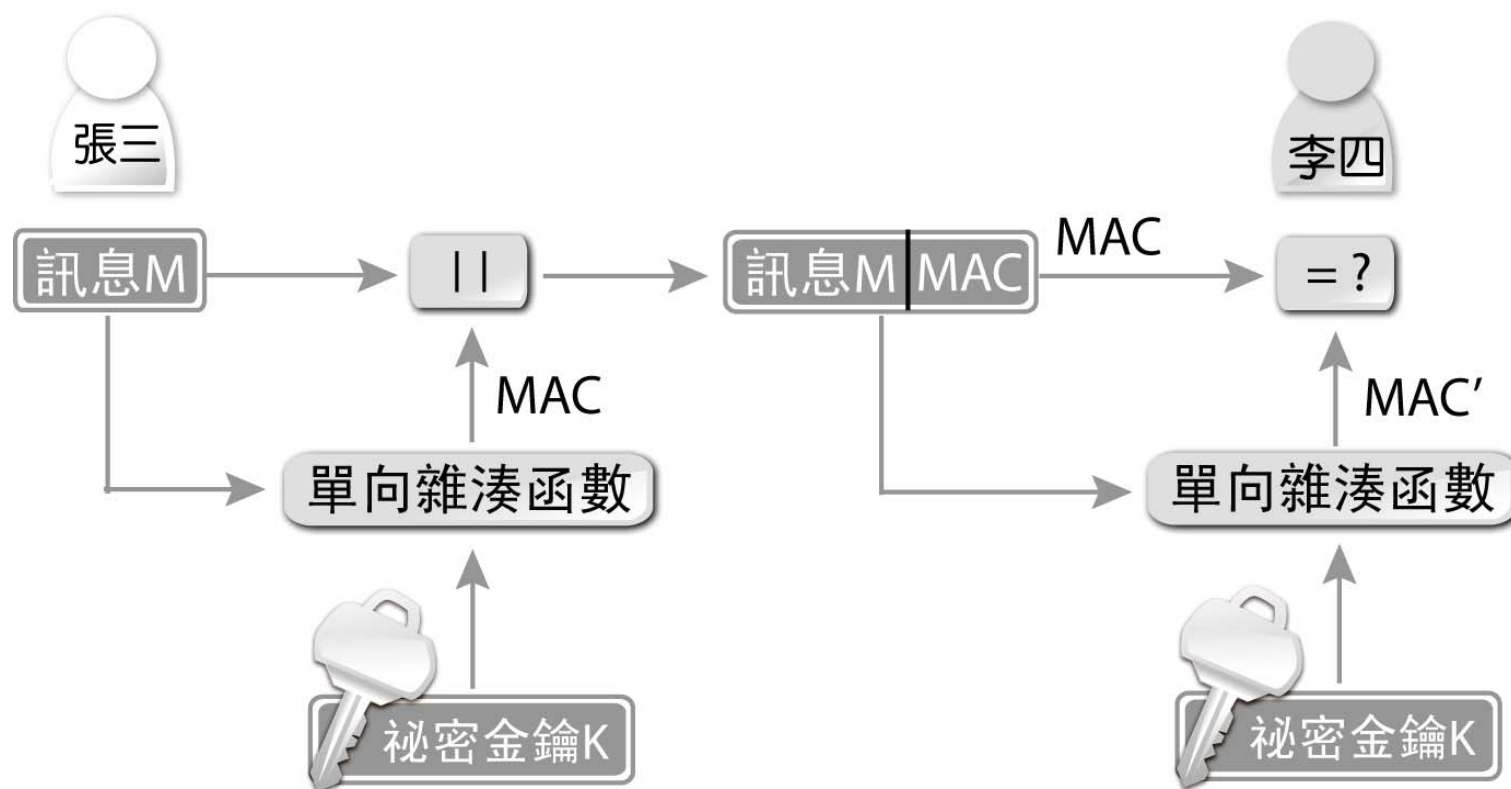
# 公開金鑰密碼系統的訊息鑑別機制

- 假設張三的公開金鑰為 $PK_a$ ，私密金鑰為 $PR_a$ 。
- 要傳送訊息 $M$ 給李四時，張三計算出該訊息的訊息摘要 $MD=H(M)$ ，接著再用私密金鑰計算訊息摘要的數位簽章 $S=D_{PR_a}(MD)$ ，其中 $D(.)$ 為一公開金鑰密碼系統的簽章演算法。然後將文件訊息 $M$ 與數位簽章一起送給李四。
- 李四收到後，先計算出訊息 $M$ 的訊息摘要 $MD=H(M)$ ，接著利用張三的公開金鑰 $PK_a$ 來驗證 $E_{PR_a}(S)?=MD$ 。若相等，則李四就可信該訊息是正確的。



## 8.6.3 金鑰相依單向雜湊函數的訊息鑑別機制

- ◎ 為改善前面兩種方法需大量運算的缺點，便有了植基於金鑰相依單向雜湊函數，此種機制又稱文件訊息鑑別碼(MAC)：



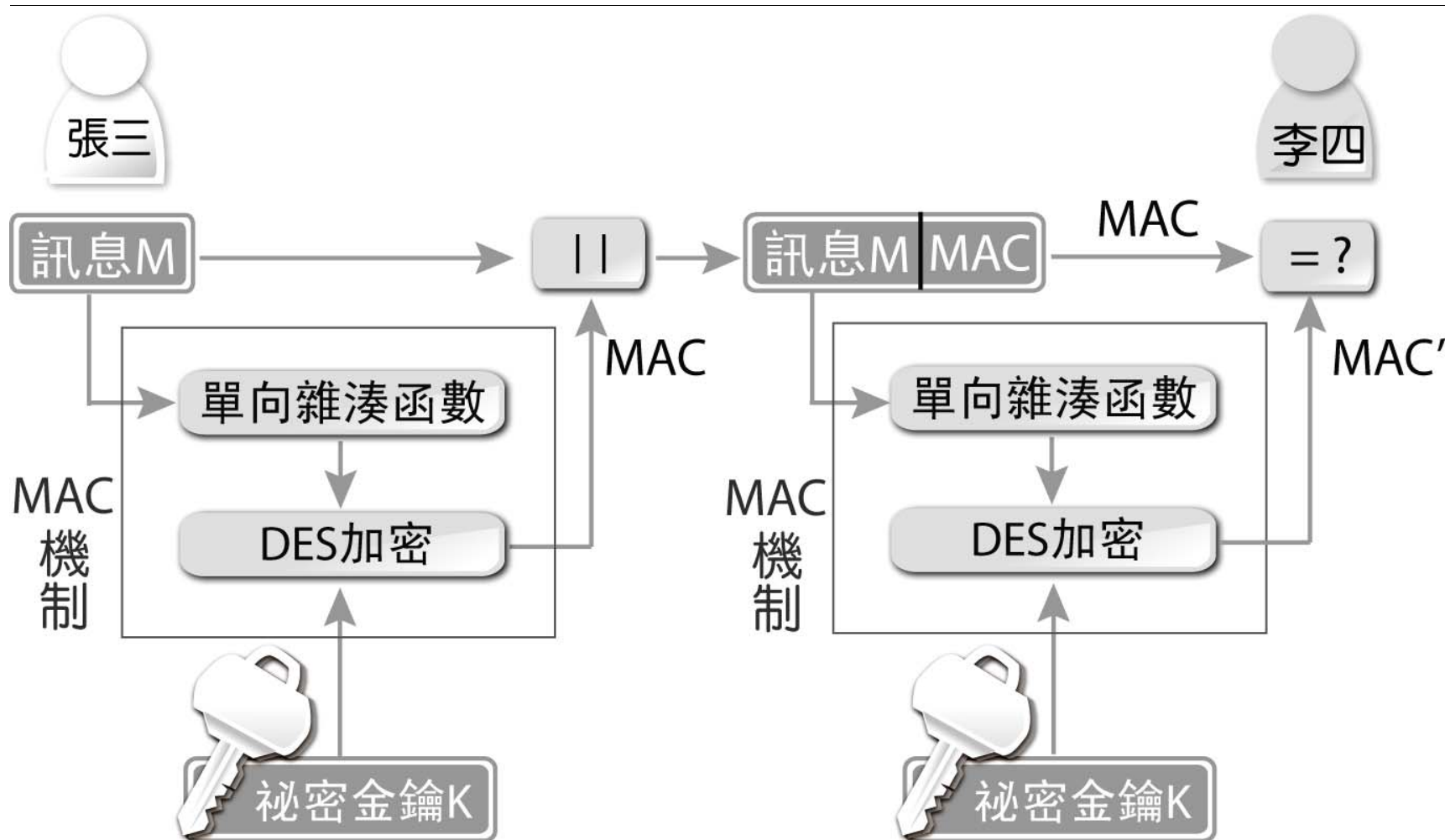
# 金鑰相依單向雜湊函數的訊息驗證機制

- 張三與李四先協議一把共同的祕密金鑰 $K$ ，當張三要傳送一訊息 $M$ 給李四時，張三會先利用 $K$ 對此文件訊息 $M$ 產生一文件訊息驗證碼 $MAC = H(K||M)$ 。然後將 $M$ 連同 $MAC$ 一起送給李四。
- 李四收到後會先將 $M$ 及自己的祕密金鑰 $K$ ，來算出 $MAC' = H(K||M)$ ，再與接收到的 $MAC$ 進行比對，若相等，則確信文件訊息在傳送過程中沒有遭到篡改。

## 8.7 訊息鑑別碼

- ◎ 文件訊息鑑別碼(Message Authentication Code, MAC)可用來驗證文件訊息是否為約定好通訊的雙方所傳送，並可驗證文件訊息在傳遞過程中是否遭到篡改。

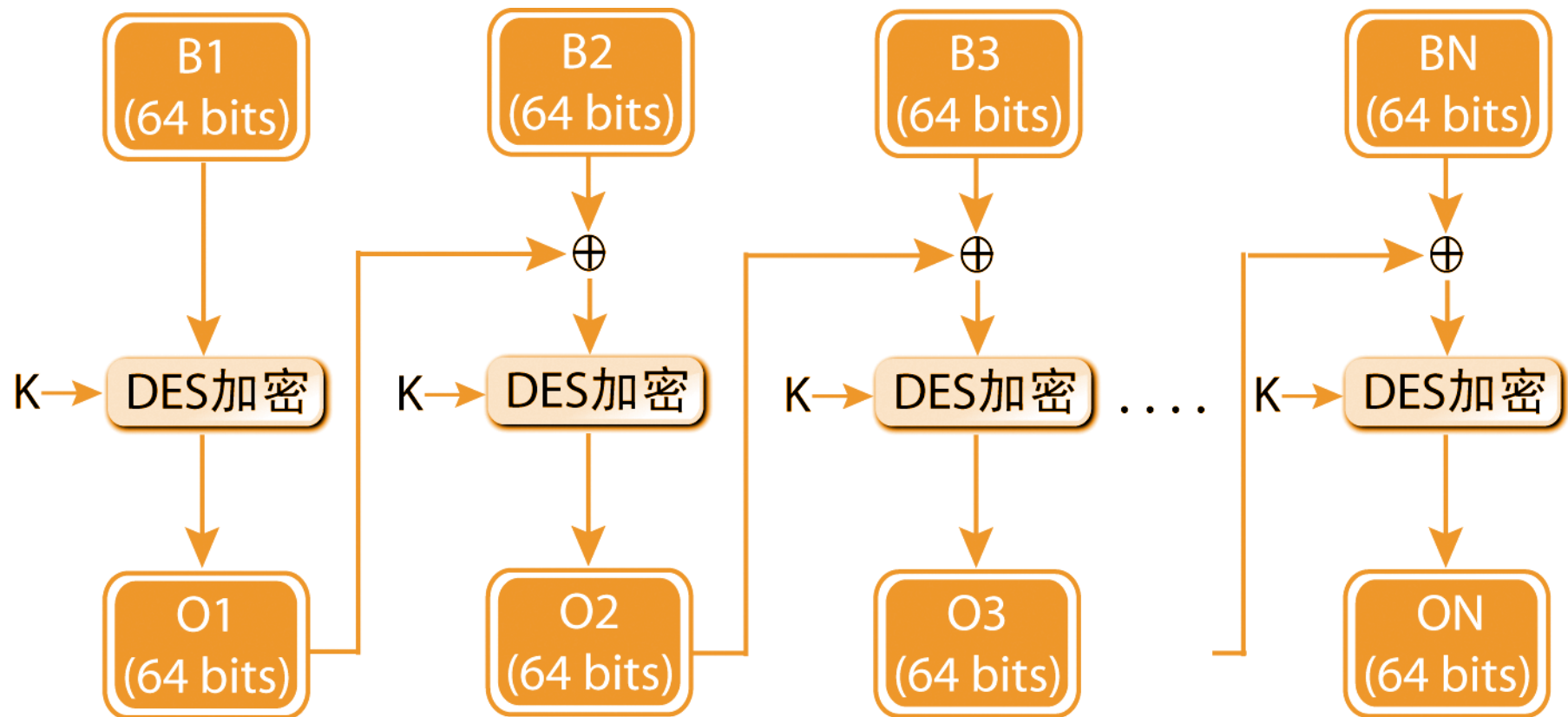
## 利用秘密金鑰密碼系統及單向雜湊函數所構成的MAC架構圖



## 8.7.1 CBC-MAC

- ◎ CBC-MAC是一種用CBC區塊加密模式來實作文件訊息鑑別的做法，只需將文件訊息用CBC模式來加密即可。
- ◎ MAC的產生過程有三個步驟：
  - (1)將所要傳遞文件訊息長度擴充為64位元的位數，不足以0補上。
  - (2)將此擴充文件訊息依每64位元切割成一個區塊。
  - (3)先對第一個區塊作DES的加密，加密時需輸入一56位元的祕密金鑰，所得到的64位元的密文，再與下一個文件訊息區塊作互斥或(XOR)運算，然後再用DES對其加密，以此類推。

# CBC-MAC



## 8.7.2 單向雜湊函數MAC

- ◎ 這種MAC類型是利用一單向雜湊函數，配合一祕密金鑰所構成的訊息鑑別碼。
- ◎ 此類的文件訊息鑑別碼機制也可讓使用者自行來決定要採用何種單向雜湊函數，在實作上相當便利也具有彈性。
- ◎ 串接方式可以是 $H(K||M)$ ，但不安全。較安全的串接方式是 $H(M||K)$ 、 $H(K_1||M||K_2)$ 、 $H(K, H(K||M))$ 、 $H(K_1, H(K_2||M))$ 。



# 以 $H(K||M)$ 串接方式不安全

- 可任加一個區塊的文件訊息 $M'$ 到原文件訊息的後面其驗證的結果都會正確。
- 以MD5單向雜湊函數演算法為例，若王五攔截到張三要傳給李四的文件訊息 $M$ 及其訊息鑑別碼 $H(K||M)$ ，那麼王五根本不需要知道張三跟李四所協議的共同秘密金鑰 $K$ 就可以任加一段訊息 $M'$ 至原訊息後面，且有辦法得到正確的訊息鑑別碼 $H(K||M||M')$ 。
  1. 利用攔截到的128位元訊息鑑別碼 $H(K||M)$ 作為MD5中 $A$ 、 $B$ 、 $C$ 、 $D$ 四個暫存器的初始值。
  2. 將添加訊息 $M'$ 分割成數個512位元的訊息區塊，再透過MD5演算法來做運算。
  3. 最後得到的128位元輸出結果就等於 $H(K||M||M')$ 。

SHA也有同樣的問題