第六章　訊息確認

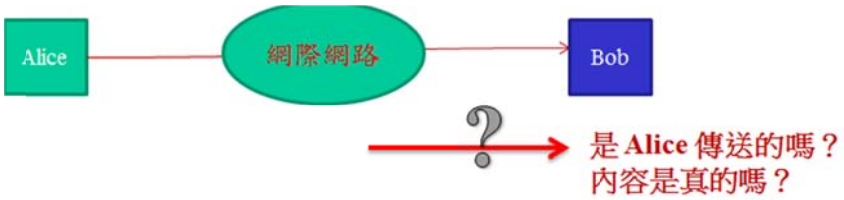
簽收快遞包裹之後，如何來判斷它不是來自恐怖份子的炸彈包裹？或裡面的巧克力是否已被盜取了一大半？

6-1 訊息確認簡介

當春嬌由網路收到一份由志明傳來的訊息(或郵件)時，春嬌率先考慮到的問題是：

(1) 該訊息（或郵件）在傳遞當中是否遭受他人竄改。

(2) 該訊息果真是由志明所發送或是他人冒名發送的？



解決上述第一個問題，僅需仰賴『完整性檢查』(Integrity Check) 即可。至於要完全解決第一與第二個問題，則需要良好的『訊息確認』(Message Authentication) 機制不可。換言之，欲保障所收到的訊息無誤的話，可區分為完整性檢查與訊息確認等兩個層次，至於決定採用何種機制視通訊環境需求而定。基本上，『訊息確認』與『數位簽章』(Digital Signature)，都是被設計來解決上述兩大問題的技術。且兩者都需要使用鑰匙來表明身分，不過對鑰匙的來源，之間有很大的差異。一般來講，『訊息確認』是以雙方共享的秘密鑰匙（或會議鑰匙）來確認傳送者的身份；『數位簽章』大多是以傳送者的公開鑰匙來確定身份。至於兩者的應用範圍也有很大的區隔，一般『訊息確認』較著重於訊息傳輸方面；『數位簽章』則大多使用於身份識別方面（如 PKI 系統），但爾後的應用發展，實難預料。本章以訊息確認為主要介紹對象，至於數位簽章部分留於第七章再詳細介紹。

6-2 完整性檢查值 - ICV

所謂『完整性檢查值』(Integrity Check Value, ICV) 是利用雜湊函數（如 MD4、MD5、SHA-1）而得。通常傳送端發送訊息之前，會先將訊息經過某一個雜湊函數計算

出雜湊碼，再將此雜湊值附加在訊息的後面，一併傳送給接收端；接收端收到訊息之後，也以相同的雜湊函數計算出雜湊碼，如果所計算出的結果與訊息後面的雜湊值相同的話，即表示訊息在傳輸當中未遭受它人竄改。由於此雜湊碼為檢查訊息是否遭受竄改的依據，因此稱為『完整性檢查值』。

基本上，完整性檢查不會使用到鑰匙，並沒有利用到加密/解密機制，其安全性完全仰賴雜湊函數的強度而定。但就安全性通訊而言，強度再強的雜湊演算法都不能保證其絕對可靠性，因此，完整性檢查值大多在有保護的安全連線下傳輸。為配合各種安全連線的傳輸，ICV 可分下列兩種製作方式。

6-2-1 明文計算 ICV

明文計算 ICV 機制（如圖 6-1 (a) 所示），表示利用某一種雜湊函數直接將明文計算出一個 ICV 值，再將 ICV 附加在訊息後面傳送給對方；接收端也將所收到的訊息(M') 經過同一種雜湊函數計算出另一個 ICV 值，如果兩個 ICV 值相同的話，則表示訊息是正確的。在此機制下，ICV 值與訊息都是以明文傳送，並沒有任何安全考量；攻擊者取得訊息之後，可試著竄改訊息內容以得到相同的 ICV 值，如此一來，便可達到破壞攻擊的目的。至於攻擊者是否可偽造新的訊息並產生相同的 ICV 值的困難度，這完全視雜湊演算法的強度如何（第五章介紹）。由此可見，明文計算 ICV 機制僅適合於有安全保護下的連線使用；不然就得將訊息與 ICV 值一起加密後，再傳送給對方。

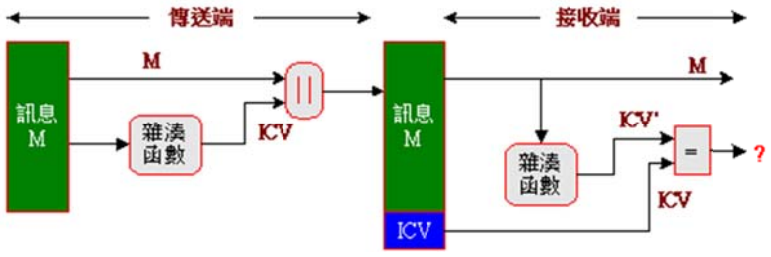


圖 6-1-1 完整性檢查製作方式

6-2-2 密文計算 ICV

密文計算 ICV 機制，表示訊息經過加密之後，再利用密文來計算 ICV 值，並將 ICV 值與密文一起傳送給接收端，如圖 6-1 (b) 所示。很明顯可看出，它的安全性會比明文計算高一點，攻擊者若想藉由修飾沒有規律的密文，產生相同的 ICV 值，似乎不是

那麼容易，因此可以獨立作業，不一定非得在有安全保護的連線下傳輸。一般對比較不重要的訊息，利用此機制來傳輸即可（反正訊息已加密），不需要額外的安全連線。

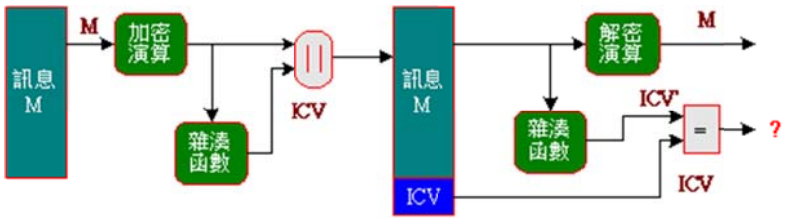


圖 6-1-2 完整性檢查製作方式

6-3 訊息確認碼 - MAC

『訊息確認』(Message Authentication)必須包含兩層次的功能：一者為保證訊息的『完整性』；另一者為如何將秘密鑰匙加入訊息之內（如加密功能），以作為發送訊息者的身份確認，來達到『確認性』功能。就前者而言，傳送端一般都利用『雜湊函數』(Hash Function)計算出一個固定長度的雜湊值，作為一個獨一無二的訊息認證；並且該值必須隨輸入訊息的改變而改變，因此無法冒充，故可稱之為『完整性檢查碼』(ICV)。後者係將秘密鑰匙加入雜湊值(或 ICV)之中，成為一個可確認身份的『訊息確認碼』(Message Authentication Code, MAC)。當然，將鑰匙加入訊息之中，大多是採用加密演算法。但也不完全如此，因為將鑰匙加入雜湊值（或 ICV）的目的是要確定傳送者身份，並非要保護該雜湊值，因此，除了訊息確認碼較普遍的加密演算法（MAC-DES）之外，我們也會介紹其它相關的演算法（如 HMAC，6-5 節介紹）。

一般情況，MAC 機制大多採用秘密鑰匙系統（有些系統會採用公開鑰匙系統）。通訊之前雙方必須透過其它機制，使能共享一把秘密鑰匙。接收端則依據對方所使用的鑰匙是否和自己的相同，來確認訊息是否來自可信任的一方；因此，將共享鑰匙植入雜湊碼內，同時完成『完整性』與『確認性』的功能。圖 6-2 為訊息確認的功能圖，運作程序如下：首先將訊息經由雜湊演算法計算出雜湊值（H），再利用鑰匙加密（或植入）之後，得到一個稱之為『訊息確認碼』(MAC)；接著將它附加在訊息的後面，傳送給接收端；接收端收到訊息之後，利用相同的雜湊演算法將訊息（M'）計算出雜湊值（H'），再與解密（或植入）後的雜湊值（H）相比較，如果兩者相符，則表示訊息正確（或身份正確）；否則表示訊息也許在傳送中發生錯誤（被竄改），或是鑰匙不對（偽造訊息或仿

冒身份)。至於傳送端如何將鑰匙加入雜湊碼之中成為 MAC 碼，又接收端如何驗證 MAC 碼中所植入鑰匙的正確性，則延伸有種演算法，本書僅介紹 MAC-DES 與 HMAC 等兩個 MAC 系統，以下分別介紹之。

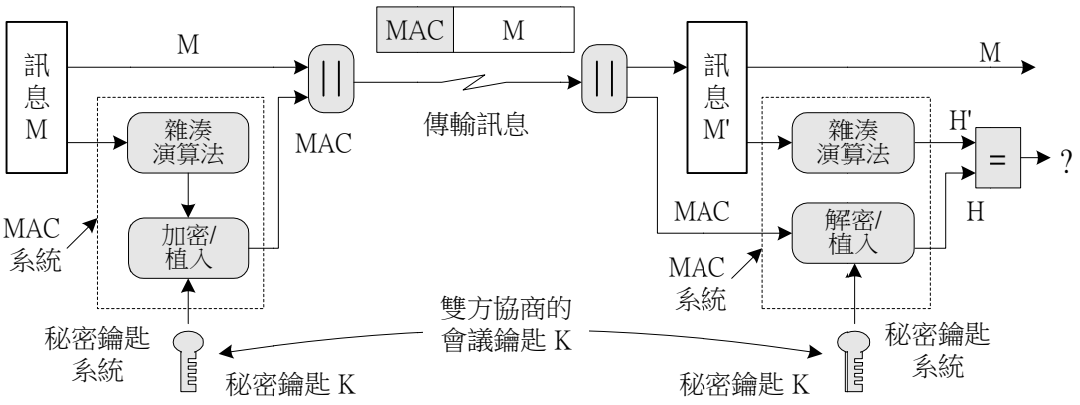


圖 6-2 訊息確認的運作程序

6-4 MAC-DES 演算法

本書到目前為止，已介紹了加密演算法(秘密鑰匙系統與公開鑰匙系統)、雜湊演算法，接下來，就必須將這些技術整合在一起，建構一個完整的 MAC 系統。雖然 MAC 包含了加密演算法與雜湊演算法，但它們之間是混合在一起；至於如何混合這兩種演算法，也是值得研究的課題。

首先介紹一種最常用的 MAC 系統，它是以 DES 加密系統為基礎而建立，一般稱為『MAC-DES』演算法。這種演算法已被 FIPS 發表為標準(FIPS PUB 113)，同時也成為 ANSI 的標準(X9.17)；標準規範內稱之為『資料確認演算法』(Data Authentication Algorithm, DAA)，所產生的確認碼稱為『資料確認碼』(Data Authentication Code, DAC)。

MAC-DES 使用 DES 的 CBC 模式(請參考第 2 章)，並將起始向量設定為零，如圖 6-3 所示。MAC-DES 的運作如下：首先將欲確認的資料(訊息、紀錄、檔案、或程式)以 64 個位元為單位，分割成若干個區塊($D_1, D_2, ..., D_N$)，其中如有不足的部分全部補上 0。接下來，以區塊為單位，連續送給 DES 演算法運算，計算時會輸入 56 位元的加密鑰匙(或稱秘密鑰匙)，輸出的密文仍為 64 個位元。前一區塊所產生的密文，與本次的區塊明文之間執行 XOR 運算，再進入 DES 演算法編碼；依此類推，直到最後區塊的密文輸出(64 bits)。由最後輸出的密文(Q_N)中取最高位元的若干位元(16 ~ 64

bits)，作為 MAC 值（或稱 DAC）的輸出。

由此可見，MAC-DES 的特點是，將加密演算法的操作當作雜湊演算法來使用，而且也順利的將鑰匙嵌入其中。許多應用系統都是採用 DES 標準，因為較容易實現 MAC-DES 演算法。但 MAC-DES 的 MAC 長度只有 64 個位元，防禦暴力攻擊的能力有所不足；通常採用此演算法的鑰匙大多只使用一次(或是一個運作程序)，以策安全。

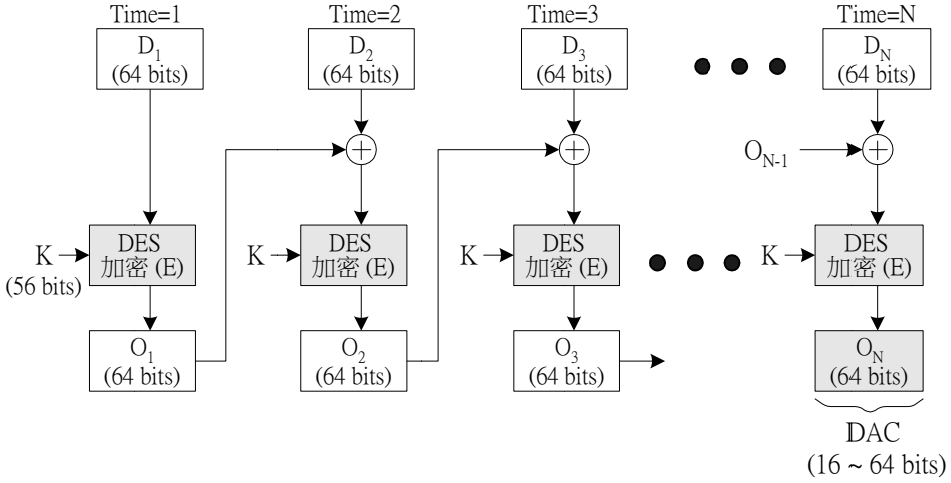


圖 6-3 MAC-DES 運作程序

雖然 MAC-DES 是採用 56 位元鑰匙的 DES 演算法，但許多地方為了提高安全性，也漸漸採用 128 位元的 AES 演算法（如第三章介紹）。

6-5 HMAC 演算法

基本上，『訊息確認碼』(MAC) 及『數位簽章』(DS) 皆是利用加密演算法與雜湊演算法共同組合而成。在許多情況下，我們發現並非得利用加密演算法來建構 MAC 系統不可。若能將秘密鑰匙與訊息混合起來，再經過雜湊演算法處理，可以同時達到加密與訊息確認的功能，此種技術稱之為『雜湊訊息確認碼』(Hash Massage Authentication Code, HMAC) 系統 (RFC 2104)。圖 6-4 為 HMAC 的處理程序，假設通訊雙方都擁有秘密鑰匙 K；發送端將秘密鑰匙崁入訊息之中，一起計算出 MAC 碼，再附加於訊息之後傳送給接收端；接收端以同樣的方法計算出 MAC'；再比較兩者是否相同，如果相同，表示訊息與鑰匙都是正常的（當然也表示發送端並非冒充的）。

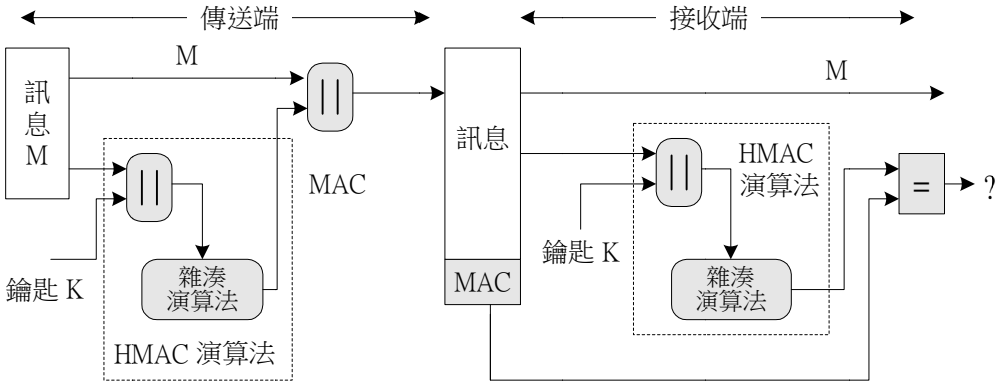


圖 6-4 HMAC 的運作程序

早期開發的雜湊演算法並非針對 MAC 而設計(在資訊安全上有許多地方會用到雜湊演算法)，它也不會使用到鑰匙。因此，一昧的利用加密演算法與雜湊演算法來製造 MAC，並不是十分恰當。目前有許多研究發現，將鑰匙與訊息混合 (排列方式) 經過雜湊演算法計算的 HMAC 方法，反而比較受歡迎。目前 IP security 與 SSL 規範都已採用 HMAC 的訊息確認方法。

然而，HMAC 到底具備哪些優點？我們可從 RFC 2104 中所列出的 HMAC 設計目標看出，歸納如下：

- ◈ 能在不修改現有的雜湊演算法之下，將它嵌入 HMAC 演算法之內。尤其可將那些執行速度快，又可以免費取得的軟體雜湊演算法，植入 HMAC 系統內，不但執行效率高，而且根本不用付費。
- ◈ 當需要更快速的雜湊演算法時、或是新的雜湊演算法被發展出來的時候，我們希望能容易地更新內嵌的雜湊演算法。
- ◈ 當雜湊演算法被嵌入 HMAC 演算法之內後，期望能維持它原來的執行速度，以免造成執行效能大幅滑落。
- ◈ 可以簡單的使用鑰匙。
- ◈ 希望所嵌入的雜湊演算法能滿足一些合理的條件，並且可以根據這些條件分析該 HMAC 的安全強度。

前面兩項條件是 HMAC 被廣泛使用的主要原因，可將雜湊演算法視為一個『黑盒

子』，如此有下列兩種優點：(1) 實作 HMAC 時，可將雜湊演算法當成一個模組來實現，並且有許多現成的程式碼可以直接使用；(2) 當需要更換雜湊演算法時，只要更新模組即可，不會影響到其他程式的架構。在實務應用上，目前已有許多 HMAC 規範使用 MD5、SHA-1 或 ROPEMD-160 等雜湊演算法，並且可由通訊雙方協議出採用何種演算法（如 IPSec 規範）。

6-6 MAC 操作方式

MAC 的操作方式牽涉到雜湊演算法與加密系統之間的處理，至於明文或密文與 MAC 的傳送，以下介紹是幾種較常見的操作方式。

6-6-1 明文與 MAC 傳送

圖 6-7 為明文訊息與 MAC 碼一起傳送，攻擊者可由訊息與 MAC 碼去嘗試各種鑰匙，只要能找出鑰匙 K_1，就可竄改或偽造訊息，來達到攻擊的目的。

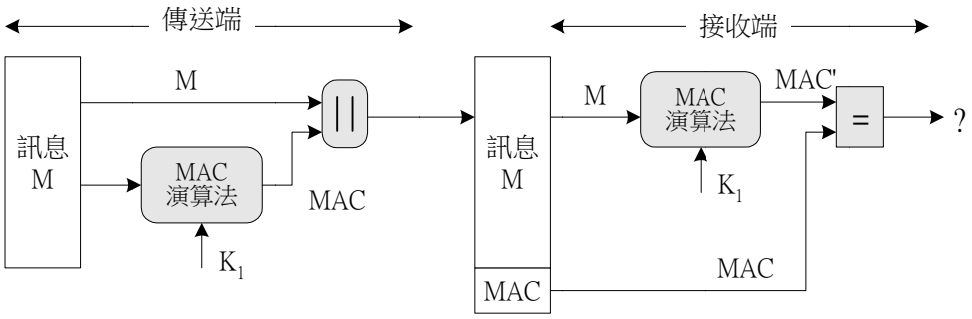


圖 6-7 訊息附加 MAC 碼傳送

6-6-2 明文與 MAC 加密後傳送

圖 6-8 為訊息（M）與 MAC 碼連結之後，經過加密系統加密成為密文（$E_{K1}[M \| MAC]$），再傳送給接收端；這種架構似乎比較安全，攻擊者必須先找出加密鑰匙，才可以破解 MAC 演算法。許多地方為了方便操作，加密與 MAC 都使用同一把鑰匙，但這樣的話，第一關破解成功，也同時解開第二關，對安全性反而沒有幫助。由此可見，採用這種機制必須使用兩把鑰匙，分別處理資料加密與 MAC 加密。

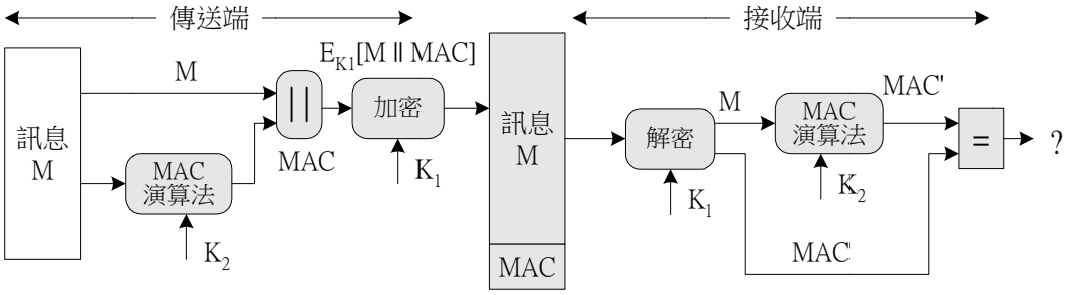


圖 6-8 訊息與 MAC 經加密後傳送

6-6-3 密文與 MAC 傳送

將訊息加密後再與 MAC 一起傳送，如圖 6-9 所示。這種傳輸方式使用兩把鑰匙，看起來似乎安全多了。攻擊者必須先解開訊息的加密鑰匙，得到訊息的明文之後，才可以進一步去攻擊 MAC 演算法；許多安全性較高的傳輸都是使用這種架構，尤其是用在數位簽章時，訊息可以用秘密鑰匙來加密，而數位簽章可以使用私有鑰匙加密（利用公開鑰匙解密），攻擊者必須連破 2 把鑰匙才能達到目的。

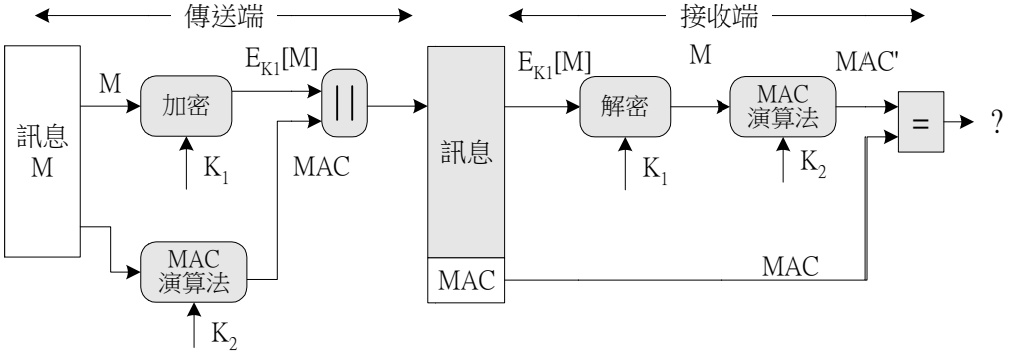


圖 6-9 訊息密文附加 MAC 傳送

6-6-4 密文計算 MAC 後傳送

圖 6-10 為訊息加密再計算 MAC 值，乍看起來好像安全性很高，其實不然。雖然使用了兩把鑰匙，但這兩把鑰匙所處理的資料是分開的（訊息與 MAC），攻擊者可以分別找出 K_1 與 K_2，兩者並不互相衝突。

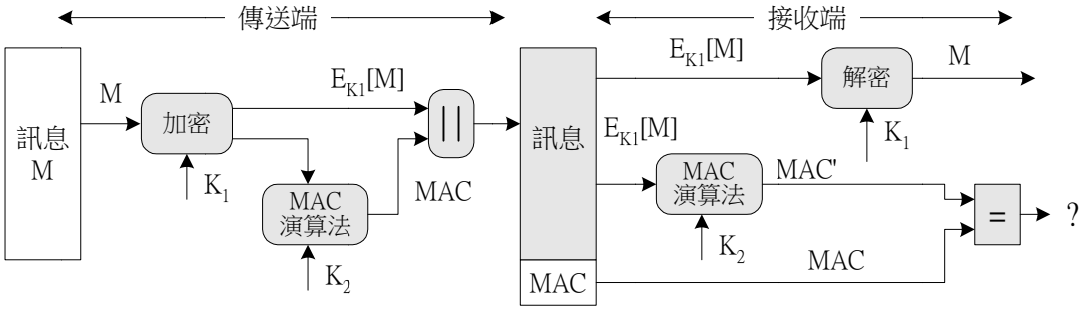


圖 6-10 訊息加密後再計算 MAC