第二章　傳統秘密鑰匙系統

所謂『傳統密碼學』，就是在取代與換位的遊戲規則中，演變出一套加密系統，本章以 DES 演算法為範例來說明這套遊戲的精髓所在，並加入了秘密鑰匙的分配問題。

2-1 密碼學概論

密碼（Cryptography）一字是由希臘文字 “kryptos”（隱藏）和 “graphein”（寫字）組合而生，意思是將原來資料用『位置混排處理』（Position-Scrambling Process），使原始資料的次序呈現不規律狀態，來達到訊息保密的功能。在討論訊息保密過程中，我們將原來資料稱為『明文』（Plaintext），經過編密後稱為『密文』（Ciphertext），這個動作則稱為『加密』（Enciphering）；相反動作稱為『解密』（Deciphering），整體通稱為『密碼系統』（Cryptosystem）。圖 2-1 為密碼系統的運作圖，加密與解密分別使用一把鑰匙（圖 2-1 中 K_1 與 K_2），用下列式子表示加密和解密的運算：

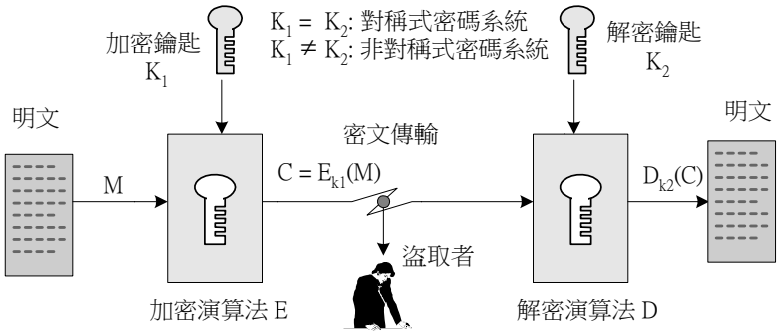


圖 2-1 密碼系統的運作程序

M = 明文

C = 密文

K_1 = 加密鑰匙（key）、K_2 = 解密鑰匙

E = 加密演算法

D = 解密演算法

它們之間的關係為：

加密：$C = E_{k1}(M)$

解密：$M = D_{k2}(C) = D_{k2}(E_{k1}(M))$

加密處理是利用加密演算法 E，與鑰匙 K_1 關鍵因素，將明文 M 編碼處理得到密文 $C = E_{k1}(M)$。加密的反向即是解密處理，即是利用解密演算法 D 與鑰匙 K_2，將密文 C 編碼回原明文 $M = D_{k2}(C)$。如果加密鑰匙和解密鑰匙是同一把鑰匙的話（即 $K_1 = K_2$），則稱之為『對稱式密碼系統』；否則稱為『非對稱式密碼系統』（$K_1 \neq K_2$）。另外，演算法 E 與 D 兩者之間一定有某些關聯性，甚至某些密碼系統（如 DES）兩者採用同一套演算法。

對稱式密碼系統是表示傳送端與接收端共享同一把秘密鑰匙，其加解密也都使用同一把鑰匙。看似簡單，然而如何分配這把秘密鑰匙卻是一大難題。而且為了防患他人盜用，必須嚴密收藏這把鑰匙，故稱為『秘密金鑰系統』（Secret Key Cryptosystem）。

2-2 傳統密碼學的基本原理

『傳統密碼學』（Conventional Cryptosystem）是由兩個基本加密動作所組成，一者是『換位加密法』（Transposition Cipher），另一個是『取代加密法』（Substitution Cipher）。不論多複雜的傳統密碼演算法都是由這兩種加密法演變出來的。欲了解傳統密碼學，首先必須由這兩個基本元件談起，為了方便說明，我們還是以英文字元的加密作為範例來說明。

2-3 換位加密法

『換位加密法』（Transposition Cipher）是利用一個特定排列的規則，將明文中的字元重新排列過，來產生另一個無規律的密文；解密時再使用該特定規則，將密文倒回原來的明文，所謂特定排列規則就是加密與解密的『鑰匙』。我們用最簡單的換位加密法來看其結果，如明文是：

I SIT BY MY WINDOW WAITING FOR YOU

此明文經過加密後成為：

UOY ROF GNITIAW WODNIW YM YB TIS I

由以上的例子，我們很容易看出換位的規則是將明文以相反次序來重寫。所以換位加密法只是改變字母之間的順序位置，並沒有更動字母原來的意義。譬如，上例中字母"U"的位置更換之後，仍然代表"U"原來字母，只是該字母原來在明文中的位置變更而已，這是換位加密法最主要的精神。由上述例子，可以很容易的由明文和密文之間比較出換位的規則（加密鑰匙）。得到換位規則之後，就可以破解爾後利用此規則來加密的任何密文。所以換位法的規則必須有所變化，否則根本沒有保密性可言，以下介紹三種基本的換位加密法。

2-3-1 鐵軌法

『鐵軌法』（Railroad Method）是將明文排列成像鐵軌一樣上下兩行，排列中明文必須扣除空白鍵，並以上下次序排列。明文中不計空白鍵，且明文的字元數必須是四的倍數，不到四的倍數則以 "E" 字元補滿。我們還是用上述的例子來編碼：

- 明文：I SIT BY MY WINDOW WAITING FOR YOU
- 鐵軌排列：

I	I	B	M	W	N	O	W	I	I	G	O	Y	U
S	T	Y	Y	I	D	W	A	T	N	F	R	O	E

- 再將鐵軌排列中的字母，由左至右、由上而下，依序編寫出來，即成為加密後的密文，密文如下：

  密文：IIBMWNOWIIGOYUSTYYIDWATNFROE

- 為了方便起見，我們將密文以每 4 個字母為單位排成一數，期間用空白隔開：

  密文：IIBM WNOW IIGO YUST YYID WANT FROE

以下說明為什麼密文的長度必須是 4 的倍數的原因。當接收端收到此密文之後，因為他知道加密的順序，因此他可將密文由中間切開分成兩半，如下所示：

IIBM WNOW IIGO YU | ST YYID WANT FROE

然後左右兩半依序輪流讀出字母便可還原成原來的明文了：

ISITBYMYWINDOWWAITINGFORYOU

當然，鐵軌法是用兩行來排列，我們也可以用三行或四行來排列，轉換成更複雜的編碼技巧。這些簡單的變形，請讀者自行玩一玩。

2-3-2 鑰匙排列法

『鑰匙排列法』的排列與鐵軌法非常相似，但是使用多行列矩陣方式排列；同鐵軌排列法將明文的長度調整為四的倍數，調整後的明文由左至右；由上而下的順序排列填入矩陣的方格中，如果還有空格的話，則填入"E"來補滿。另外採用一把取出密文的鑰匙，鑰匙的長度與矩陣的行數目相同；每相對應行的鑰匙數字，為取出密文的次序。譬如，鑰匙為 {4, 3, 1, 2, 5, 6, 7}，則表示第 3 行是第一個被取出，接著取出第 4 行；依此類推。取出每一行內的字母時，依由上至下的順序。

(A) 加密範例如下：

◈ 明文：I SIT BY MY WINDOW WAITING FOR YOU

鑰匙：	4	3	1	2	5	6	7
明文：	I	S	I	T	B	Y	M
	Y	W	I	N	D	O	W
	W	A	I	T	I	N	G
	F	O	R	Y	O	U	E

填入矩陣後，可依照鑰匙所指定的順序取出矩陣內字母，即得密文如下：

◈ 密文：IIIRTNTYSWAOIYWFBDIOYONUMWGE

(B)解密範例如下：

解密時，只要依照鑰匙所指定的順序將密文填入矩陣中，再由左至右、由上到下的順序取出，便可以還原明文了。

◈ 密文：IIIRTNTYSWAOIYWFBDIOYONUMWGE

每四個字元一組：IIIR TNTY SWAO IYWF BDIO YONU MWGE，依序填入陣列內，如下：

鑰匙：	4	3	1	2	5	6	7
密文：	I	S	I	T	B	Y	M
	Y	W	I	N	D	O	W
	W	A	I	T	I	N	G
	F	O	R	Y	O	U	E

再由左至右，從陣列內取出明文，如下：

◈ 明文：I SIT BY MY WINDOW WAITING FOR YOU

如果一次排列的加密易被破解的話，可利用同樣一把鑰匙重複加密，以增加其複雜度與破解的困難度。

2-3-3 位元換位箱

前面所介紹的兩種方法，都是採用字元的換位方法。但在資訊系統上大多是以二進位元來表示資料，因此，我們必須針對二進位中數字來排列移位，以達到加密的功能。二進位的移位排列最常用的是『換位箱』（Permutation Box）換位方法，如圖 2-2 所示。解密時，我們只需要以反向對應即可將密文還原成明文。換位箱的方法是許多加密法中的主要配備，也是資訊安全加密法的根本基礎。

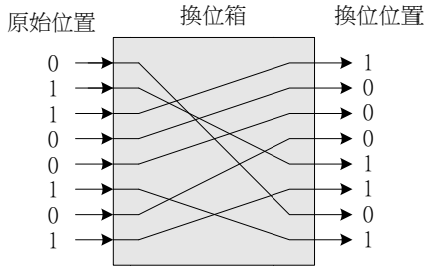


圖 2-2 位元換位箱

2-4 取代加密法

簡單的說，『取代加密法』（Substitution Cipher）就是將原文中的字母由另一個字母來取代，取代的順序不一定有一定的規則。加密與解密雙方必須共同擁有一份替換表，而此替換表便就是雙方共享『鑰匙』。這與換位法之間有很大的不同，換位法只改變字母排列順序，原文的字母與加密後的字母還是一樣，只是位置不同而已。然而，使用取代加密法則不然，加密後的字母與原文字母完全相異。我們用一個簡單的替換表，來觀察取代加密法的運作程序；替換表如下：

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

我們還是利用前一節的明文為例子來看加密後的密文結果：（空白鍵不計）

明文：I SIT BY MY WINDOW WAITING FOR YOU

密文：RHRGYBNBDRMWLDDZRGRMTULIBLF

利用替換表的取代加密法太容易被破解了，這是因為每個字母都有一定的替換字母;且在我們日常生活中常常重複出現許多單字或句子,而這些單字大多有一定的規則,有心人士只要去統計一些常出現的單字，便可以找出替換表了。以下介紹兩種較複雜的取代法。

2-4-1 旋轉取代法

利用替換表最大的缺點就是取代字母是固定的，我們可以針對這個缺點來改進，讓替換字母隨著一定的規則改變，這就是『旋轉取代加密法』（Rotation Substitution Cipher）。我們將兩個圓盤重疊在一起，下面圓盤是固定的，而上面的圓盤可以左右轉動，就好像保險箱的號碼鑰匙一樣。上下圓盤都刻有英文字母，下圓盤代表原文字母；而上圓盤表示加密後的字母，如圖 2-3 所示。

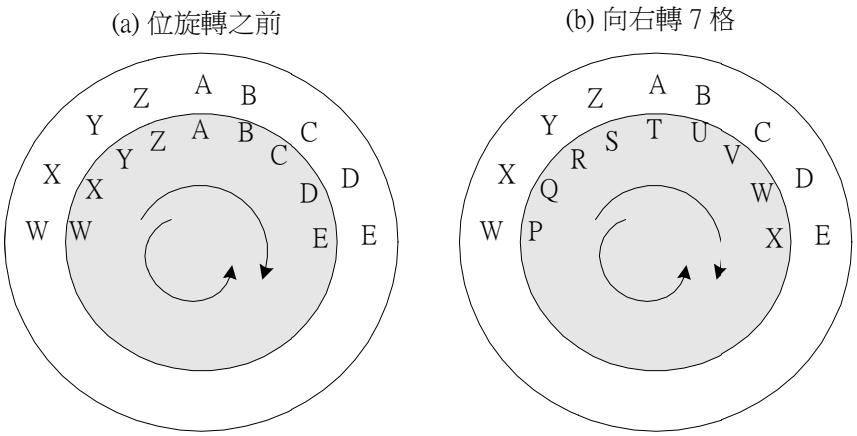


圖 2-3 旋轉取代加密法

我們將圖 2-3 (a) 的上旋轉盤向右轉 7 格後如圖 2-2 (b) 所示，所編出來的密文如下：（空白不計）

明文：I SIT BY MY WINDOW WAITING FOR YOU

密文：BLBMURFRPBGWHPPBMBGZYHKRHN

乍看起來，旋轉替換取代法與替換表並沒有兩樣，不要緊張，好戲還在後頭呢！加密與解密雙方可以協議旋轉盤的轉動次序，作為雙方的秘密鑰匙；譬如，鑰匙為 {5, 7, 4, 2}，則表示第一個字母轉動 5 格、第二個字母再轉動 7 格、第三個字母再轉動 4 格、第四個字母再轉動 2 格，第五個字母以後又恢復 5、7、4、2 的轉動，依此類推。如此一來，表示替換表隨每一個字母變化。這樣還是不夠的話，還可以如保險箱的密碼一樣，增加了左右轉動，夠複雜了吧！

2-4-2 旋轉機

一層的旋轉替換還不夠複雜的話，我們可以結合多個旋轉盤成為一個『旋轉機』（Rotor Machine），夠恐怖了吧！這是第二次世界大戰時，德軍讓聯軍百思不解的加密技巧，最後還是由一個俘虜中得到破解的方法，到底聯軍還是無法自己破解它。以圖 2-4 為範例，德軍非常聰明的將旋轉機安裝在打字機內，其中包含了三個旋轉盤，旋轉盤內部都有電線直接連接，旋轉盤之間也可接觸電流導通。每天的最後一次通訊，由指揮中

心以密語通知每一個旋轉盤的轉動次數。當通訊者敲入每一個字母（明文輸入）時，電流將隨者轉盤接觸導通道另一個字母位置，然後發射出該字母的電波訊號。

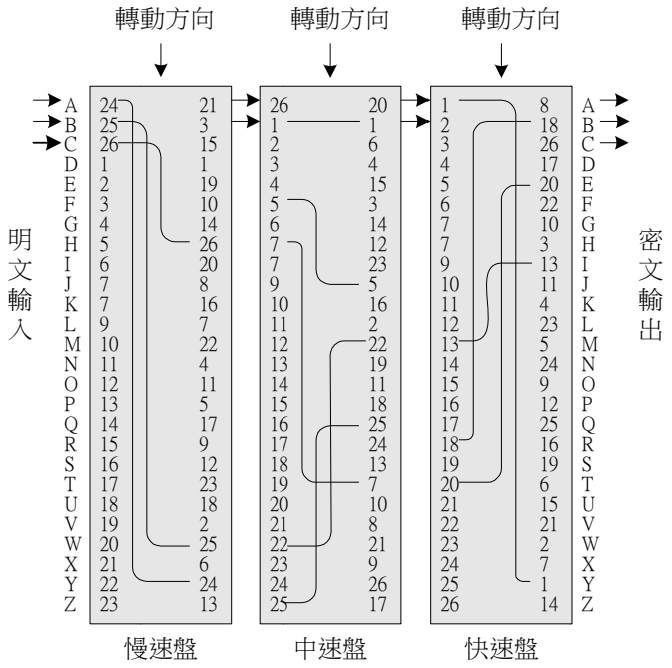


圖 2-4 旋轉機加密法

2-5 區塊加密法的基本原理

數位資料是一串不定長度的 0 與 1 組合而成，對於它的密碼系統就很難直接用取代與換位方法，而需要資料分段來加密或解密，這就是『區塊加密法』(Block Cipher)。它將不定長度的資料，以某一固定長度為單位（大多是 64 bits），分割為若干個區塊。每一明文區塊經過加密編碼後，產生相同長度的密文區塊。圖 2-5 為區塊加密法的概略圖，明文經過分割區塊後，一個接著一個進入加密器處理；處理完成後，密文區塊也是一個接一個輸出。實作上並非完全如此，如果區塊之間完全是獨立加密的話，攻擊者也可以分別採用獨立的密文與明文配對來破解。因此，輸出密文區塊之間需要特殊處理才行，這方面稱為操作模式，於 2-7 節中再介紹。

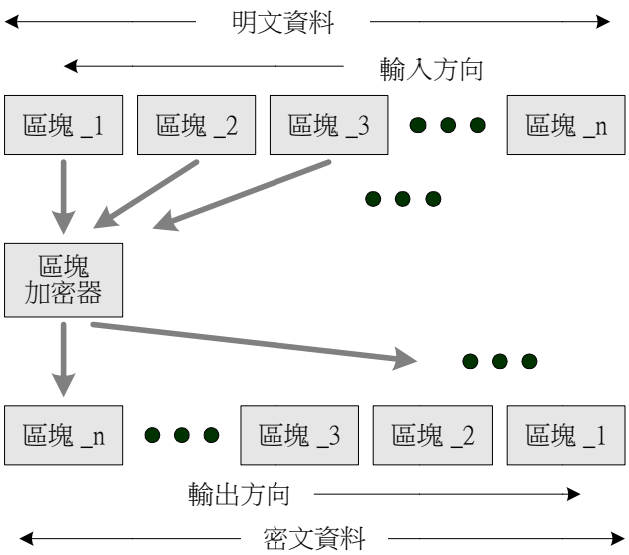


圖 2-5 區塊加密法

一個明文資料 M 被分割成許多小區塊後，表示為：$M_1 \| M_2 \| M_3 \cdots \| M_n$，每個區塊都使用相同的鑰匙（key）來加密，最後可得到密文：

$$C = E_k(M) = E_k(M_1) \| E_k(M_2) \| E_k(M_3) \| \cdots \| E_k(M_n)$$

既然，我們將明文區分為若干個區塊，來產生同等量的密文。區塊加密法就是探討如何將這固定長度的明文，加密成相同長度的密文。

2-6 Feistel 密碼演算法

2-6-1 Feistel 系統概念

Feistel 是將乘積加密法導入區塊加密法的先趨，到目前為止，許多密碼系統還是引用 Feistel 的基本架構。他的基本原理是利用 XOR(⊕)的的基本特性，如下：

(1) If $A \oplus B = C$ 則：

$B \oplus C = A$ and $A \oplus C = B$

(2) $0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0,$

(3) $A \oplus 0 = A$

驗證：

If A = 11011010，B = 10110111

11011010
10110111

A⊕B = 01101101 = C

then

10110111
01101101

B⊕C = 11011010 = A

and

11011010
01101101

A⊕C = 10110111 = B

圖 2-7 為 『Feistel 加密法』（Feistel Cipher）的基本架構，它兩個重要的特點如下：

(1) 加密與解密都是相同的編碼器，亦是，明文經由編碼器處理後，則輸出為密文；密文再經過編碼器處理後，即還原明文。

(2) 不受取代裝置 F 函數的影響，亦是，無論 F 函數為任何功能，都不會影響編碼器還原明文的功能。

(3) F 函數取有取代加密的功能，越複雜則明文與密文之間的複雜度越高，亦是，越 F 函數越複雜，由密文猜測出明文越困難。

(4) 它將所欲編碼的資料(64 bits)分割成兩群：右邊(R，32 bits)與左邊(L，32 bits)，再交叉換位，因此，也具有換位加密的功能。

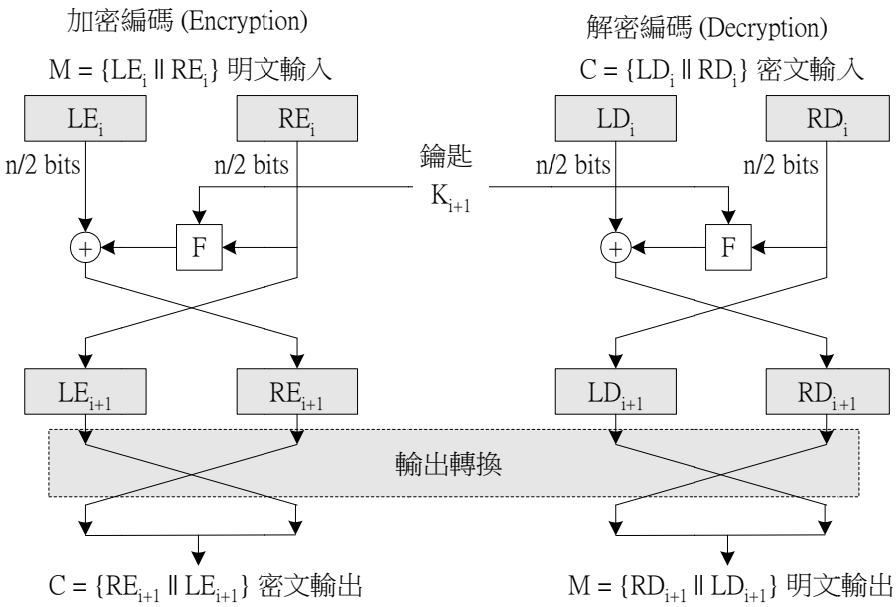


圖 2-6 Feistel 密碼系統的基本觀念

圖 2-6 僅使用一支鑰匙加密或解密處理，多支鑰匙連續處理也如同此道理，但輸出轉換僅會在最後支鑰匙才會出現。左邊是加密編碼處理，它將 n 位元的明文區塊(一般都 64 bits)分割成為左右兩個子區塊(n/2 位元，32 位元): LE_i 和 RE_i($\{LE_i \| RE_i\}$，其中 || 符號表示排列的意思) ；再使用鑰匙 K_{i+1} 對左右子區塊加密，而得到另外兩個子區塊 LE_{i+1} 和 RE_{i+1}。

(A)加密後推演如下：(圖 2-6 左邊處理)

明文輸入：$M = \{LE_i \| RE_i\}$

$$LE_{i+1} = RE_i \qquad \text{式子 (1)}$$

$$RE_{i+1} = LE_i \oplus F(K_{i+1}, RE_i) \qquad \text{式子 (2)}$$

密文輸出：$C = \{RE_{i+1} \| LE_{i+1}\} = \{ LE_i \oplus F(K_{i+1}, RE_i) \| RE_i \}$

經過該支鑰匙加密編碼後所得到為 $\{LE_{i+1} \| RE_{i+1}\}$，如再經過輸出轉換 (左右區塊交換位置) ，則密文輸出為 $C = \{RE_{i+1} \| LE_{i+1}\}$。上述中函數 $f(K_{i+1}, RE_i)$ 表示 R_i 使用鑰匙 K_{i+1} 加密，F() 函數表示一個特殊處理的取代加密器，然而取代函數是依照鑰匙來變化。另外，⊕ 符號表示 Module-2 的加法 (XOR 的運算) 。

(B)解密推演如下：(圖 2-6 右邊處理)

密文輸入：$C = \{LD_i \,||\, RD_i\}$

$LD_{i+1} = RD_i$ 式子 (3)

$RD_{i+1} = LD_i \oplus F(K_{i+1}, RD_i)$ 式子 (4)

$LD_i = LE_{i+1}$ 式子 (5)

$RD_i = RE_{i+1}$ 式子 (6)

則 Feistel 解碼器輸出為：

$LD_{i+1} = RD_i$ (引用式子 (3))

$= LE_{i+1} = RE_i$ (引用式子 (6) 與 (1))

$RD_{i+1} = LD_i \oplus F(K_{i+1}, RD_i)$ (引用式子 (4))

$= RE_{i+1} \oplus F(K_{i+1}, LE_{i+1})$ (引用式子 (5) 與 (6))

$= LE_i \oplus F(K_{i+1}, RE_i) \oplus F(K_{i+1}, RE_i)$ (引用式子 (2) 與 (1))

$= LE_i \oplus 0$ (因為：$A \oplus A = 0$)

$= LE_i$ (因為：$A \oplus 0 = A$)

經過解密編碼後輸出為 $\{RE_i \,||\, LE_i\}$，再經過輸出轉換得到 $M = \{LE_i \,||\, RE_i\}$，其與圖 2-7 左邊明輸入相同，因此，Feistel 編碼系統具有加密與反向解密的功能。另一個更直得注意的重點是，編碼過程中，不受加密鑰匙與 F() 函數所影響。有就是說，不論鑰匙是何值，編碼函數 F 為何，Feistel 編碼皆具有加密與反向解密的功能。如以『取代加密法』與『換位加密法』概念而言，圖 2-7 中將明文分割成左右兩段並交換處理，即是『換位』；另外，經由 $F(K_{i+1}, RE_i)$ 函數處理，即是『取代』處理。

(C)驗證 Feistel 編碼器功能

我們用一個範例來驗證 Feistel 架構是否可行。假設 $M = \{4, 8\}$、$k_i = 6$、$F = \oplus$。則加密過程為：

$LE_i = 4$、$RE_i = 8$

$RE_{i+1} = LE_i \oplus F(RE_i, K_i)$

$=4 \oplus 8 \oplus 6 = A$

$LE_{i+1} = RE_i = 8$

則密文為：$C = \{RE_{i+1}, LE_{i+1}\} = \{A, 8\}$

解密過程為：

$LD_i = A$、$RD_i = 8$

$LD_{i+1} = RD_i = 8$

$RD_{i+1} = LD_i \oplus F(RD_i, K_i)$

$= A \oplus 8 \oplus 6 = 4$

則明文為：$M = \{RD_{i+1}, LD_{i+1}\} = \{4, 8\}$

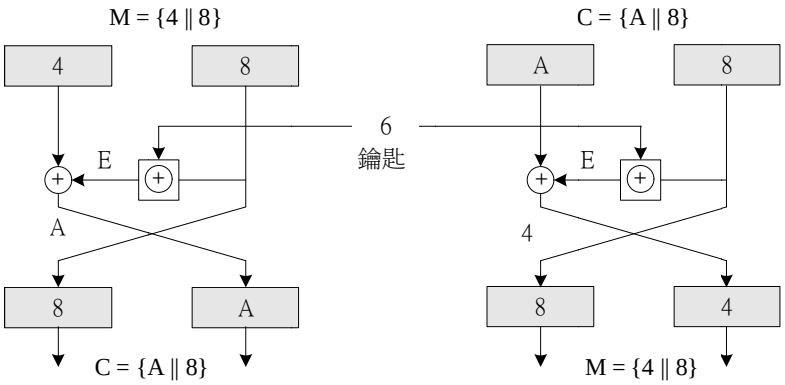


圖 2-7 驗證 Feistel 具有加密/解密功能

2-6-2 Feistel 系統架構

Feistel 的構想是如能利用圖 2-7 的加密程序，重複運算幾次，並且每一次都給予不同的子鑰匙，便能完全打散資料的關聯性，達到複雜加密的效果。圖 2-8 為 Feistel 密碼系統的架構圖，我們將重複加密的次數（假設 16 次），以相同的加密方塊連結畫出來；如此，就可以看出明文是經過多個加密方塊連續的運算出密文來；值得一提的是，加密與解密均使用同樣的演算法，但子鑰匙分配的次序顛倒。

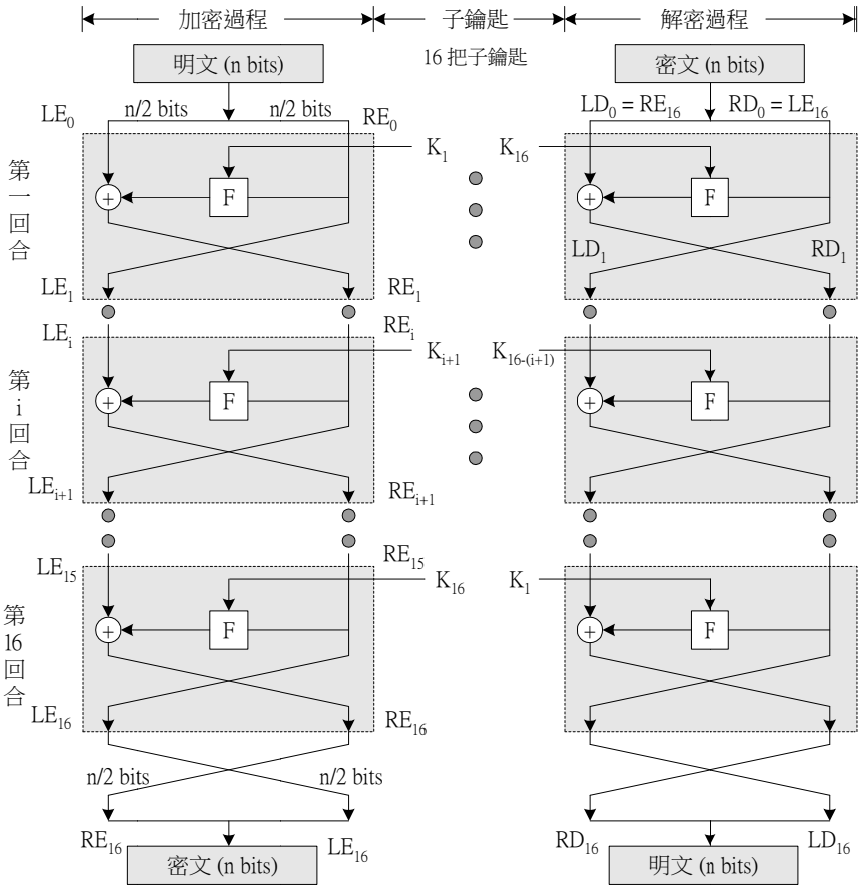


圖 2-7-1 Feistel 密碼系統架構

Feistel 編碼系統的運作程序是通訊雙方皆擁有一把相同的秘密鑰匙 K，再利用某一鑰匙產生函數，將此鑰匙製作出多把子鑰匙（K_i，i=1, 2, .., 16）。傳送端將訊息利用子鑰匙由 $K_1, K_2, ..., k_{16}$，連續編碼後再經由輸出轉換產生密文，發送給接收端。接收端收到密文後，也將秘密鑰匙 K，經過相同的鑰匙產生函數，製造出相同的子鑰匙序列，利用相同的 Feistel 編碼器，將子鑰匙序列倒過來，由 $K_{16}, K_{15}, ..., k_1$，連續編碼後再經過輸出轉換解碼回原明文。

2-7 傳統密碼系統的摘要

自從 Feistel 發表 取代與重排所建立的『乘積加密法』之後，20 多年來許多密碼學專家，藉其基本架構發展出各種形式的加密演算法，譬如 DES、IDEA 等等。圖 3-1 為秘密鑰匙演算法的基本模型，其運作程序是：首先將輸入的明文區塊經過『初始排列』，使其適合接下來的加密運算；接著，為了增強明文與密文之間的複雜度（擴散能力），將明文經過多次的重覆排列（如 DES 為 16 次排列），不同的演算法都有各自的排列

格式。另一方面，為了增加密文與鑰匙之間的複雜度（混淆能力），首先將加密鑰匙轉換成若干個子鑰匙（如 DES 為 16 把子鑰匙），每一子鑰匙作為加密函數的關鍵性輸入，加密函數是將明文區段與子鑰匙之間，做某些邏輯運作與替代轉換的功能，也是加密演算法最主要的精髓所在，經由許多密碼學專家研究出各種不同的操作方法，延伸出來許多加密驗算法的規範。然而，明文（或稱演算中的密文）重新排列與加密函數混合重覆執行多次，每次輸入不同的子鑰匙，來增加『混淆』與『擴充』的能力。密文經過多次的『取代-移位』之後，再經過最後的『輸出排列』使其回復原來明文的格式，也就是說，『輸出排列』是『初始排列』的反轉換函數。

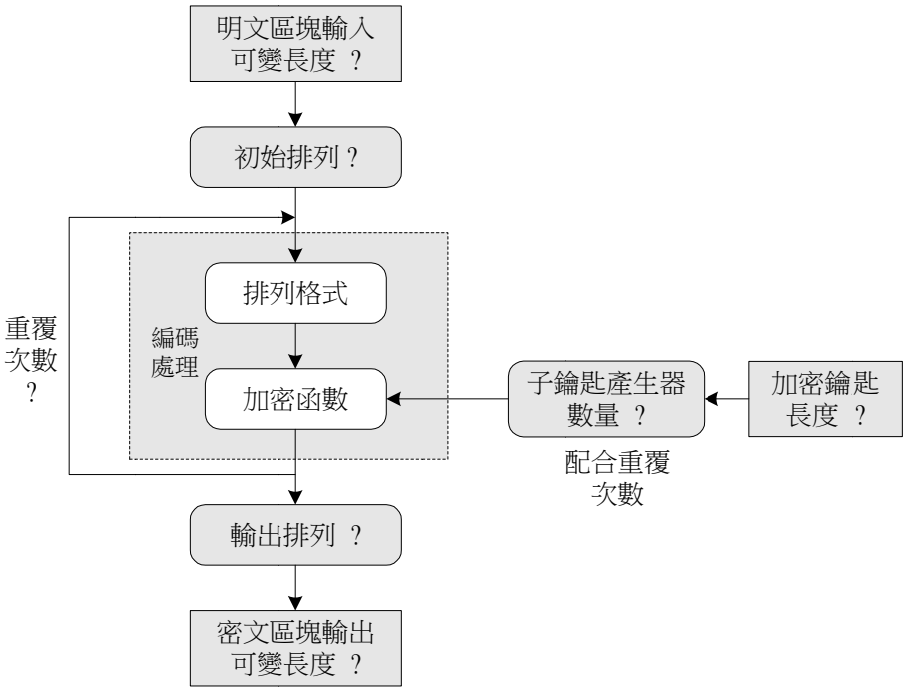


圖 2-8 秘密鑰匙演算法的基本模型

在實作或研究秘密鑰匙演算法方面，可由圖 2-8 的基本模型中設計及更改各種參數，來增加演算法的複雜度（或強度），相關參數如下：

◈ 區塊大小：加密較大區塊的資料，可增加密文的複雜度，但也會增加解密的時間。新的演算法多半提供 128 或 192 bits（DES 為 64 bits），甚至可以使用變動長度來加密。

◈ 初始排列：一般初始排列的功能是為了配合編碼處理，較常用的做法是將明文區塊區分為若干個群組（DES 為兩群組），由這些群組作為重新排列的對象；至於如何將明文區塊劃分為群組的選取方法，各種演算法皆有其獨特的技巧。

◈ 排列格式：一般情況，排列格式即是將上一回合加密的結果，重新排列（DES 為左右交叉排列）後再給予本次加密（重複性），如此更可增加演算法的複雜度。

◈ 編碼函數：這是各種演算法的精髓所在，它是將每個輸入位元以取代方式轉換成其它位元，而在替代過程當中，每一位元也會與子鑰匙的相對應位元之間，做一些邏輯運算（如 XOR 或 AND）。如何設計它，即是許多延伸密碼系統的關鍵因素。

◈ 鑰匙長度：較長的鑰匙較為安全（抗拒暴力攻擊），但相對的，演算法的複雜度就越高，DES 所採用的 56 位元長度，其安全性已岌岌可危，128、192、256 位元長度漸受青睞。

◈ 子鑰匙的數量：由加密鑰匙（或稱主鑰匙）所計算產生子鑰匙數目的多寡，這牽涉到編碼處理的次數，基本上，每回合的編碼處理都需要一組子鑰匙(一把或多把)。

◈ 重覆回合次數：重覆次數越多，所產生的密文也就越複雜（強度也就越高），但相對增加解碼時間。

◈ 輸出轉換：一般都是『初始轉換』的反函數，也就是，將相對應的位元回覆原來的位置。

以上是秘密鑰匙演算法的基本參數，隨著不同的需求，可由基本模型中演變出各式各樣的演算法。

2-8 DES 密碼系統

2-8-1 DES 的演算流程

『資料加密標準』（Data Encryption Standard, DES） 是 IBM 公司於 1970 年代中期所發展出來的加密系統，並於 1977 年被美國國家標準局製定為標準。DES 在

工業界已使用了 20 幾年，它的處理速度非常快，安全性也相當高。DES 演算法的製作原理大多來自 Feistel 架構（如圖 2-8 所示），它的特性歸類如下：

- ◈ 鑰匙長度：56 bits（主鑰匙）。
- ◈ 區塊長度：64 bits。在 DES 演算法中，明文與密文區塊的長度都是 64 bits。
- ◈ 重複次數：每個區塊重複經過加密器計算 16 次，每次計算時使用一把獨立的子鑰匙。
- ◈ 子鑰匙產生器：主鑰匙經過產生器處理之後，產生 16 把子鑰匙，每一把子鑰匙的長度為 48 bits。
- ◈ 演算法相容：基本上，加密與解密演算法是相同的，解密時將子鑰匙輸入順序與加密相反即可。

DES 加密區塊的長度為 64 bits，但一般明文資料都會超過 64 bits，因此以 64 bits 為單位，將明文分割為若干個區塊（如圖 2-5 所示），如果最後不足 64 bits，則以 “0” 補足。所產生的密文與明文長度相同，解密時，如有補足部分再將其刪除。另一方面，DES 所用的加密和解密鑰匙的長度也是 64 位元，但在這 64 位元主鑰匙之中，有 8 個位元是同位元檢查碼，是用來檢出錯誤，所以真正用來加密的長度只有 56 個位元。

圖 2-9 為 DES 演算法的加密流程，其中包括下列步驟，每一步驟都有其特殊功能，說明如下：

1. 明文資料分割，每一區塊 64 bits。最後區塊如果不足於 64 bits，則以 “0” 補足 64 bits。
2. 初始排列（Initial Permutation，IP）（換位功能）。
3. 選擇鑰匙（key）K_i，共有 16 把鑰匙，每一把鑰匙的長度是 48 bits。
4. 加密處理（換位與取代功能）。
5. 重複加密處理 16 次，每次使用不同的鑰匙加密。

6. 終結排列（Final Permutation，FP）（換位功能）。

簡單說明如下：開始時，明文資料已被分割成干個區塊，每區塊長度為 64 位元。當明文區塊（M）經過初始排列（IP）處理後得到 $T_0 = IP(M)$；然後將 T_0 切成兩半（$T_0 = L_0 \| R_0$），再由鑰匙（K）所產生的子鑰匙表（共有 16 把）中選擇出一把子鑰匙，子鑰匙與 T_0 一起進入加密處理器，處理後產生 T_1（$= L_1 \| R_1$）；下一回合，再重複選擇另一把子鑰匙和 T_1 作加密處理以產生 T_2（$= L_2 \| R_2$）；重複 16 次相同的步驟（共選用了 16 個鑰匙），得到 T_{16}（$= L_{16} \| R_{16}$）。之後再經過終結排列（FP）處理得到密文 $C = FP(T_{16})$。以下就依照圖 2-9 內的功能方塊圖來分別介紹，我們就可以瞭解整個 DES 的製作原理了。

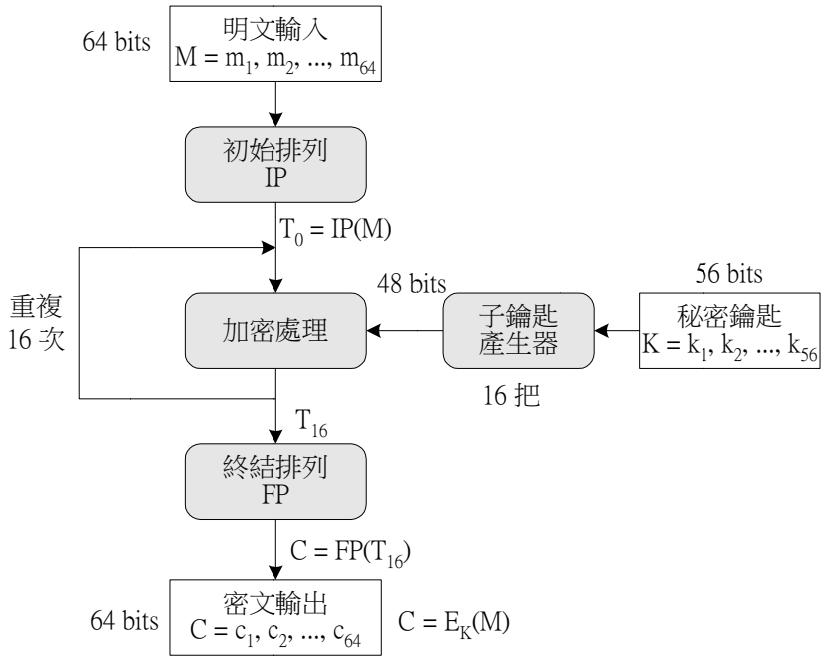


圖 2-9 DES 演算法的加密流程

2-8-2 DES 密碼系統架構

DES 演算法還是承襲 Feistel 密碼系統架構（如圖 2-8 所示）。我們將連續運算 16 回合的順序，以連接加密處理器的方式將它繪出來，如圖 2-11 所示（祇繪出加密部分，解密部份也雷同），其中陰影部分表示每一回合的運算情形。它的運作程序說明如下：

明文區塊 M 經過初始排列處理後得到 $T_0 = IP(M)$，然後將 T_0（64 位元）拆成左右兩個子區塊，各為 32 位元，則 T_0 成為：

$$T_0 = L_0 \,||\, R_0$$

L_0 和 R_0 經過如下的加密處理：

$$L_1 = R_0$$

$$R_1 = L_0 \oplus F(R_0, K_1)$$

得到 $T_1 = L_1 \,||\, R_1$，其中 K_1 是由鑰匙選擇表的 16 個鑰匙中選出的。$f(R_0, K_1)$ 表示 R_0 和 K_1 作計算處理，之後再和 L_0 作 Module-2 加法（亦是 XOR 運算）得到 R_1。重複 16 次計算最後可得到 $T_{16} = L_{16} \,||\, R_{16}$，如圖 2-11 所示。如果以 T_i 表示第 i 次計算，而 L_i 和 R_i 表示 T_i 的右半部和左半部區塊，第 i 次的加密處理關係式如下：

$$L_i = R_{i-1}$$

$$R_i = L_i \oplus F(R_{i-1}, K_i)$$

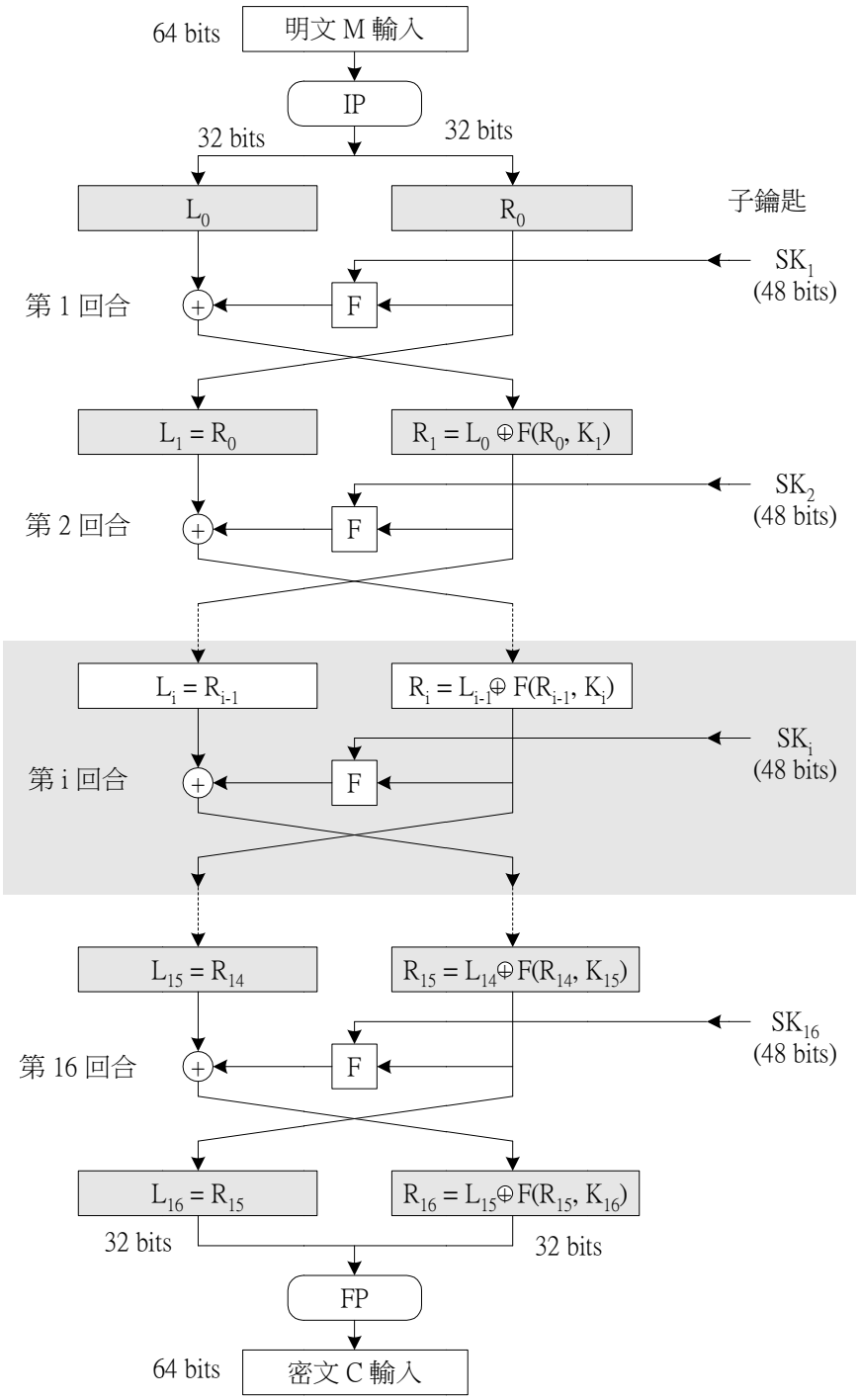


圖 2-9-1 DES 演算法的加密流程

2-9 DES 操作模式

區塊加密法是連續的將明文區塊一個接一個加密成密文區塊，如圖 2-5 所示。基本上，每一明文區塊各對應著一個密文區塊，如果沒有經過特殊處理的話，破解者可以比對明文與密文區塊配對，則可容易地找出加密鑰匙。為了增加密文破解的困難度，我們將連續產生的密文區塊之間做特殊處理，如此一來，破解者就無法找出明文與密文區

塊配對，增加破解的困難度。如何增加密文之間混擾關係，DES 密碼系統提供下列四種操作模式：

- ◈ 電子密碼書（Electronic Code Book, ECB）模式
- ◈ 密文區段串接（Cipher Block Chaining, CBC）模式
- ◈ j-位元密文反饋（j-bit Cipher Feedback, CFB）模式
- ◈ j-位元輸出反饋（j-bit Output Feedback, OFB）模式

以下分別介紹這四種操作模式。

2-9-1 電子密碼書模式

『電子密碼書』（Electronic Code Book）是區塊加密演算法，最基本操作模式。它的做法是將每一明文區塊都對應著一個密文區塊，密文區塊之間沒有經過特殊處理，如圖 2-20 所示。圖 2-20 (a) 中輸入的訊息（Message, M）也許是不定長度的，首先將補足成 64 位元區塊的整數倍（明文，P），再分別進入 DES 加密器（或演算法）處理；每一區塊(P_K, k =1, 2, .., N)加密後也成為 64 個位元的密文區塊(C_K, k =1, 2, .., N)。解密的操作方式（圖 2-20 (b)）也如同加密一樣，一個區塊接著一個區塊進入 DES 加密器（或演算法）處理，再將各個區塊組合回原明文格式，接著將補足的位元刪除掉即可。可下列式子來表示：（鑰匙 K）

◈ 加密處理：

明文：$P = P_1 \| P_2 \| P_3, \ldots, \| P_N$

密文：$C = E_K(P) = E_K(P_1) \| E_K(P_2) \| E_K(P_3) \|, \ldots, \| E_K(P_N)$

◈ 解密處理：

密文：$C = C_1 \| C_2 \| C_3, \ldots, \| C_N$

明文：$P = E_K(C) = E_K(C_1) \| E_K(C_2) \| E_K(C_3) \|, \ldots, \| E_K(C_N)$

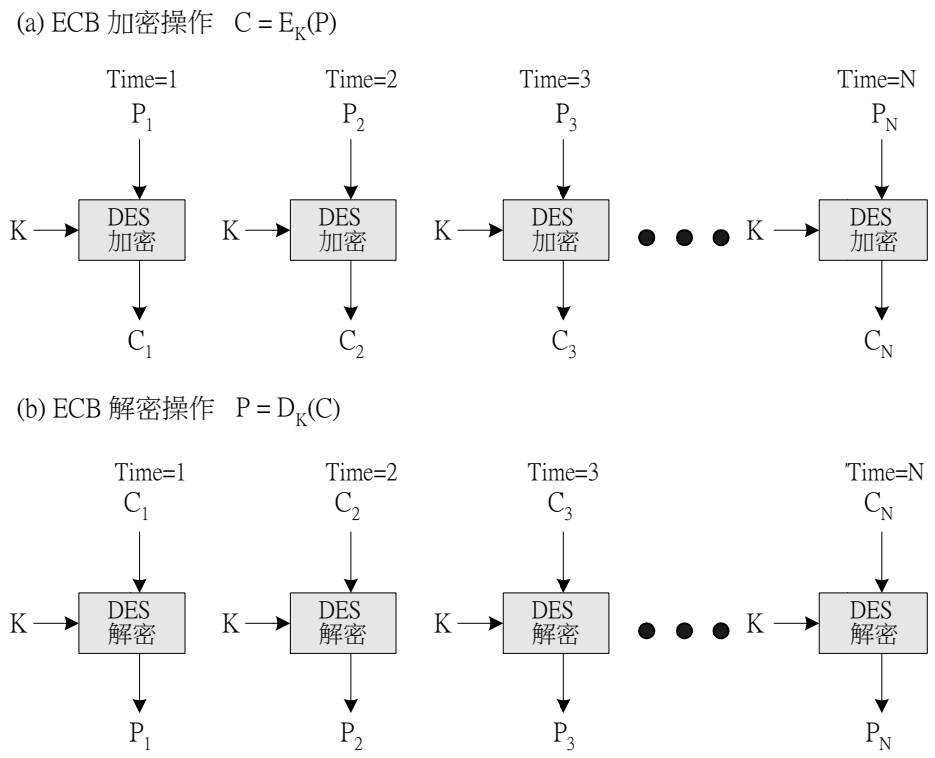


圖 2-20 電子密碼書的操作模式

電子密碼書模式只適合用在較少量的資料加密，譬如，只傳送一把秘密鑰匙給對方，則利用此模式即可。如果對於大量資料的傳輸，ECB 模式便不大適合，它的特點是明文和密文之間都是 64 個位元的區段對照。破解者只要用『明文攻擊』，發送給加密者明文，而得到相對應的密文，再將明文與密文區分各個區段來比對，如果資料過多的話，便非常容易找出它的秘密鑰匙。

2-9-2 密文區塊串接模式

ECB 模式最大缺點是可以比對出明文與密文區塊之間的關係，藉此找出加密鑰匙，『密文區塊串接』(Cipher Block Chaining, CBC)模式就是為了彌補這個缺憾而設計的。除此之外，CBC 為了增加破解的難度，於是加入『起始向量』(Initialization Vector, IV) 的措施。CBC 的每一區段同樣是 64 個位元，但相同的明文區段並不會產生同樣的密文區段，操作方式如圖 2-21 (a) 與 (b) 所示。在第一輪迴時 (Time = 1)，P_1 與起始向量(IV，64 bits)之間作 XOR 運算，然後再將運算後的結果導入加密器(或演算法)處理，而得到第一個區段的密文 C_1；第二輪迴時(Time=2)，將上一次的加密區塊(C_1，64 bits) 與本次的明文區塊做 XOR 運算，然後再將其結果導入加密器作處理，而得到

本次輪迴的密文區塊 C_2；依此類推，先將上一次的密文與本次的明文區塊做 XOR 運作後，再進入加密器編碼，如此便稱為『密文區段串接』。每一密文區段的輸出除了與本身明文有關之外，又加上一次的密文來產生本次的密文區段，因此無法比對出每一區段明文與密文之間的關係。解密的處理程序（如圖 2-21 (b) 所示），只要將加密的運作程序轉換過來即可，我們將 CBC 模式的處理方式以下列方程式表示：

加密運算程序：

$$C_1 = E_K(IV \text{ 与 } P_1)$$

$$C_i = E_K(C_{i-1} \text{ 与 } P_i)；\quad i = 2, 3, 4, \ldots N$$

$$C = C_1 \| C_2 \| C_3 \| \ldots \| C_N$$

解密運算程序：

$$P_1 = D_K(C_1) \oplus IV$$

$$P_i = D_K(C_i) \oplus C_{i-1}；\quad i = 2, 3, 4, \ldots N$$

$$P = P_1 \| P_2 \| P_3, \ldots, P_N$$

驗證：

$$D_K(C_i) = D_K(E_K(C_{i-1} \text{ 与 } P_i))；\quad i = 2, 3, 4, \ldots N$$

$$= (C_{i-1} \text{ 与 } P_i)$$

所以 $D_K(C_i) \oplus C_{i-1} = (C_{i-1} \text{ 与 } P_i) \text{ 与 } C_{i-1} = P_i$ 得證之。

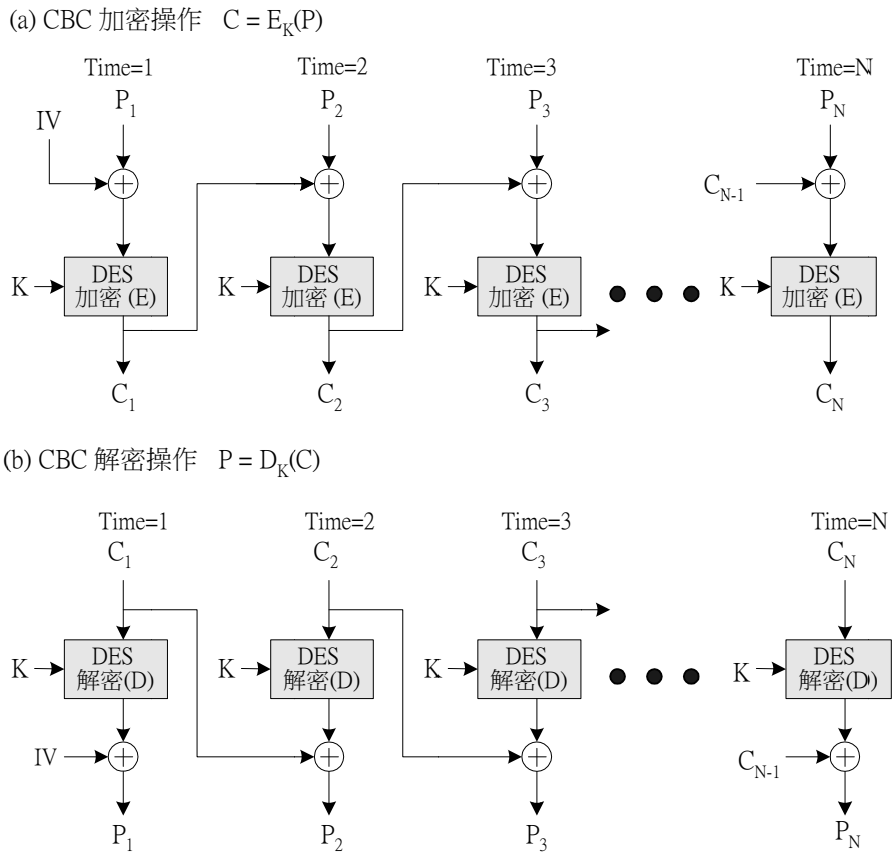


圖 2-21 密文區段串接的操作模式

『起始向量』（IV）是一個很重要的參數，如果雙方連續使用同一把秘密鑰匙，則會增加該鑰匙的曝光率，被破解的機會也相對增加。此外，每次通訊之前，雙方先協議出不同的 IV 數值，也可增加秘密鑰匙被破解的困難度。一般情況，雙方協議 IV 值時，並不需要加密處理，只要用明文傳送即可，破解者雖然知道 IV 內容，但不知道秘密鑰匙（K），依然無法破解，甭說每次傳送的 IV 值都不一樣。

很顯然的，CBC 模式比 ECB 模式增強了許多安全性，尤其在大量資料傳輸時，就不用害怕入侵者用密文比對的方法找出秘密鑰匙。另一個重要的安全性是加入了『起始向量』，可增強秘密鑰匙使用的期限。尤其在多人共享秘密鑰匙時，每一對的每次通訊都採用不同 IV 值，也可增加共享鑰匙的隱密性。譬如，IEEE 802.11 標準便採用此加密方式，來讓多人以共享鑰匙登入系統。另一個重要的應用，就是將 IV 改成『訊息摘要』，以達成『確認性』功能。因此，『密文區段串接』（CBC）是目前使用最普遍的操作模式，許多鑰匙交換協定都採用這種方法。

2-9-3 J-位元密文反饋模式

前面兩種操作模式（ECB 與 CBC）都是屬於區塊加密（Block Cipher）方式，每次輸入加密器（或 DES 演算法）的資料都是以 64 個位元為單位，若輸入訊息不足，則必須將原來訊息補足為 64 位元的倍數。但有許多應用系統並不適合區塊加密方式，而是希望每次以少量，並且是連續性的加密。最明顯的應用是交談式的連線，雙方每次以交談式傳送少量資料給對方，同時希望對方能即時（Real-time）回應。『J-位元密文反饋』（J-bits Cipher Feedback, CFB）模式就是將 DES 區塊（64 個位元區塊）加密的方式，轉換成『串流加密』（Stream Cipher）。CFB 模式每次處理 J 個位元的加密，加密後的密文也是 J 個位元，並將加密的結果串接到下一回合的加密輸入，一般應用上都是採用 8 個位元（J = 8）的加密，這是因為一般資料的編碼都是以 8 個位元方式（如 ASCII 碼）。但為了不去修改原來 DES 編碼演算法，加密與解密的運算還是以 64 個位元為單位。

圖 2-22 為 J-位元密文反饋的運作程序，首先將訊息以每 J 個位元為單位（P_m, m = 1, 2, …, N），一個單位接一個單位連續著進入加密器內處理，每一單位的密文也是 J 個位元（C_m, m = 1, 2, …, N），並以串流方式輸出，且前一回合的密文輸出被導入到下一回合的演算法輸入。CFB 模式的加密程序為：（如圖 2-22 (a) 所示，值得注意的是，加密處理器只有一個，資料連續輸入也僅是重複使用同一處理器）

1. 第一回合為第一筆資料輸入加密器（或演算法），此時『起始向量』（IV，64 bits）已被事先填入『移位暫存器』（Shift Register, SR，64 bits）內。

2. 將移位暫存器的內容輸入到 DES 加密器內，以秘密鑰匙（K）來加密處理，DES 輸出同樣也是 64 bits（$E_K(SR)$）。接下來，擷取 DES 加密器輸出的最高 J 個位元（一般都取 8 個位元），與明文資料 P_m（m = 1, 2, …, n，J 個位元）的每一相對位元之間作 XOR 運算處理，經運算後輸出便是明文資料（P_m）的密文（C_m, m= 1, 2, …, n，J 個位元）；而 DES 加密輸出較低位元的部分（64 – J 個位元）便被捨棄掉了。

3. 當下一筆資料（P_{m+1}）再進入時，加密處理器也進入下一回合的處理。首先將移位暫存器的內容（原 IV 值）向左移位 J 個位元（8 個位元），捨棄較高位元部分，而移位時較低位元則由上一回合的輸出（C_m，J 個位元）補上（此為密文反饋的意思）；接下來回到步驟 2。

4. 如果尚有資料進入，則再進入步驟 3 處理；否則便結束。

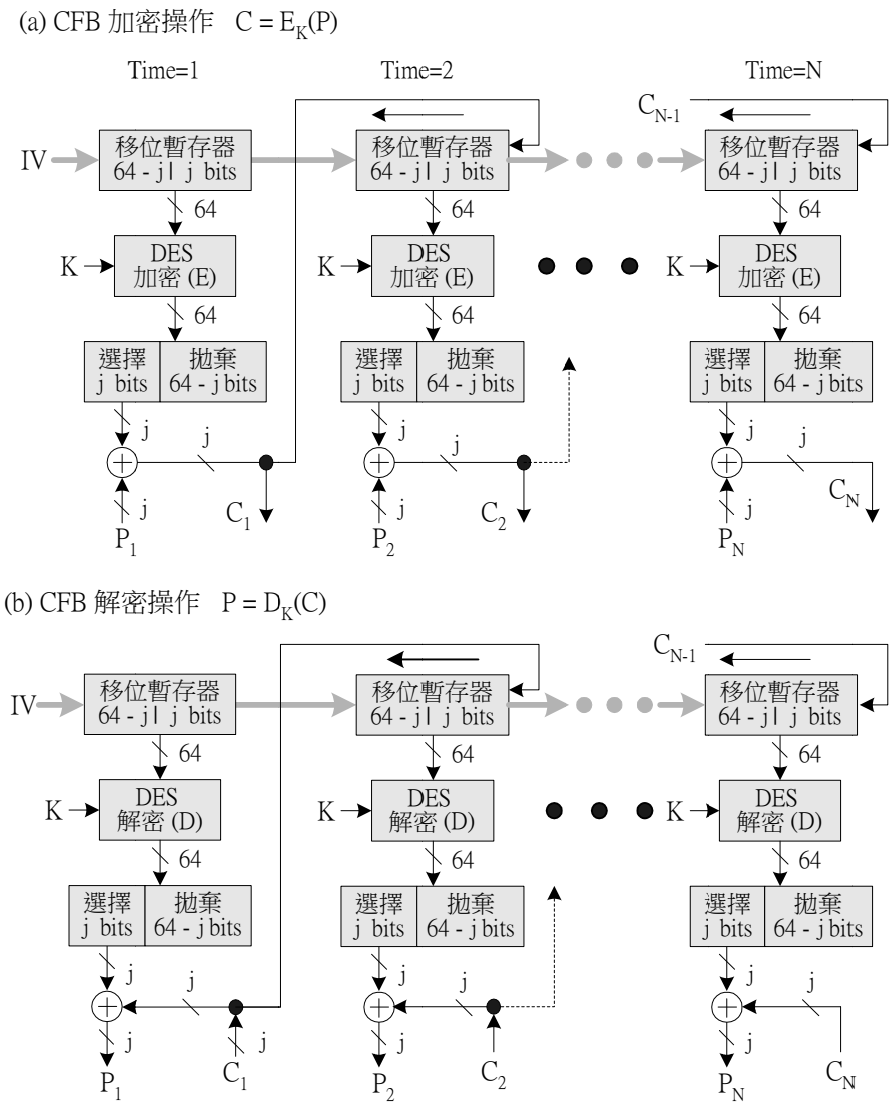


圖 2-22 J-位元密文反饋的操作模式

在加密處理過程中，每一筆明文資料（P_m，J 個位元）輸入後，緊接著輸出它的密文（C_m，J 個位元），屬於一筆接一筆資料的流動方式。因此，CFB 是串流加密（Stream Cipher）法，適合於即時性的傳輸應用。解密的處理程序（如圖 2-22 (b)）也如同加密一樣，不再贅敘。依照圖 2-22 的運算程序，密文與明文的運算結果為：（S_j 表示向

左移位 J 個位元、F_j 表示選擇較高 J 個位元、P_m 與 C_m 皆為 J 個位元、SR 為移位暫存器、"||" 排列的意思）

加密運算程序：

$$SR_1 = IV$$

$$C_1 = F_j\,(E_K(SR_1)) \oplus P_1$$

$$SR_m = S_j\,(SR_{m-1}) \,||\, C_{m-1} \quad ; \quad m = 2, 3, 4, \ldots, N$$

$$C_m = F_j\,(E_K(SR_m)) \oplus P_m \quad ; \quad m = 2, 3, 4, \ldots, N$$

$$C = C_1 \,||\, C_2 \,||\, C_3, \ldots, C_N$$

解密運算程序：

$$SR_1 = IV$$

$$P_1 = F_j\,(D_K(SR_1)) \oplus C_1$$

$$SR_m = S_j\,(SR_{m-1}) \,||\, C_{m-1} \quad ; \quad m = 2, 3, 4, \ldots, N$$

$$P_m = F_j\,(D_K(SR_m)) \oplus C_m \quad ; \quad m = 2, 3, 4, \ldots, N$$

$$P = P_1 \,||\, P_2 \,||\, P_3, \ldots, P_N$$

2-9-4 J-位元輸出反饋模式

『J-位元輸出反饋』（J-bits Output Feedback，OFB）模式與密文反饋（CFB）很相似，唯一不同的是，CFB 模式係反饋加密後的密文，而 OFB 模式是反饋 DES 加密處理後的較高 J 個位元，如圖 2-23 所示。基本上，OFB 與 CFB 模式都是屬於串流加密方式，也適用於即時性的交談式連線，譬如，安全性的 Telnet 連線。但 CFB 模式的密文是經由明文與 DES 加密輸出之間做 XOR 運算後的結果，這裡有一個先天性的缺點，就是如果上一筆資料發生錯誤將導致爾後密文產生連續性的錯誤。交談式連線有一個特點是，連線的時間較長，而每次傳輸的資料較短，因此，在長時間的通訊下，並不能保證傳輸中的資料不會發生錯誤；如果採用 CFB 操作方式，只要有一筆資料發生錯誤，就可能會導致通訊連線中斷。

因此，OFB 模式就是針對 CFB 這方面的缺點來改善，OFB 模式的反饋數值與輸入資料無關，如此可避開錯誤資料的連鎖反應。但因為 OFB 模式的密文與明文之間對應關係，比較容易遭受明文攻擊（如電子密碼書模式），這是它主要的缺點。至於 OFB

模式的運作方式（圖 2-23 (a) 與 (b)），如同 CFB 模式一樣，亦不再另述。我們將加密與解密的運算程序歸類如下：（S_j 表示移位 J 個位元、F_j 表示選擇較高 J 個位元、P_m 與 C_m 皆為 J 個位元、SR 為移位暫存器、"||" 排列的意思），加密運算程序：

$$SR_1 = IV$$
$$O_1 = F_j\,(E_K(SR_1))$$
$$C_1 = P_1 \oplus O_1$$
$$SR_m = S_j\,(SR_{m-1}) \,||\, O_{m-1} \quad ; \quad m = 2, 3, 4, \ldots, N$$
$$O_m = F_j\,(E_K(SR_m)) \quad ; \quad m = 2, 3, 4, \ldots, N$$
$$C_m = O_m \oplus P_m \quad ; \quad m = 2, 3, 4, \ldots, N$$
$$C = C_1 \,||\, C_2 \,||\, C_3, \ldots, C_N$$

解密運算程序：

$$SR_1 = IV$$
$$O_1 = F_j\,(D_K(SR_1))$$
$$P_1 = O_1 \oplus C_1$$
$$SR_m = S_j\,(SR_{m-1}) \,||\, O_{m-1} \quad ; \quad m = 2, 3, 4, \ldots, N$$
$$O_m = F_j\,(D_K(SR_m)) \quad ; \quad m = 2, 3, 4, \ldots, N$$
$$P_m = O_m \oplus C_m \quad ; \quad m = 2, 3, 4, \ldots, N$$
$$P = P_1 \,||\, P_2 \,||\, P_3, \ldots, P_N$$

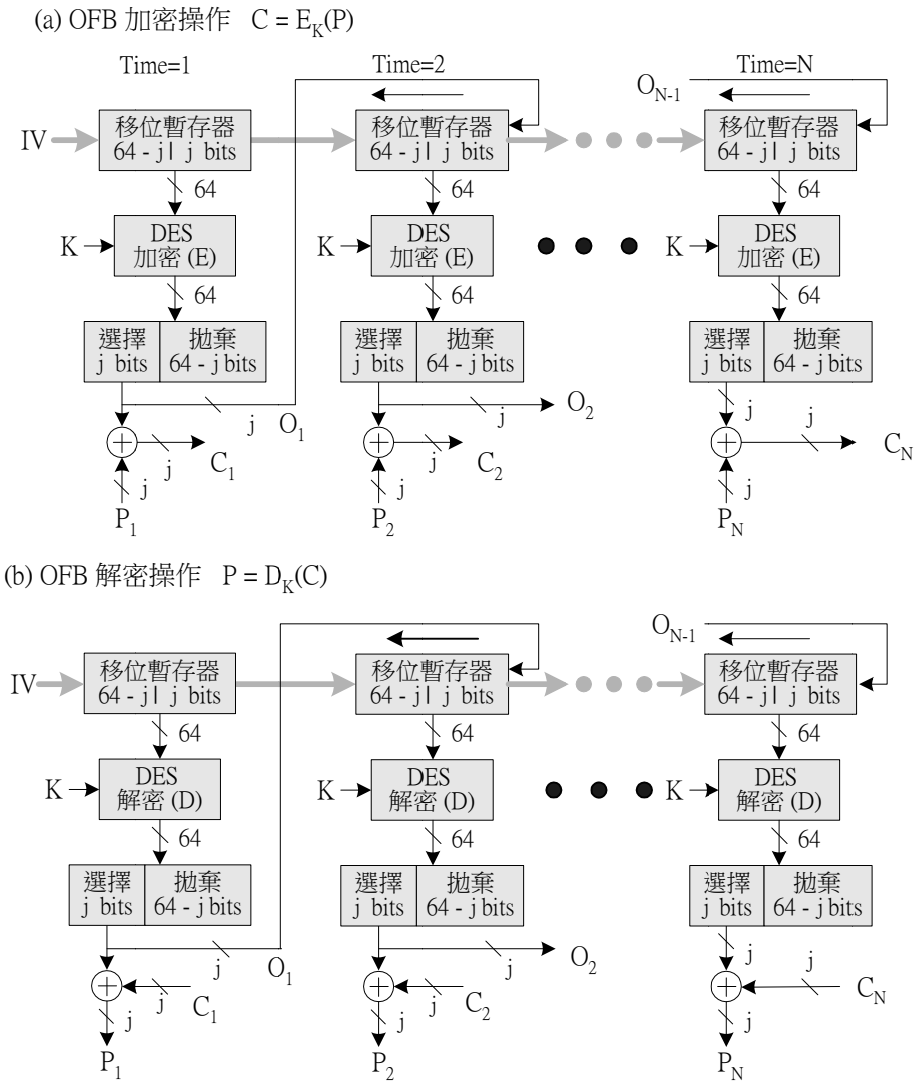


圖 2-23 J-位元輸出反饋的操作模式

2-10 AES 密碼標準

美國 NIST (National Institute of Standards and Technology) 有鑒於 DES 密碼系統在未來幾年內有可能被破解的危機 (許多破解技巧皆是針對 DES 系統而設計的) ，於 1997 年發表公開甄求『進階加密標準』(Advanced Encryption Standard, AES) 的提案。這裡僅簡單介紹 AES 的編碼技術，如讀者有興趣更進一步了解的話，可參考 AES 的官方網站：

http://csrc.nist.gov/encryption/aes

網站內除了有詳細的規範文件可供參考之外，亦有相關原始程式 (Source Program) 可供下載使用，如 AES 加密/解密的原始程式及可執行檔。

首先將 AES 演算法的特點歸類如下：

◈ 區塊長度：原來 Rijndael 演算法是提供可變長度的加密區塊，可任選 128、192、256 個位元區塊，AES 為了簡化其複雜度，只提供 128 位元區塊的加密演算法。

◈ 鑰匙長度：因為鑰匙長度可選擇 128、192、256 個位元長度，因此密碼系統有 AES-128、AES-192、AES-256 的分別。

◈ 編碼演算法：Rijndael 的編碼架構是採用『反覆區段編碼』(Iterated Block Cipher, IBC) 方法。IBC 是將各區塊以陣列格式排列，以位元組 (Byte) 為單位，陣列之間反覆排列與取代來達到編碼的目的。

2-10-1 AES 基本架構

Rijndael 演算法是利用 3 個參數決定加密與解密的處理架構，而其中一個參數是由另兩個參數演變而來的，如下說明：

◈ 明文區段數目 (N_b)：表示 32 bits 加密區段的數目，此參數為輸入的明文區塊可區分為多少個加密區段，以 ASE 標準而言，輸入明文區塊為 128 bits，因此可區分為四個加密區段 ($N_b = 4$)。

◈ 鑰匙區塊數目 (N_k)：此參數為加密鑰匙可區分為多少個鑰匙區段，每一區段的大小也是 32 bits。如 AES-128，則 $N_k = 4$；AES-192，則 $N_k = 6$；而 AES-256，則 $N_k = 8$。

◈ 重覆次數 (N_r)：此參數表示加密 (或解密) 編碼所需重覆的次數。到底需要重覆幾次，這與明文以及鑰匙的複雜度有關，必須取捨兩者之間較複雜的作為重覆次數的依據；也就是說，取明文與鑰匙之間較長者 (或稱區段數目較多者)，作為重覆次數的標準，如此，才能將資料 (明文或鑰匙) 完全的混合，此為 Rijndael 演算法較特殊的地方，計算方式為：$N_r = 6 + \max(Nb, Nr)$。譬如 AES-128，則 $N_r = 10$；AES-192，則 $N_r = 12$；而 AES-256，則 $N_r = 14$；如表 2-1 所示。

表 2-1 Rijndael 參數關係

N_r	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k = 4$	10	12	14
$N_k = 6$	12	12	14
$N_k = 8$	14	14	14

註：明文區塊數 N_b，鑰匙區塊數 N_k，回合次數 N_r

基本上，Rijndael 是利用上述三個參數來建立加密（或解密）編碼的架構，不同的明文區塊與鑰匙長度都會衍生不同的架構。但 NIST 只採用鑰匙長度為 128、192 與 256 位元，並規定明文區塊長度為 128 位元（$N_b = 4$），因此產生了 AES-128、AES-192 與 AES-256 等三種標準規範，表 3-2 為 NIST 就這三種規範所制定的相關參數值。圖 2-11 為 AES-128 的基本架構，其中輸入明文區塊為 128 bits（AES 標準），鑰匙長度為 128 bits（AES-128 標準）的加密處理架構；其中 $N_k = 4$、$N_b = 4$、以及 $N_r = 10$。

表 2-2 AES 三種標準規範

	明文區塊 N_b	鑰匙區塊 N_k	回合次數 N_r
AES-128	4	4	10
AES-192	4	6	12
AES-256	4	8	14

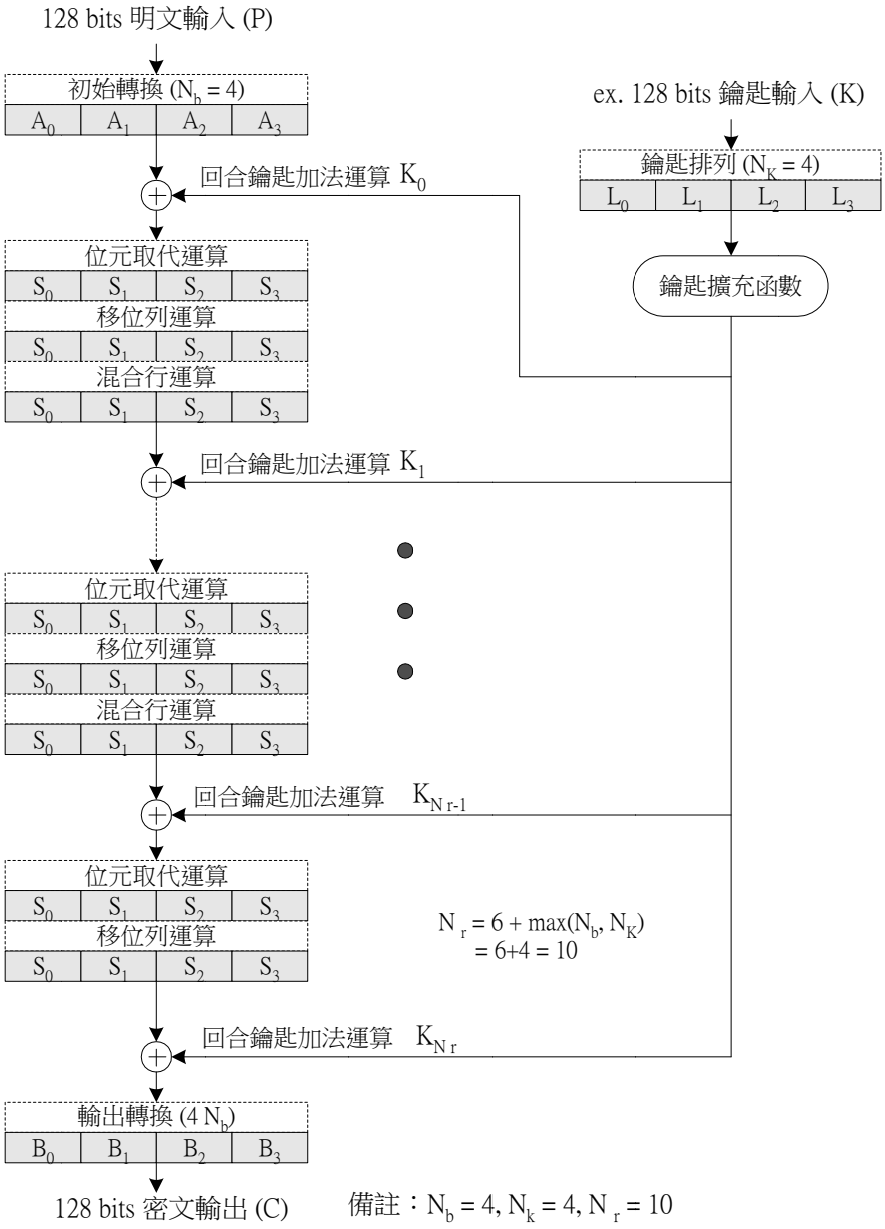


圖 2-11 AES-128 加密演算法之架構

圖 2-11 為 AES-128 加密演算法的架構，其它 AES 系統亦屬雷同，甚至解密演算法的基本架構亦是依照此模型製作。值得注意的是，AES 演算法已不再沿用 Festel 的基本架構，Rijndael 提出一套數學模型（如 $GF(2^8)$）推導出其加密與解密演算法。為使讀者能對 AES 有較通盤的瞭解，在介紹其演算法之前，有必要先就其數學基礎作簡單的說明。

2-10-2 AES 加密演算法

依照 AES 標準規範，明文區塊限制於 128 bits，也就是說，N_b 固定為 4(如表 3-2 所示），又各種規範（AES-128、AES-192、AES-256）之間不同的是 N_k（鑰匙長度）與 Nr（回合次數，$N_r = 6 + \max(N_b, Nr)$）的數值。圖 3-11 為 AES 加密演算法的虛擬碼（請參考圖 3-9），其中包含下列四個主要函數：

- 回合鑰匙加法運算：AddRoundKey()
- 位元組取代運算：SubBytes()
- 列移位運算：ShiftRows()
- 混合行運算：MixColumns()

```
Cipher (byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
        /* in 為輸入陣列、out = 輸出陣列、w = 鑰匙字元陣列 */
Begin
    Byte state[4, Nb]
    /* 明文陣列複製到狀態陣列上 */
    state = in
    /* 第 0 回合編碼 */
    AddRoundKey(state, w[0, Nb-1])
    /* 第 1 到 Nr – 1 回合編碼 */
    for round = 1 step 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
    end for
    /* 第 Nr 回合編碼 */
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
    /* 密文陣列輸出 */
    out = state
end
```

圖 2-12 AES 加密演算法

針對圖 3-11 AES 加密演算法說明如下：

(1) 首先將明文區塊(in 陣列，128 bits)依照圖 3-10 方式填入狀態陣列(state)之中。

(2) 狀態陣列與子鑰匙做相加運算（ AddRoundKey()，容後說明 ）。

(3) 接下來，執行回合加密，每一回合執行編碼處理的順序為 SubBytes()、ShiftRows()、MixColumns()，以及 AddRoundKey()，至於執行的回合次數則依照不同規範而定；以 AES-128 (Nb = 4、Nk = 4)為例，回合次數為 10(計算為 $N_r = 6 + \max(N_b, N_r) = 6 + \max(4, 4) = 10$)，但在迴圈中只執行 9 次（ $= N_r - 1$ ）。

(4) 回合加密後，緊接著再執行 SubBytes()、ShiftRows()、以及 AddRoundKey()，最後再將狀態陣列 (state) 填入輸出陣列中 (如圖 3-10 所示)，並完成該區塊的加密處理；另外，密文輸出如同明文一樣，皆是 128 bits 長度。

圖 3-11 中可以發現兩個重點，一者為進入回合加密之前，先執行一次 AddRoundKey() 函數 (為了方便說明，將其稱為第 0 回合編碼)，因此子鑰匙的數目必須是回合次數加一 (即是 $N_r + 1$)；另一者為最後編碼回合 (第 N_r 回合) 並沒有執行 MixColumns() 函數，因此，沒有在迴圈裡運算。以下介紹上述中四個加密函數，以及鑰匙擴充函數。

2-11 Triple DES 密碼系統

56 位元的 DES 編碼系統已面臨被暴力攻擊破解的危機，如何在現有的系統上來增加它的安全性，『三重 DES』(Triple DES，3DES) 編碼系統就在此情況下被發展出來 [139]。它有兩種系統架構，以下分別說明之。

2-11-1 Two-Keys 3DES

簡單的說，Triple DES 系統是連續使用 DES 編碼器三次，如只用到兩把 56 個位元的鑰匙 (112 bits)，則稱為 Two-keys 3DES (如圖 3-2 所示)。加密時，明文由 DES-加密、DES-解密、DES 加密連續編碼三次得到密文，兩把鑰匙 K_1 與 K_2 交叉使用；解密運作是，密文經 DES-加密、DES-解密、DES 加密連續碼三次得到明文，使用與加密時相同的兩把鑰匙。計算公式如下：

加密：$C = E_{K1}[D_{K2}[E_{K1}[P]]]$

解密：$P = D_{K1}[E_{K2}[D_{K1}[C]]]$

鑰匙：$K = K_1 \| K_2$ （排列組合）

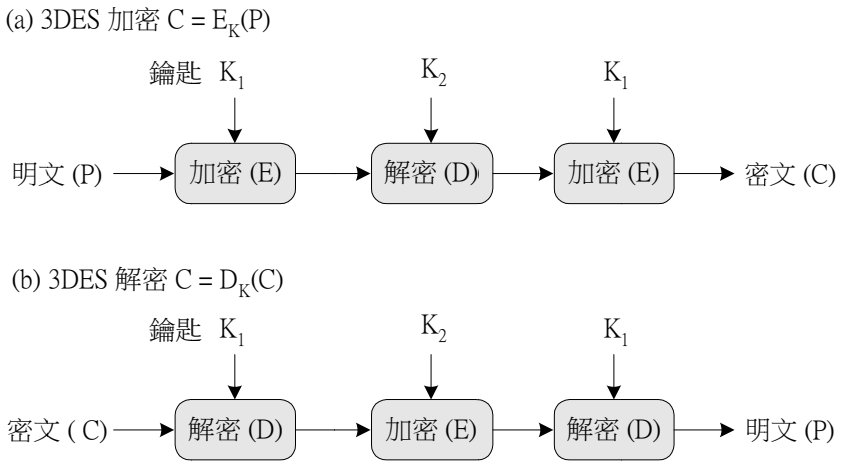


圖 2-13 3DES 加密的運作程序

同樣的演算法，利用不同的鑰匙加密兩次以上，是否可以達到雙倍以上的安全性，實在是有待商榷的問題。對付這種加密法的破解方法為『中間交會攻擊法』(Meet-in-the-middle Attack) [56]，首先來觀察以兩把鑰匙執行同一演算法的密文輸出：

$C = E_{k2}[E_{k1}[P]]$

令：

$X = E_{k1}[P] = D_{K2}[C]$

假設已經知道（P，C）的組合（選擇明文攻擊），首先，將排列可能出現的 2^{56} 個 K_1 對 P 加密，並建立一個組合表格（P，X）；接下來，再排列出 2^{56} 個 K_2 對 X 加密的可能組合（X，C）。可以發現，如果 2^{56} 機率的編碼系統可以破解的話，雙重加密系統的安全度並沒有提高多少，祇不過增加一點程序而已。

克服『中點交會攻擊法』最基本的方法是增加加密器的串接，而採用三階段三鑰匙的加密法，可以大大增加『選擇明文攻擊』的困難度，雖然仍有可能被破解，但所花耗的成本也相對增加許多。如果真是如此，鑰匙長度將會延伸到 168 個位元(56 * 3)，似乎有點過於龐大。Triple DES 採用『加密 – 解密 – 加密』的串接方式，只要兩把

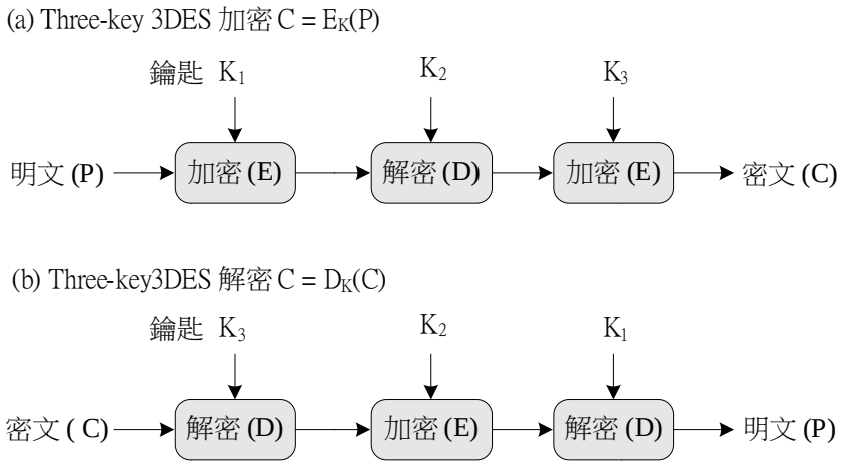
鑰匙即可，鑰匙長度只有 112 個位元（56 * 2），同樣可以得到三重加密的效果（如圖 3-2 所示）。目前使用兩把鑰匙的 Triple DES 加密法，非常受工業界歡迎，並且已成為 ANS 與 ISO 的標準規範：ANS X9.17 與 ISO 8732。

2-11-2 Three-keys 3DES

但話說回來，還是採用三把鑰匙三重加密較為安全(強度較高)[81]，鑰匙長度 168 個位元，組合方式為：$K = K_1 \| K_2 \| K_3$，其定義如下：（系統架構與圖 3-2 相似）

加密：$C = E_{K3}[D_{K2}[E_{K1}[P]]]$

解密：$P = D_{K1}[E_{K2}[D_{K3}[C]]]$



目前有許多網路應用系統採用三把鑰匙的 Triple DES 加密法，如 PGP 或 S/MIME 等系統皆是。有趣的是，使用 Triple DES 密碼系統並沒有違背早期美國政府的出口限制，因為它還是採用 56 位元 DES 的密碼系統，祇不過重複使用 3 次而已。美國政府只限制幾位元密碼系統的出口，並沒有限制幾位元長度的鑰匙。由此可見，如果認為 AES（128 位元）不夠安全的話，也可試著使用 Triple-AES 來試看看。

2-12 密碼系統的安全性

2-12-1 密碼系統的真實情況

一般而言，密碼系統常被誤認為僅提供資料的隱密性功能，如此就太小看密碼學的威力了。欲了解密碼學的功能，應從『鑰匙』（Key，或稱金鑰、金匙、關鍵）的觀念

來談。鑰匙擁有者通常有加密或解密的權利，因此非特殊身分者不可擁有鑰匙。由此可見，鑰匙是身分的表徵，其功能就如『印章』一般，文件經由『用印』之後，便成為有效的公文。相同的，一份文件經由某人使用鑰匙加密之後，就表示此人對此文件負責，該文件縱使非他本人所發出的，至少他經手審查過。依此觀念密碼學才會有：隱密性、確認性、完整性與不可否認性的功能。

常見密碼系統可將所欲傳送的明文加密成為一些沒有意義的亂碼，這一串亂碼便稱為『密文』，只有擁有解碼鑰匙者方可將密文解譯成明文。但密文在網路上傳輸被盜取之後，果真無法解密回明文嗎？這是值得商榷的問題。在未探討這個問題之前，首先我們來了解密碼系統在網路上運作的一些真實情況：

1. 密碼演算法必須是公開的：在網路上並不可以確定或限制通訊的對象，尤其是電子商務方面，多半與陌生人通訊，並無法私底下告知對方某一特定的密碼演算法。更進一步，我們期望在網路上建立共通標準的密碼演算法規範，所以演算法必須是公開的，何況為了提高演算法的處理速度，已有許多演算法被嵌入硬體晶片之內。

2. 明文格式是無法隱藏的：一般來講，某些特殊用途的明文格式都有一定的規範，如稽核檔案、採購清單、信用卡號碼等等。針對某一格式的明文經過編碼加密之後，非常有可能出現一些相似的地方，入侵者經由比對這些相似的地方即可找出破解的方法。更嚴重的，若入侵者可發送一份明文要求傳送端加密的話，也可取得相對應的密文，如此不難找出明文與密文之間的關聯性。

3. 密文是唾手可得的：密文在網路上傳輸，任何人都可輕易取得該資料。如果入侵者有意破解某一系統的話，只要不斷地收集該系統所發出的密文，由同一把鑰匙所加密的密文愈多，則被破解的機率也相對應提昇。

由此可見，密碼系統並非完全可靠的。甚至可以大膽直言，沒有不可能被破解的密碼系統，祇是破解難易程度不同而已。在密碼學中，將密碼系統的防禦被破解能力稱為『強度』（Strength）。即是『強度』愈高的密碼系統，愈難破解。另外，最起碼可以

肯定的是，一把秘密鑰匙絕不可以使用過久，太久的話，終究會被破解的，甚至要求一把鑰匙只能使用一次（容後介紹）。

2-11-2 密碼系統的計算上安全

密碼系統被破解的容易度，關係著該演算法的複雜度，越複雜的演算法所需的計算時間相對越高。再說為了抗拒暴力攻擊法，最基本的方法是增加鑰匙的長度。簡單的說，任何密碼演算法都有被破解的可能，不管鑰匙的長度如何，都有可能被暴力搜尋出來，關鍵在於破解密碼是否合乎下列兩種條件：

◈ 破解密碼所需的成本是否合乎該訊息的價值。

◈ 破解密碼所需的時間是否超過該鑰匙的壽命。

如果能克服上述兩個條件的密碼系統，便稱之為『計算上的安全』(Computationally Secure)；密碼系統安全與否的衡量標準在於破解者需要多少時間、花費多少成本才能破解密碼。以 DES 密碼系統為例，鑰匙長度為 56 個位元，破解者可能必須嘗試 2^{56} 把鑰匙來暴力攻擊；可能鑰匙的總數為 $2^{56} = 7.2 * 10^{16}$，如果每微秒 (us) 加密一次 (執行 DES 演算法)，則需要 2^{56} us = 1142 年的計算時間才可以成功；預估未來 10 年電腦的整合處理速度將可以達到每微秒 10^6 次，如此只要 10.01 小時便可測試完畢，而平均只要 5 小時便可以成功。因此，可預估得到 DES 密碼系統已不再是安全了，此癥結在於 DES 系統不夠複雜，演算法太容易被計算出來。唯有發展更複雜的加密演算法，來拖延電腦的計算時間，或是增加鑰匙長度，來增加嘗試鑰匙的數量，才能達到『計算上的安全』。

2-11-3 演算法的複雜度

我們將密碼系統抗拒破解的能力，稱為該系統的『強度』(Strength)。如果一個密碼系統的強度很高的話，則表示較不易被破解，然而破解方法也許是已知明文或已知密文方法。一般來講，破解技巧大多採用『統計分析破解法』，方法是取出密文與明文、或密文與鑰匙之間的關聯性，再以統計分析的方法找出明文或鑰匙；所以對抗統計分析破解法的最根本技巧，便是增加演算法的複雜度，要達成這個目標可由兩方面來看：

◈ 『**混淆**』（**Confusion**）：是指密文與鑰匙之間的關聯性。混淆程度越高，則表示密文與鑰匙之間的關聯性越複雜。也就是說，同一把鑰匙加密多筆明文後，所產生的密文之間的相似性越少越好，如此一來，較困難由許多密文及明文配對推導出加密鑰匙。

◈ 『**擴散**』（**Diffusion**）：是指明文與密文之間的關聯性。擴散程度越高，則表示明文與密文之間的關聯性越複雜。也就是說，同一筆明文經過多把鑰匙加密後的密文，所產生的密文之間的相似性越少越好，如此一來，較困難破解出加密演算法。

由此可見，『混淆』與『擴散』能力，就是評估兩個（或兩個以上）明文之間的內容大致上相同，但各自所產生密文之間的差異度。如果差異很大的話，則混淆與擴散能力就愈強，相對其密碼演算法的『強度』就愈高。混淆與擴散能力夠強的密碼系統，較能抵擋『統計分析破解法』。一般破解法大多採用『選擇明文攻擊』。亦是，首先由中間人選擇某些明文發送給傳送端，藉由對方傳送過來的相對應的密文。若所選擇的明文有一定規則的格式，並且已加入一些特殊的文字標誌。此時，混淆能力不足的演算法，所產生的密文就會發生一定規則的變化。如此，破解者就可以利用統計分析的方法找出加密鑰匙。所以比較理想的系統，縱然兩個明文之間只有一個字元的變化，所產生的密文也要有相當大的差異，因此稱之為『混淆』與『擴散』的能力。

2-13 密碼破解技巧

不經加密鑰匙或是使用偽造鑰匙，便能夠將密文解譯回原來的明文，稱之為『密碼破解』（Cryptanalysis）。值得注意的是，密碼系統的應用中，密碼演算法是公開的，所有機密性的問題完全在於『鑰匙』，如果鑰匙被盜取或被偽造的話，則密碼系統的所有安全性將會完全崩潰。破解鑰匙最基本的是『統計分析破解法』與『暴力破解法』（Brute-Force Attack）兩種方法。前者是統計分析的方法，找出已知明文與密文之間的相依性，再利用此相依性分解出所欲破解密文的相對應明文。暴力破解法即是嘗試鑰匙所有可能出現的情況，來破解密碼系統。因為密碼系統演算法是公開的，且加密後密文是唾手可得的。吾人可利用已知密文與加密演算法，嘗試所有可能出現的鑰匙，如果能

將密文解密出合乎正常規格的文件，即是找出通訊鑰匙，再利用此鑰匙盜取其他通訊內容。由此可見，鑰匙長度愈長者，所需嘗試破解的數目相對就越多，如此可以增加鑰匙被破解的困難度(延長其破解時間)。如果一把鑰匙使用太久的話，絕對有可能被破解。

首先，我們將一般密碼破解方法歸類如下：

- 只知密文破解：破解者蒐集所有可能攔截到的密文，由這些密文之中比對其相似的地方，將密文解譯出明文。只由密文直接破解實在很困難，因為破解者持有的資訊太少。但如果連這樣也可以破解的話，該密碼演算法實在愚不可及。
- 已知明文破解：破解者使用已知明文與其相對應的密文，來推演出其加密鑰匙。採用『暴力攻擊法』大多達到此破解法。
- 選擇明文破解：這是非常可怕的破解方法，攻擊者可利用特殊方法(如反射攻擊)，將明文發送給傳送端，再由傳送端取得加密後的密文。攻擊者可選擇各種明文格式 (如稽核檔、資料記錄檔、文書檔案) ，要求對方加密。再比對同一資料格式加密的各種型態，便可輕易的推演出它的加密鑰匙。
- 選擇密文破解：如同選擇明文破解一樣，將密文傳送給接收端，再由接收端得到解密後的明文；如果資訊充足的話，也可以比對出秘密鑰匙。

前面兩種破解方法大多採用暴力攻擊法；後面兩種大多使用統計分析破解法。至於暴力攻擊法是最簡單也最可怕的攻擊技巧，它嘗試所有可能發生的秘密鑰匙，來攻擊密碼系統。譬如，DES 演算法的鑰匙長度為 56 個位元，是 2^{56} 之中的一種機會，攻擊者連續使用 2^{56} 個鑰匙來嘗試解出明文。克服暴力攻擊最簡單的方法是，資料在加密之前，先經過壓縮處理。如此的話，攻擊者無論破解成功與否，都無法看出有意義的資料，而不知道是否破解成功。但如此是否能有效防止被破解，也是值得懷疑的，因為壓縮演算法也可能需要公開。