致理技術學院

資 訊 網 路 技 術 系

實 務 專 題 報 告

對稱式加解密之實作

指導教師：蕭勝華

學生：周東諺(29534503)

黃琮聖(29534513)

江旭涵(29534520)

張堉程(29534526)

中華民國 96 年 12 月

致理技術學院

資 訊 網 路 技 術 系
實 務 專 題 報 告

指導教師：蕭勝華

學生：周東諺(29534503)

黃琮聖(29534513)

江旭涵(29534520)

張堉程(29534526)

本成果報告書經審查及口試合格特此證明。

指導老師: _____

中華民國 96 年 12 月

專題研究授權書

本授權書所授權之專題研究為_____

_____

共_____人，在致理技術學院資訊網路技術系 _____學年度第_____學期完成資網實務專題。

專題名稱：_____

□同意　□不同意

本組同學共_____人，皆同意著作財產權之論文全文資料，授予教育部指定送繳之圖書館及本人畢業學校圖書館，為學術研究之目的以各種方法重製，或為上述目的再授權他人以各種方法重製，

不限地域與時間，惟每人以一份為限。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。上述同意與不同意之欄位若未勾選，該組同學皆同意視同授權。

指導教師姓名:

專題學生簽名:　　學號:

中華民國　　年　　月　　日

誌 謝

首先，要感謝我們的指導老師蕭勝華老師，因為在蕭老師的悉心及耐心的指導之下，專題才得以順利的完成，也感謝老師在我們專題遇到困難時，給予許多指導及督促，也促使我們在專題的研究上進步許多，幸蒙尊師教誨，讓我獲益良多。

誠摯的感謝指導蕭勝華老師，老師悉心的教導使我得以一窺加解密管理領域的深奧，學習之路永無止境，恩師教化永銘於心。，感謝高楊達老師不時的討論並指點我正確的方向，使我在專題中獲益匪淺。使我學會word世界是多麼的深奧老師對學問的嚴謹更是我輩學習的典範,蒙高楊老師督促鼓勵，學生銘感於心。

感謝林正平老師，使我在寫作程式方面有了更進一步的功力師之教、記於心，師之情、長相憶。感謝陳文雄老師讓我學會的如何使用將程式放至ＰＤＡ裡面，如何使用 PDA 想當初筆連在哪都找不到，得師匡正之心，行必嚴謹敦行，老師的諄諄教誨，使我們獲益良多。

感謝不厭其煩的指出我研究中的缺失，且總能在我迷惘時為我解惑，也感謝阿凱、皓塵、柏維、祖文、茂智、修維、星豪、大炳、鈞皓、茶茶、小啾同學的幫忙，恭喜我們順利走過這兩年。

摘 要

近年來拜無線網路的技術發達所賜，不管是筆記型電腦(Note Book)、手機或是 PDA(Personal Digital Assistant)就連最近正火熱的掌上遊戲機 PSP(PlaySation® Portable)與 NDSL(Nintendo® DS Lite)都具有行動上網的能力，只要能收得到訊號的地方，幾乎都能自由自在的暢遊網路，也因此有越來越多的上班族不需要到所謂的辦公室，只要有能上網的行動裝置以及基地台就能隨時隨地的辦公。為了讓私密的個人檔案或是公司的機密文件不容易被惡意的第三方擷取破解，所以我們決定在行動裝置上設計一個加解密的程式來防止文件被擷取破解。

本專題著重於設計一個使用於行動裝置上的檔案加解密程式，首先先探討主要使用兩個演算法 AES(Advance Encryption Standard)和 DES(Data Encryption Standard)之技術發展與差異，由於 PDA 較相似於電腦所以選擇 PDA 當作開發環境並使用 C++來開發加密解密程式。

目　　錄

圖　目　錄

表　目　錄

第一章　緒論

第一節 重要性與發展演進

(一)、加密解密法的演進

在近代以前，經典加密法只考慮到訊息的機密性（confidentiality）：如何將易於理解的訊息轉換成難以理解的訊息，並且使得有秘密訊息的人能夠逆向回復，而缺乏秘密訊息的攔截者或竊聽者則無法解讀。近數十年來，這個領域已經擴展到涵蓋身分認證（或稱鑒權）、訊息完整性檢查、數位簽章、互動證明、安全多方計算等各類技術。

其實在西元前，秘密書信已用於戰爭之中。西洋「史學之父」希羅多德（Herodotus）的《歷史》（The Histories）當中記載了一些最早的秘密書信故事。西元前 5 世紀，希臘城邦為對抗奴役和侵略，與波斯發生多次衝突和戰爭。於西元前 480 年，波斯秘密集結了強大的軍隊，準備對雅典（Athens）和斯巴達（Sparta）發動一次突襲。希臘人狄馬拉圖斯（Demaratus）在波斯的蘇薩城（Susa）裏看到了這次集結，便利用了一層蠟把木板上的字遮蓋住，送往並告知了希臘人波斯的圖謀。最後，波斯海軍覆沒於雅典附近的沙拉米斯灣（Salamis Bay）。

由於古時多數人並不識字，最早的秘密書寫的形式只用到紙筆或等同物品，隨著識字率提高，就開始需要真正的密碼學了。最古典的兩個加密技巧是：

置換（Transposition cipher）：將字母順序重新排列，例如『help me』變成『ehpl em』；

替代（Substitution cipher）：有系統地將一組字母換成其他字母或符號，例如『fly at once』變成『gmz bu podf』（每個字母用下一個字母取代）。

這兩種單純的方式都不足以提供足夠的機密性。凱撒密碼（Caesar Cipher）是最經典的替代法，據傳由古羅馬帝國的皇帝凱撒所發明，用在與遠方將領的通訊上，每個字母被往後位移三個字母所取代。

加密旨在確保通訊的秘密性，例如間諜、軍事將領、外交人員間的通訊，同時也有宗教上的應用。舉例來說，早期基督徒使用密碼學模糊他們寫作的部份觀點以避免遭受迫害。666 或部分更早期的手稿上的 616 是新約基督經啟示錄所指的野獸的數字，常用來暗指迫害基督徒的古羅馬皇帝尼祿（Nero）。史上也有部份希伯來文密碼的記載。古印度愛經中也提及愛侶可利用密碼來通信。隱寫術（steganography）也出現在古代，希羅多德記載將訊息刺青在奴隸的頭皮上，較近代的隱寫術使用隱形墨水、縮影術（microdots）或數位浮水印來隱藏訊息。

由經典加密法產生的密文很容易洩漏關於明文的統計資訊，以現代觀點其實很容易被破解。

阿拉伯人津帝（al-Kindi）便提及到如果要破解加密資訊，可在一篇至少一頁長的文章中數算出每個字母出現的頻率，在加密信件中也數算出每個符號的頻率，然後互相對換，這是頻率分析的前身，此後幾乎所有此類的密碼都會馬上被破解。但經典密碼學至今仍未消失，經常出現在謎語之中。這種分析法除了被用在破解密碼法外，也常用於考古學上。在破解古埃及象形文字（Hieroglyphs）時便運用了這種解密法。

中世紀至第二次世界大戰的期間裡，所有的密碼仍然受到上述的破密法的危害，直到阿伯提（Leon Battista Alberti）約在 1467 年發明了多字元加密法（polyalphabetic cipher），阿伯提的創新在於對訊息的不同部分使用不同的代碼，他同時也發明了可能是第一個自動加密器，一個實現他部分想法的轉輪。多字元加密法最典型的例子是維瓊內爾加密法（Vigenere cipher）：加密重複使用到一個關鍵字（key word），用哪個字母取代端視輪替到關鍵字的哪個字母而定。

許多物理裝置被用來輔助加密，例如古希臘斯巴達的密碼棒（scytale），這是一個協助置換法的圓柱體，可將資訊內字母的次序調動，利用了字條纏繞木棒的方式，把字母進行位移，收信人要使用相同直徑的

木棒才能得到還原的資訊。在歐洲中世紀時期，密碼欄（cipher grille）用在某類隱寫術上。多字元加密法出現後，更多樣的輔助工具出現，如阿伯提發明的密碼盤（cipher disk）、特裡特米烏斯發明的表格法（tabula recta）、以及美國總統湯瑪士傑佛遜（Thomas Jefferson）發明的多圓柱。

廿世紀早期，多項加解密機械被發明且被註冊專利，包括最有名的轉輪機（rotor machines），第二次世界大戰德軍所用，別名『謎』（Enigma machine），其加密法是在第一次世界大戰後針對當時破密術所做最好的設計。

第二次世界大戰後計算機與電子學的發展促成了更複雜的密碼，而且計算機可以加密任何二進位形式的資料，不再限於書寫的文字，以語言學為基礎的破密術因此失效。多數計算機加密的特色是在二進位字串上操作，而不像經典密碼學那樣直接地作用在傳統字母數字上。

然而，計算機同時也促進了破密分析的發展，抵消了某些加密法的優勢。不過，優良的加密法仍保持領先，通常好的加密法都相當有效率（快速且使用少量資源），而破解它需要許多級數以上的資源，使得破密變得不可行。

雖然頻率分析是很有效的技巧，實際上加密法通常還是有用的。不使用頻率分析來破解一個訊息需要知道目前是使用何種加密法，因此才會促

成了諜報、賄賂、竊盜或背叛等行為。直到十九世紀學者們才體認到加密法的演算法並非理智或實在的防護。

實際上，適當的密碼學機制（包含加解密法）應該保持安全，即使敵人知道了使用何種演算法。對好的加密法來說，金鑰的秘密性理應足以保障資料的機密性。這個原則首先由奧古斯特。

柯克霍夫（en:Auguste Kerckhoffs）提出並被稱為柯克霍夫原則（Kerckhoffs』 principle）。資訊理論始祖香農（Claude Shannon）重述：「敵人知道系統。」

大量的公開學術研究出現，是現代的事，這起源於一九七零年代中期，美國國家標準局（en:National Bureau of Standards, NBS；現稱國家標準枝術研究所，en:National Institute of Standards and Technology, NIST）制定數位加密標準（DES），Diffie 和 Hellman 提出的開創性論文，以及公開釋出 RSA。

從那個時期開始，密碼學成為通訊、電腦網路、電腦安全等上的重要工具。許多現代的密碼技術的基礎依賴於特定基算問題的困難度，例如因數分解問題或是離散對數問題。許多密碼技術可被證明為只要特定的計算問題無法被有效的解出的話，那就比較安全。除了一個著名的例外：一次

安全密碼（en:One-Time Password，OTP），這類證明是偶然的而非決定性的，但是是目前可用的最好的方式。

密碼學演算法與系統設計者不但要留意密碼學的歷史，而且必須考慮到未來發展。例如：持續增加計算機處理速度會增進暴力攻擊法（brute-force attacks）的速度。量子計算的潛在效應已經是部份密碼學家的焦點。

二十世紀早期的加密解密法本質上主要考慮語言學上的模式。從此之後重心轉移，現在密碼學運用大量的數學，包括資訊理論、計算複雜性理論、統計學、組合學、抽象代數以及數論。

密碼學同時也是工程學的分支，但卻是與別不同，因為它必須面對有智能且惡意的對手，大部分其他的工程僅需處理無惡意的自然力量。檢視密碼學問題與量子物理間的關聯也是目前熱們的研究。

第二節 研究動機與目的

(一)、研究動機

近年來無線網路的技術發達所賜，不管是筆記型電腦(Note Book)、手機或是 PDA(Personal Digital Assistant)就連最近正熱們的掌上遊戲機 PSP(PlaySation® Portable)[1]與 NDSL(Nintendo® DS Lite) [2]都具有行動上網的能力，只要能收的到訊號的地方，幾乎都能自由自在的暢遊網路。

因此有越來越多的上班族不需要到所謂的辦公室，只要有能上網的行動裝置以及基地台就能隨時隨地的辦公；為了讓私密的個人檔案或是公司的機密文件不容易被第三方惡意的擷取破解，所以我們決定在行動裝置上設計一個加解密的程式。

(二)、研究目的

由於 PDA 與個人電腦有較相近的操作環境所以我們選擇了 PDA 當我們實作的環境，由於 AES(Advance Encryption Standard)具有下列特性：(1)對抗所有已知的攻擊；(2)能在各種平台上快速執行，並具備簡潔的程式碼；(3)設計簡單易懂；而 DES(Data Encryption Standard)雖然已經逐漸被 AES 所取代，但直到目前為止還是被廣泛的應用，可見此演算法之重要性。

因此我們使用了 DES 與 AES 這兩種最具代表性的加密解密演算法來設計一個適用於 PDA 上操作的系統。

第二章　理論與技術探討

第一節 資料加密原理簡介

密碼學(Cryptography),它是如何讓資訊達到密碼性的科學。目前的密碼系統可分為兩大類型(如圖 1):

(一)、對稱型金鑰密碼系統(Symmetric key Cryptosystem)，與非對稱型金鑰密碼系統(Asymmetric key Curptosystem)。

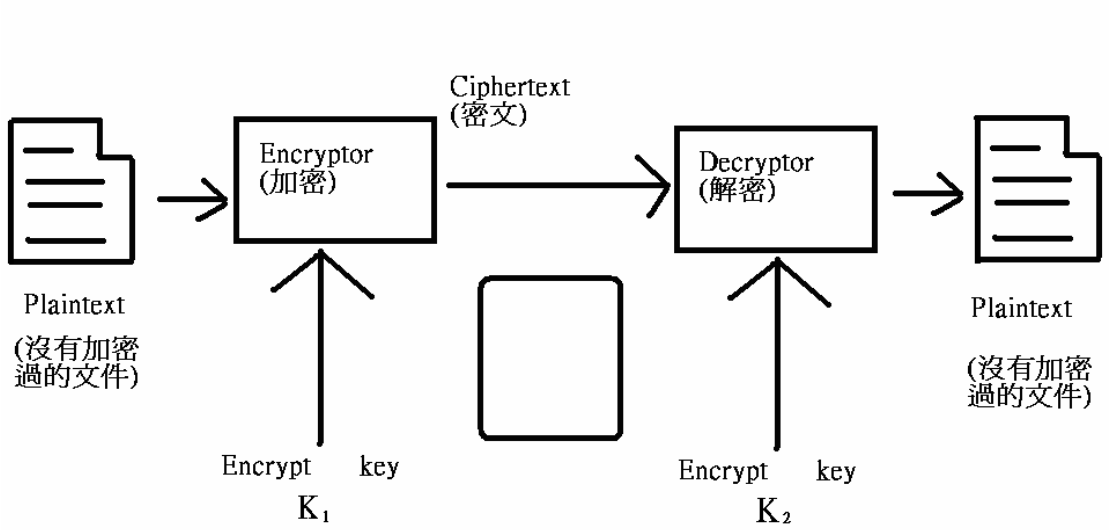


圖 2-1、從典型密碼系統區分對稱型與非對稱型金鑰密碼系統圖

對稱型金鑰密碼系統的特點是加密金鑰與解密金鑰是相同的。而其優點是處理的時間較短，計算所需的記憶體不用太大。

然而它也有下列缺點：(1)收發雙方如何獲得金鑰呢？若雙方互不認識時，此問題更嚴重。(2)金鑰所需要的數目太大，若網路上有m人，則每個人必須擁有（m－1）把金鑰，而網路中心需有m（m－1)／2 把金鑰，如何管理那麼多的金鑰也是一大難事。(3)由於發送方A與接收方B都知道雙方的金鑰，因此發送方A可在送過訊息後否認他有送過，若接收方B他有竄改或偽照A送過的訊息，第三者C若沒有求證是無法分辨是A抵賴送過的訊息，或是B自己捏照有收過的。

非對稱型金鑰密碼系統又稱為公開金鑰密碼系統，他的最主要精神就是當存在一對加密與解密金鑰K_1與K_2，公開K_1，但只有一個人會知道K_2。任何人都可以利用K_1加密，但是只有知道K_2的接送者才能解密。公開金鑰密碼系統雖然具有許多優點，但仍有其缺點存在。最受人詬病之點，即在於加解密運算複雜且速度慢。

需注意的是，密碼學中所討論的系統或是演算法，其”安全性”是最高的準則。一個再好的加解密系統，若其演算法不足以抵抗攻擊則就是一文不值了。所以本論文中所挑選的系統其安全性皆以受嚴格的考驗而存留下來。接下來，在下面兩小節中，我們將介紹本論文所選用的對稱型金鑰密碼系統－ＡＥＳ（rijndael）,以及非對稱型密碼系統－ＲＳＡ（Rivest__Shamir__Adleman）。

第二節 對稱型金鑰密碼系統

美國自 1977 年以來，所用之非機密資料加密標準ＤＥＳ（Data Encryption Standard）其加密鑰匙只有 56 位元，這對現在具高速計算處理能力之電腦而言，已顯得過短，以致ＤＥＳ之安全性被質疑多年。基於此因，美國標準技術研究所（en：National Institute of Standard and Technology，ＮＩＳＴ） 自 1997 年起計畫公開徵求先進加密標準ＡＥＳ（Advanced Encryption Standard）以取代 DES，終於在２０００年１０月經ＮＩＳＴ宣佈ＡＥＳ獲選的演算法為 Rijndael。

當各因素同時考量時，Rijndael 在安全、性能、效率之整體性結合上比較其他演算法較佳，且易於操作與彈性，這促使 Rijndael 成為ＡＥＳ最適合之選擇。

而ＡＥＳ是一個類似ＤＥＳ反覆運算的加解密演算法，它允許可變動的資料區塊及金鑰的長度。資料區塊與金鑰長度的變動是各自獨立的，這是與ＤＥＳ相比所沒有的。它在各種不同的硬體或軟體計算環境下，皆表現的非常出色。它的鑰匙設定時間是最短的。而且因為計算所需之記憶體空間很少，非常適合於有記憶體大小、執行速度限制之環境，如ＩＣ卡，而經由測試也確實顯示出其優秀性能。

所以把它來當作新一代對稱型加解密演算法再安全、性能、效率之整體性來看是名副其實的。在安全性的評估上，由於ＡＥＳ之鑰匙可使用三種長度：１６、２４、３２位元組。與ＤＥＳ相比，ＤＥＳ鑰匙長度７位元組，所以ＡＥＳ　１２８位元鑰匙數約為ＤＥＳ　５６位元鑰匙數之４.７＊１０^{21}倍。以上對ＡＥＳ做一簡要介紹之後，我們統整ＡＥＳ的優點與限制。

(一) 、ＡＥＳ有以下優點：

（1）ＡＥＳ可以實作在 586 等級以上之電腦，並經測試後仍有相當快的速度處理運算。

（2）ＡＥＳ可以實作在空間限制型之環境，例如智慧卡（Smart Card）上，使用少量的記憶體，少量的程式碼，在ＲＯＭ與效率之間也是可以做取捨的。

（3）加密法不採用算術運算，不會因為不同處理器架構而有所偏差而導致不同。

（4）設計上不像ＤＥＳ需引用其他加密文件，如Ｓ－ｂｏｘ。

（5）加密法緊湊，不易藏入暗們（Trap Door）等程式碼，如ＤＥＳ的Ｓ－ｂｏｘ。

（二）、ＡＥＳ的限制：

(1) 實作在智慧卡時，解密不如加密來得有效率，解密需要更多的程式碼及計算時間。

(2) 以軟體而言，加密和解密需使用不同的程式，這將花費記憶體空間去記錄這些程式。

(3) 以硬體而言，解密只能重用部份加密的電路，這將花費硬體的空間和能量的消耗。

第三節 非對稱型金鑰密碼系統

在公鑰密碼系統中，每個使用者都擁有兩把鑰匙－公開金鑰（public key）與私密金鑰(private key)公開金鑰公步給所有人知道，誰都可以用，私密金鑰則是由自己秘密保存者。這兩個金鑰之間存在者相互依存關係，即用其中任一個金鑰加密的信息只能用另一個密鑰進行解密。若以公鑰做為加密金鑰，以用戶專用金鑰（私鑰）作為解密金鑰，則可實現多個用戶加密的信息只能由一個用戶解讀，反之以用戶私鑰作為加密金鑰而以公鑰作為解密金鑰，則可實現由一個用戶加密的信息而多個用戶解讀。

以圖 2-2 來說，若 A 方想要把訊息送給 B 方且不想讓其它人知道訊息的內容，A 方可以用 B 方的公鑰將訊息加密後送出，加密過的訊息只有擁有金鑰的人才能解開讀到真正的內容。在正常的情形下，只有 B 方一人知道這把解密的鑰匙。

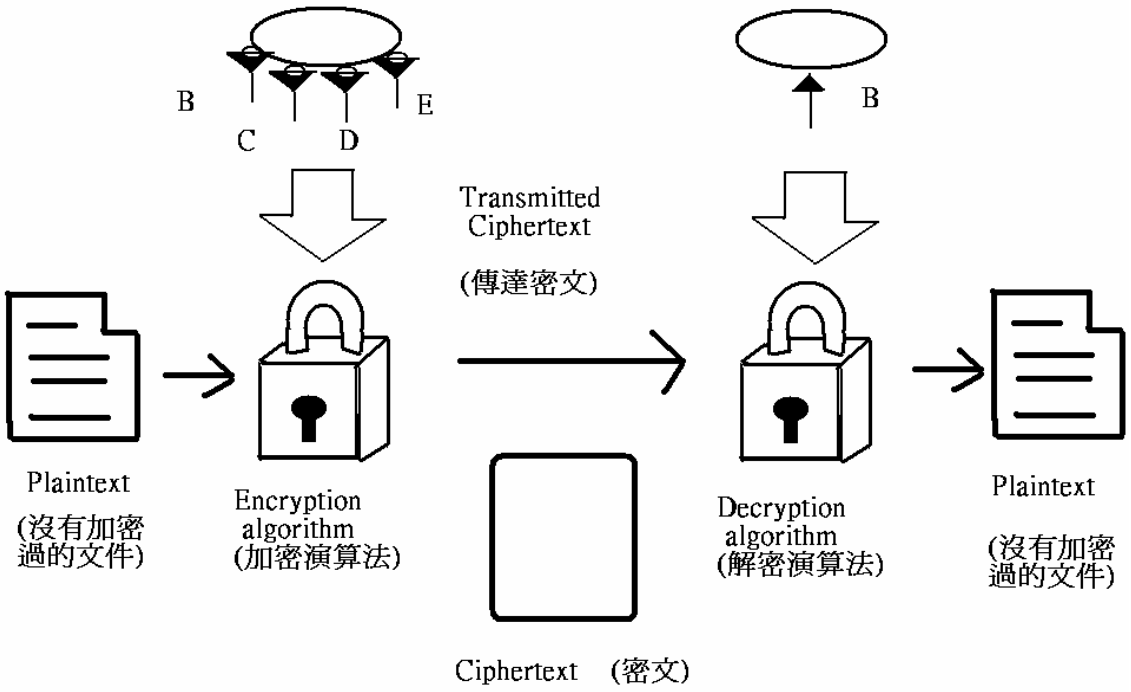


圖 2-2、ＲＳＡ：公鑰與金鑰示意圖

ＲＳＡ密碼系統是第一個將安全性質基於於分解質因數的困難程度上的加解密系統，它產生了兩個大的質數，“幾乎”無法由公鑰計算推導出私鑰，而也就是靠這種計算不可逆性才得以保密並確保其安全性。假如藉由計算推導確實不易取得私鑰，那麼我們可以稱這系統是安全的。因為大家公認去分解一個大個數值是相當困難且花費時間的，所以說ＲＳＡ系統可說是相當安全的。

表 1、對稱與非對稱金鑰密碼學之間的比較表

特性	對稱型密碼學	非對稱型金鑰密碼學
用來加密＼解密的金鑰	加密和解密使用同樣的金鑰	加密使用一把金鑰，解使用另一把不同的金鑰
加密＼解密速度	非常快	慢
密文的大小	通常與原始內文大小相同或更少	超過原始內文大小
金鑰協議＼交換	許多問題	沒有問題
在訊息交換中須要的金鑰數與參與者比較使用	金鑰數約等於參與者的平方，所以可擴充，是一個問題通常使用在加密與解密(機密性)無法應用在數位簽章上，(缺少完整性及不可否認性)	金鑰數與參與者相擴充不是問題可以使用在加密、解密(機密性)以及數位簽章(完整性和不可否認性)

第四節　ＡＥＳ演算法

(一)、ＡＥＳ的數學背景

在 Rijndael 的演算法中，使用了相當多的位元組的運算，而這些運算是以 GF(2^8)為基礎架構。另外在金鑰擴展的部份也使用了一些字組的運算。對於這些運算所使用到的基礎數學原理，在以下的部份會做個簡單的介紹。

(二)、ＧＦ（2^8）的定義

有限場是一群元素的組合，而元素的數目是有限的。而 Byte 在 AES 中是最基本的運算單元，對於一個位元組 b，我們可以視為由 8 個元素所組成，且係數不是 0 就是 1，可以以{b7,b6,b5,b4,b3,b2,b1,b0}的形式來表示。

ＧＦ（2^8）的表示方法有很多種，我們可以將上述的Ｂ述的 B_i

想像成一個 7 次多項式的係數，以多項式的形式來表示:

$$B_7X^7+B_6X^6+B_5X^5+B_4X^4+B_3X^3+B_2X^2+B_1X+B_0 = \sum_{i=0}^{7} b_i x^i \qquad (1)$$

例如，｛01010111｝多項式為表示法為 $X^6+X^4+X^2+X+1$。此外位元組亦可以１６制形式來表示，故｛010101111｝也可標記為{37}。

一般而言，ＡＥＳ是使用位元組作為運算單元，但是在位元組乘法運作操作上，若遇上溢位情形發生，則模餘一個事先定義的８次多項式，其二進制表示為９個 bit。

對於這額外的的位元 b_8，以及 16 進制標記為{01}，置於另 8 個位元左側。例如一個 9bits 的序列(100011011) ，以 16 進制表示為{01}{1b}。

第五節 Ｓｔａｔｅ 二維陣列

在ＡＥＳ演算法中，明文與密文被視為一連串的二進位位元流，在加解密處理時，我們會將之表示為位元組的陣列型式。例如一個１２８bit 的 input 序列如下所示:

$Input_0$ $input_1$ $input_2$ $input_3$・・・・・$input_{126}$ $input_{127}$

這些 iuput 序列，可以形成共具 16 個 Byte 的陣列如下:

a_0 a_1 a_2 a_3 a_4・・・・・a_{14} a_{15}

其中

a_0 ＝ {$input_0$ $input_1$ ・・・・・$input_7$}；

a_1 ＝ {$input_8$ $input_9$ ・・・・・$input_{15}$}；

・

a_{15} ＝{$input_{120}$ $input_{121}$・・・・・ $input_{127}$}；

至於其他不同長度的序列，例如ＡＥＳ＿１９２與ＡＥＳ＿２５６的金鑰長度，依照同樣的方法，我們可以用下列的通式來表示之：

$a_n = \{input_{8n}\ input_{8n+1} \cdots\cdots\ input_{8n+7}\}$；　　(2)

在金鑰擴展時，也會使用到字組(32bit)的運算，因此對於位元、位元組、字組的相對的索引編碼，在表 2 與表 3 更一目了然。

表 2 、byte 與 bit 的索引編碼表

Bit sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	…
Byte number	0								1								2								…
Bit mumbers in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	…

表 3 、words 與 bytes 的索引編碼表

Byte numbers	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
word number	0				1				2				3				…
Byte mumbers in word	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	…

ＡＥＳ的運算，主要是一個二維的 byte 型態矩形陣列中執行，此矩陣稱為 State，所存放的內容為資料區塊，包括加解密前、運算中或加解密後的資料，其大小在 Rijndae 加密演算法中定義為 4 X Nb，即 state 的列數 r 的範圍為 0<=r <4，行數 c 的範圍為 0<=c <Nb，因此 Rijndael Algorithm 是允許可變動長度的資料區塊。

但在ＡＥＳ標準中，只允許三種不同長度的加密金鑰，但對於資料區塊的參數 Nb，則以固定 Nb = 4，故 State 的大小即為 4x4 的矩陣。State 輸入與輸出的操作如圖 2-3 所示

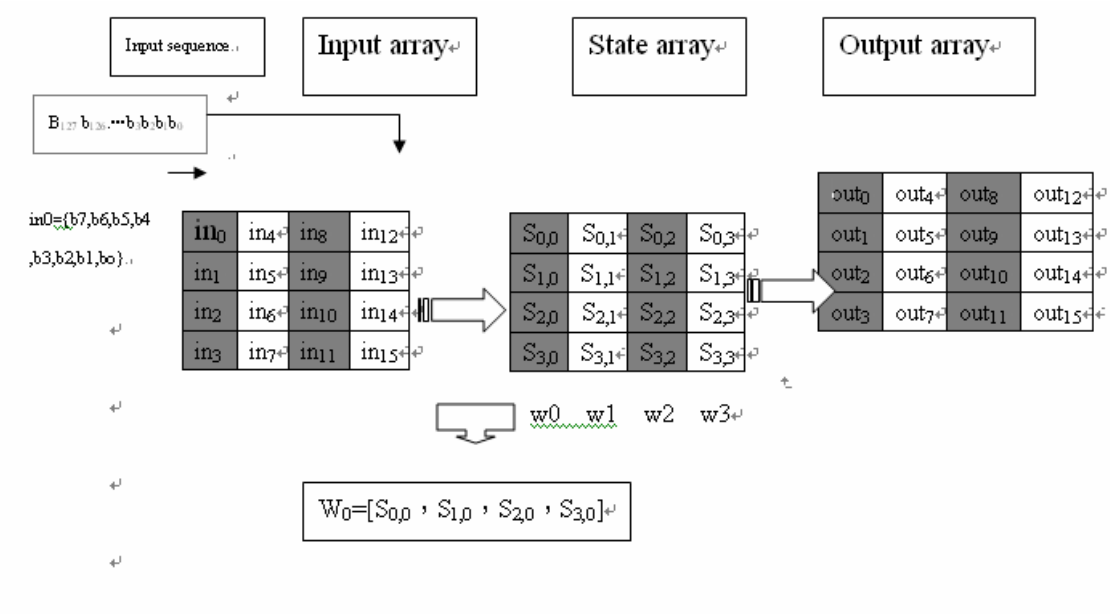


圖 2-3、State 陣列的輸入與輸出圖

當 State 進行字組運算時，如同圖 2-3 所描述，State 的每一行都視為一個字組(word) ，可以表示如下:

$W_0=[S_{0,0}，S_{1,0}，S_{2,0}，S_{3,0}]$　　$W_1=[S_{0,1}，S_{1,1}，S_{2,1}，S_{3,1}]$

$W_2=[S_{0,2}，S_{1,2}，S_{2,2}，S_{3,2}]$　　$W_3=[S_{0,3}，S_{1,3}，S_{2,3}，S_{3,3}]$　　(3)

若字組的運算以多項式的型態來表示，則與位元組的定義稍有不同，以 W_0 為例的四次多項式表示為:

$$W_0(x)=S_{0,3}x^3，S_{0,2}x^2，S_{0,1}\ x，S_{0,0} \qquad (4)$$

在(4)式的多項式的每一項係數皆是 GF(2^8)有限場的形式。

xtime 函式

在ＧＦ（2^8）的多項式乘法中，會碰上溢位的問題，也要用到除法的運算，故在ＡＥＳ演算法中，使用了ＧＦ（2^8）的多項式乘以ｘ次冪，所產生的特性，使得乘法運算得以簡化為單存的 shift 動作或有溢位產生時的 shift+ＸＯＲ的動作。對於這項特性，在ＡＥＳ中標記為 xtime()。

這樣的結果，相當於位元組 b={b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0}作左移一個 bit 的動作後，在與｛01｝｛1b｝(十六進制)作 XOR 的動作。也可以這樣表示：

x • b(x)＝＞｛00000010｝•{1 b_6 b_5 b_4 b_3 b_2 b_1 b_0}

＝　（｛00000001｝｛b_6 b_5 b_4 b_3 b_2 b_1 b_0 0｝）⊕

({00000001})({00011011})

＝（｛b_6 b_5 b_4 b_3 b_2 b_1 b_0 0｝）⊕　{00011011}

(一)、若位元組 b 中的最高位元 b_7 為 0，則 xtime(b)等於位元組 b 左移一個 bit，且最低位元補 0。

(二)、若位元組 b 中的最高位元 b_7 為 1，則 xtime(b)等於位元組 b 左移一個 bit，且最低位元補 0，接著再與{1b}作 bit-XOR 的動作，更細部的說只有最低位元與 b_0， b_2， b_3的部分需作反相的動作。

第五節 ＡＥＳ演算法描述

ＡＥＳ演算法主要運算的部份，是執行特定數目的加密或解密迴圈，不同的金鑰長度，有不同的回合數，詳細的情形如表 4:

表 4、金鑰長度-資料區塊-回合數對照表

項目	ＡＥＳ_128	ＡＥＳ_192	ＡＥＳ_256
金鑰長度(Nk 字組)	4	6	8
資料區塊大小(Nb 字組)	4	4	4
執行回合數(Nr)	１０	１２	１４

在 AES 中，主要可分為 Cipher、Inverse Cipher 及 Key Expansion 三大部分，分別由以下三個子節來說明。

（一）加密區塊（Cipher）

整個 Cipher 的演算流程如圖 2-4。

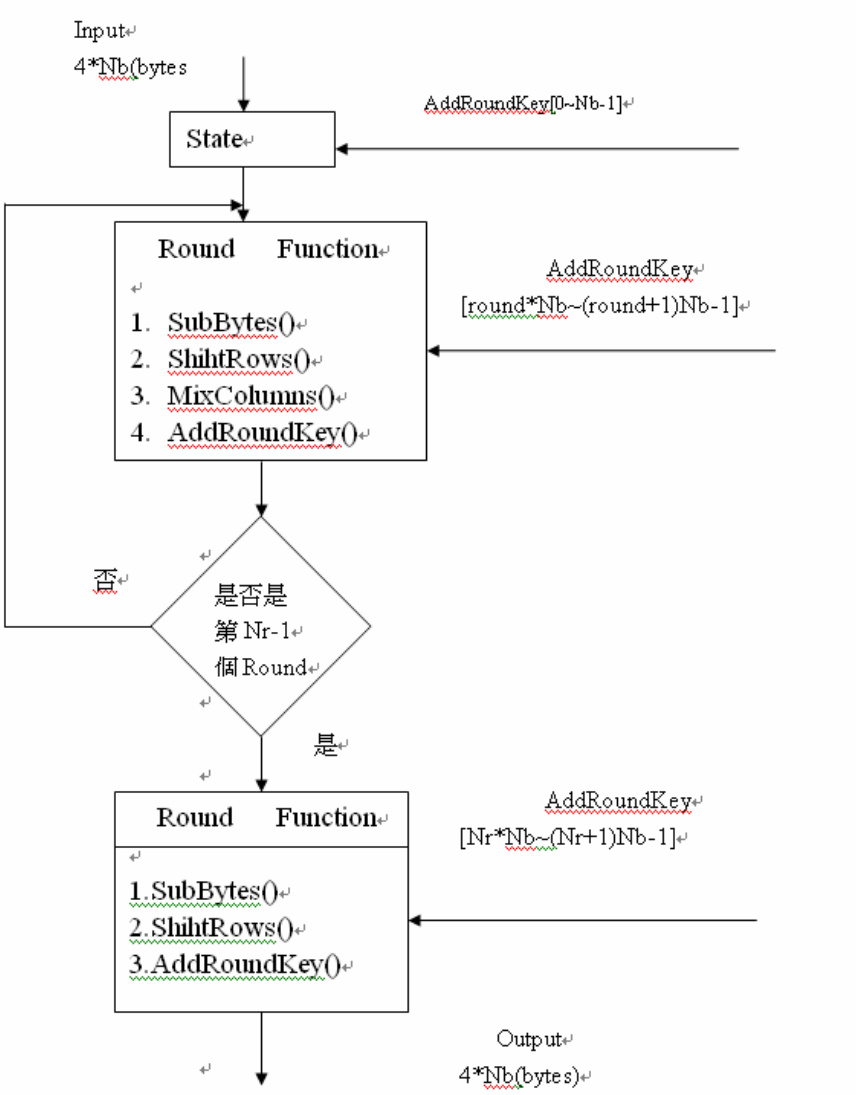


圖 2-4、 Cipher 的演算流程圖

Cipher 中每一回合的運算，除最後一回合外，對於資料區塊皆作了四種函式的轉換操作，分別為

(1)SubBytes()(2)ShiftRows()(3)MixColumns()(4)AddRoundKey()，其中每一回合有不同的回合金鑰稱為 RoundKey，而 RoundKeyg 是由 Key Expansion 所產生的。

SubBytes()是進行非線性的位元組置換，主要是使用置換表(S-box)，使用 S-box 的過程是可逆的，故存在 Inverse S-box，使之所置換的資料在 Inverse Cipher 中可以得到回覆。

※ S-box 原理:

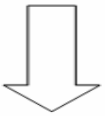
S-box 是 16x16 的矩陣，每個元素的內容均為(2^8)位元組，其形成主要基於兩個步驟的轉換。

step1:尋找乘法反元素

一個原始尚未進行任何轉換的矩陣定義如表 5

表 5 、建構 S-box 的初始矩陣表

y / x	1	2	3	⋯	16
1	0	1	2	⋯	15
2	16	17	18	⋯	31
3	32	33	34	⋯	47
⋮	⋮	⋮	⋮	⋱	⋮
16	240	241	242	⋯	255



y / x	1	2	3	⋯	f
1	00	01	02	⋯	0f
2	10	11	12	⋯	1f
3	20	21	22	⋯	2f
⋮	⋮	⋮	⋮	⋱	⋮
f	F0	F1	F2	⋯	ff

接下來要替表 6 的每一個 GF(2^8)位元組尋找其乘法反元素。假設有一位元組多項式 a(x)，則其乘法反元素為 $a^{-1}(x)$，滿足

$$a(x) \bullet a^{-1}(x) = 1 \qquad (5)$$

由(5)式可以為表 6 中所有的元素找出所有的乘法反元素，其中定義 $\{00\}_{16}$ 的乘法反元素為其本身。找出乘法反元素之後的 S-box 矩陣，如表 6

表 6 、建構中的 S-box(找出乘法反元素)表

x \ y	1	2	3	…	f
1	00	01	8d	…	c7
2	74	b4	Aa	…	b2
3	3a	6e	5a	…	c2
⋮	⋮	⋮	⋮	⋱	⋮
f	5b	23	38	…	1c

Step2:仿射轉換

仿射轉換是屬於 bit(ＧＦ（２）)的操作，轉換的公式如下所示：

$$b_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \qquad (6\text{-}1)$$

其中 0≦i＜8， i 代表位元組的第 i 個 bit。至於位元組 c 則為ＡＥＳ所事先定義，其值為 c ＝｛63｝$_{16}$={01100011}$_2$。

式子(6-2)是一個很固定的操作模式，故也可以化作矩陣的形式：

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad (6\text{-}2)$$

經過 S-box 的兩個建構步驟後，我們可以得到如表 7 的 S-box。在使用上是利用位元組的高四位元代表索引 x，低四位元代表索引 y，來做位元組置換的工作。

表、7 S-box[16]表

y / x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	F2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	A1	89	0d	bf	e6	42	68	41	99	2d	0f	B0	54	Bb	16

至於 State 利用 S-box 對於 SubBytes()的操作相當的簡單易懂，圖 2-5 說明了 State 應用 SubBytes 函式的行為模式。

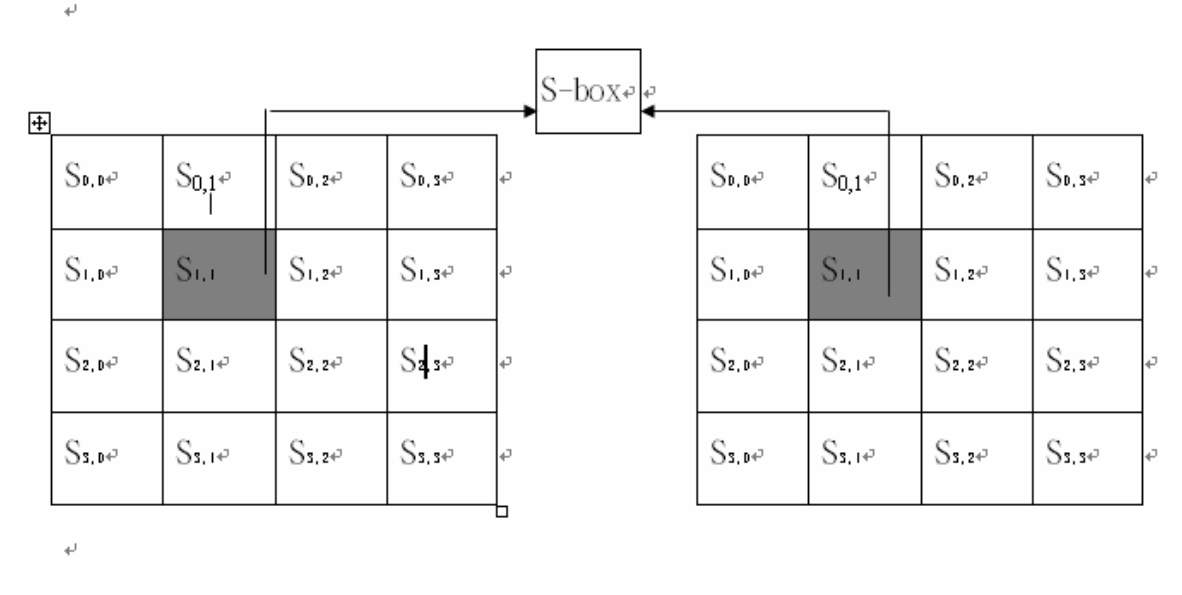


圖 2-5 、State 操作 SubByte 函式圖

(3) MixColumns 函式轉換

MixColumns 是在加密過程中，唯一使用到字組運算的地方，此函式是將 State 中每一 Column 的內容 s(x)作混合，而得到新的 column 內容 s =a(x)⊕s(x)。此處所定義的多向式為

$$a(x)=\{03\}x^3+\{01\}x^2+\{01\}x+\{02\} \quad (7)$$

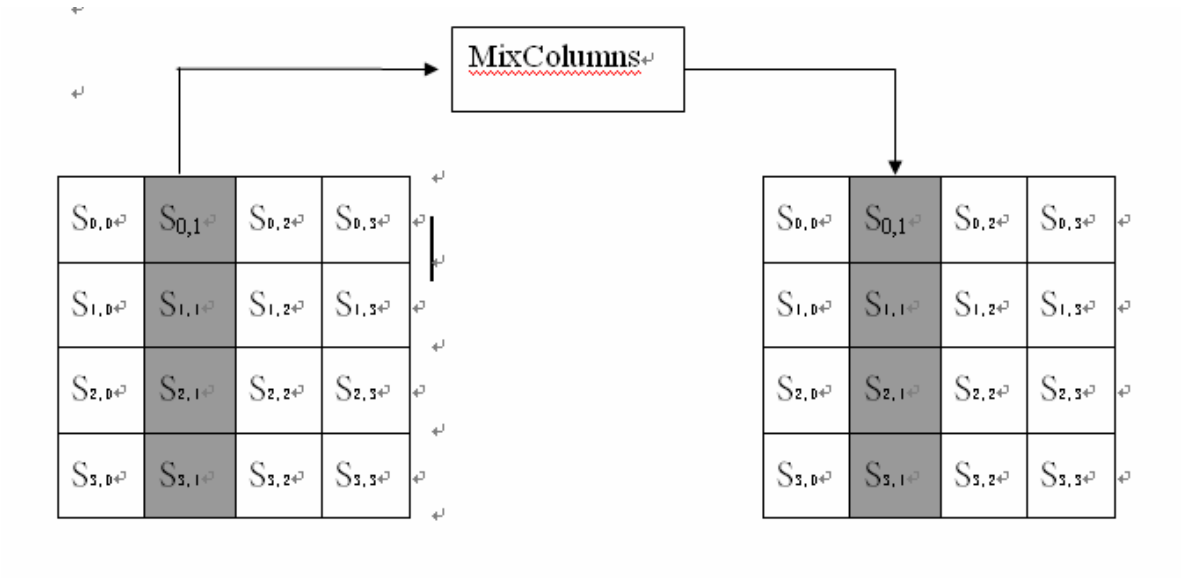


圖 2-6、說明了 State 應用 MixColumns 函式的行為模式圖

(3) AddRoundKey 函式轉換

每一回合的金鑰 RoundKey 包含了 Nb 個字組，在 AddRoundKey 函式轉換下，回合金鑰 RoundKey 是和 State 執行 bitwise ＸＯＲ的操作，使得

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] =$$

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] \oplus$$

$$[W_{round*Nb+c} \text{ for } 0 \leqq c < Nb \qquad (8)$$

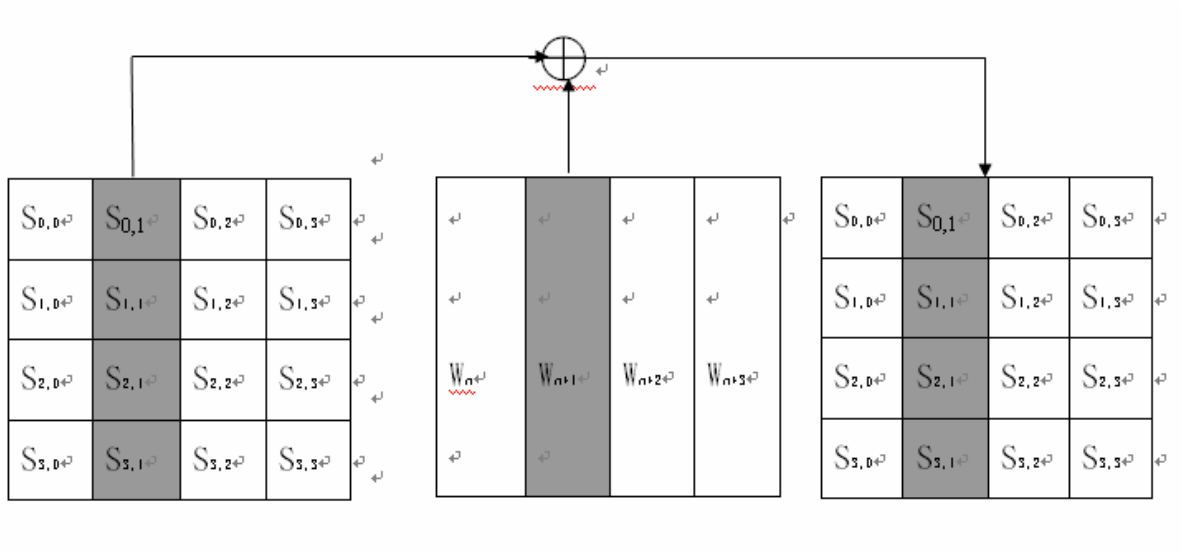


圖 2-7、 State 操作 AddRoundKey 函式圖

(4) MixColumns 函式轉換

原始的金鑰長度為 Nk 個字組，為了增加安全度，使ＡＥＳ每個回合利用不同的金鑰來做加解密的動作，故定義了一套金鑰擴展為一個 4-byte 的線性陣列，標記為 W[i] ，其中 0≦i＜Nb(Nr+1) ，因此金鑰擴展後共有 Nb(Nr+1)個字組。

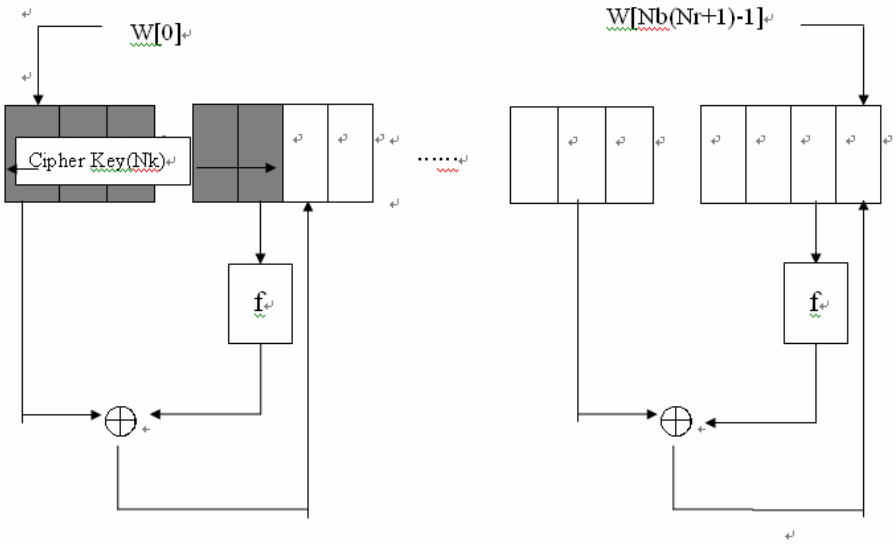


圖 2-8、 金鑰擴展的示意圖

由圖 2-8 可以知道回合金鑰的產生大致上大同小異，唯一不同的地方是圖 2-8 中的 f 代表函式運算的組合，不同位置的回合金鑰，其操作的行為就不一樣，圖 8 顯示實際的流程。而 f 所代表的函式組合包含了三種函式，分別為 SubWord()、RotWord()及 XOR Rcon[i]。

SubWord 函式：

SubWord 函式的操作如同 SubBytes 函式一樣，也是利用 S-box 來做非線性位元組置換，只是 SubWord 函式一次是做四個位元組的置換。

RotWord 函式：

RotWord 函式是傳回一個經過旋轉的字組，例如一個字組為 $[a_0,a_1,a_2,a_3]$ ，經過 RotWord 函式處理後，其回傳字組為$[a_1,a_2,a_3,a_0]$。

Rcon[i]函式：

Rcon[i]稱為回合常數，因為是用於字組的 bitwise XOR的操作，所以它的長度為一個字組，內容為〔x^{i-1},{00},{00},{00}]，其中 i 為從 1 開始。

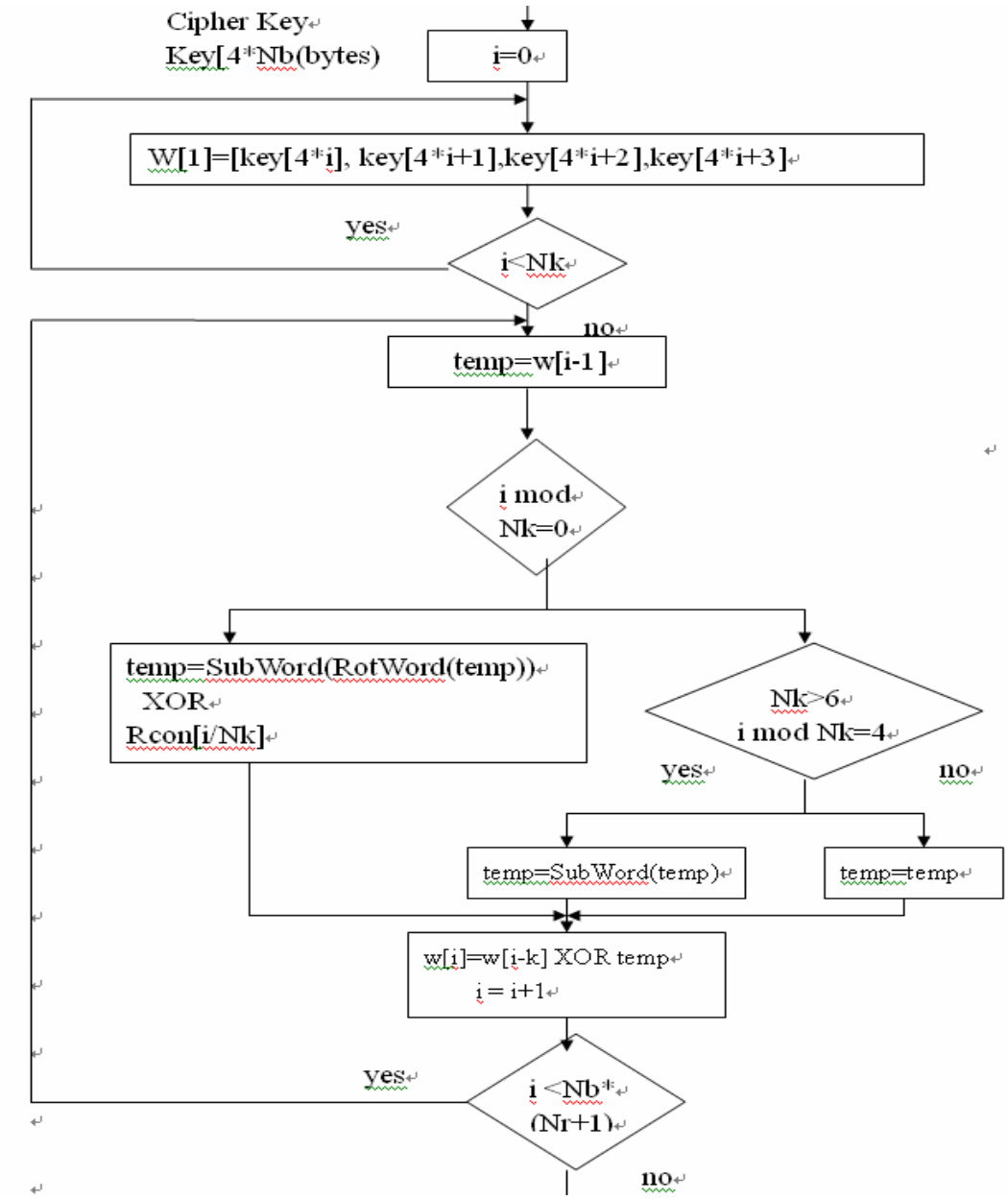


圖 2-9、金鑰擴展流程圖

(5) 解密區塊(Inverse Cipher)

Inverse Cipher 的描述如圖 2-10，只要與圖 2-10 比較就知道 Inverse Cipher 的操作相同，主要是一些參數的設定不同，以下將一些細微的差異加以說明。

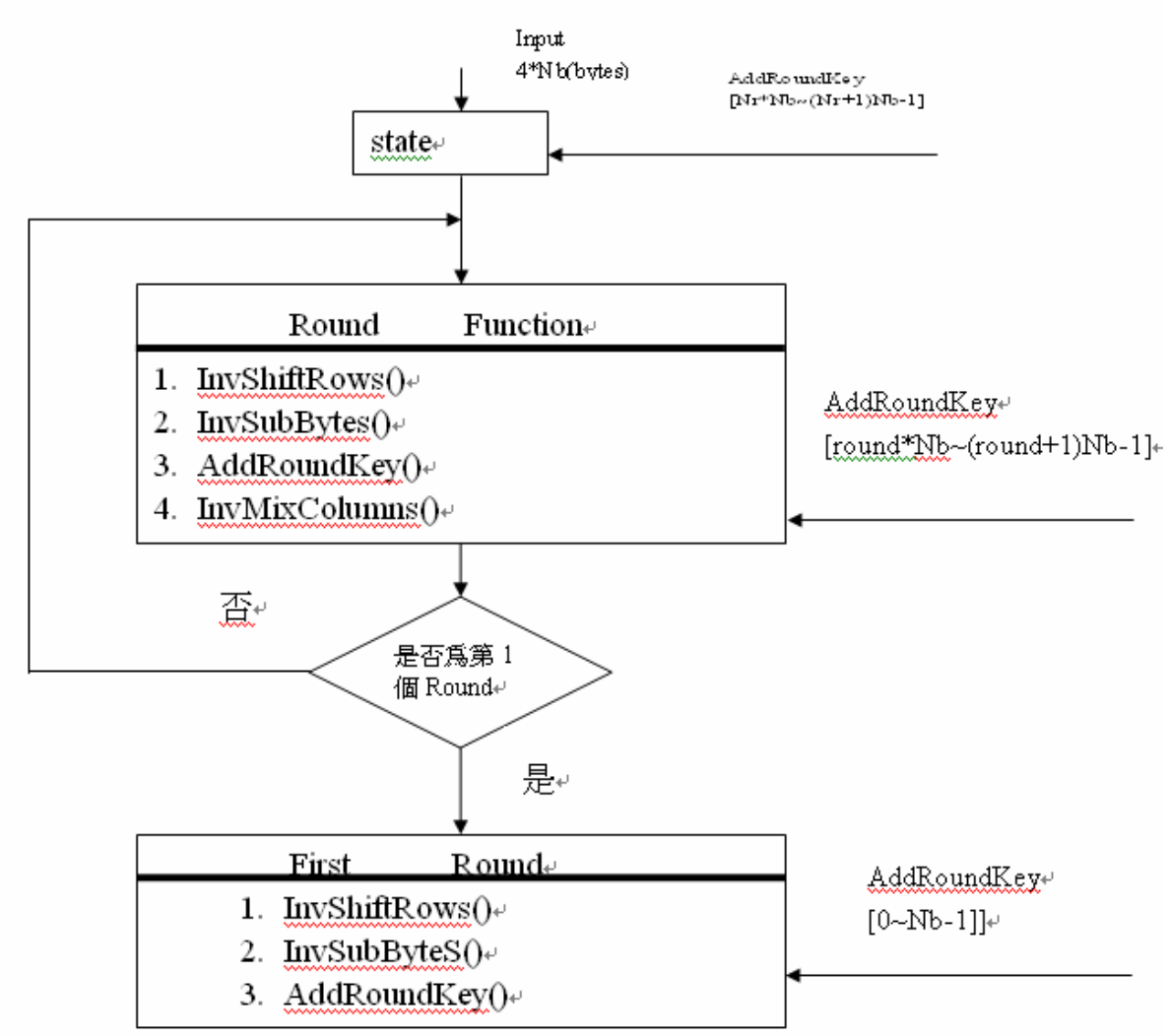


圖 2-10、Inverse Cipher 的演算流程圖

InvShiftRows 函式:

在 InvShiftRows 函式轉換中，除了第一列不進行 offset 外，針對 State 的每一列進行不同程度的位元組的循環移位，整個處理的程序如(7)式，只是其中的 shift(r,Nb)的每一列所移位的方向與 ShiftRows 相反。如下圖所示：

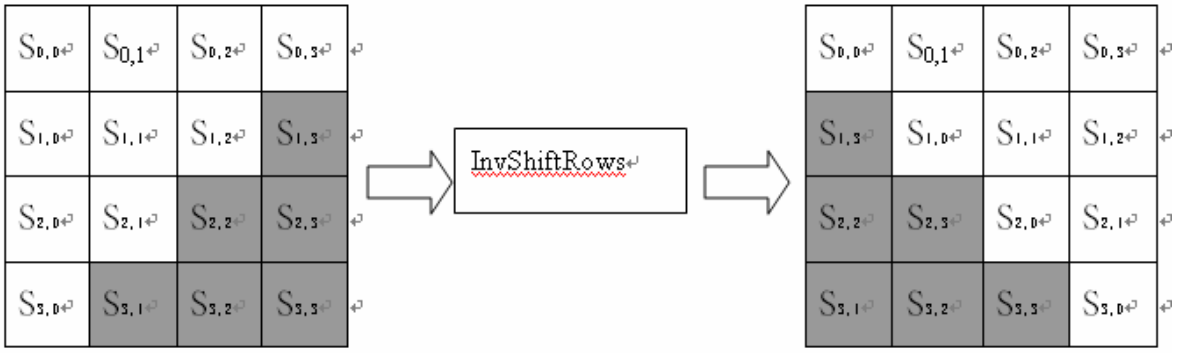


圖 2-11、 State 操作 InvSubBytes 函式圖

InvSubBytes 函式:

InvSubBytes 函式的功用主要是用來解回 SubBytes 函式所進行的非線性置換位元組。故它是利用 InvS-box 來進行轉換，而 InvS-box 的產生是由 S-box 的索引及內容相對調而來。

AddRoundKey 函式:

在圖 2-11 的流程中並沒有看到 InvAddRoundKey 函式，因為 AddRoundKey 是執行 bitwise ＸＯＲ的操作，因此只要執行兩次的 AddRoundKey 即可回復到原有的值。

InvMixColumns 函式:

在 InvMixColumns 中，我們使用了 s=a(x)⊕s(X)的字組的乘法運算且模餘 x^4+1，其中 a(X)為$\{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$，在 Inverse Cipher 中為了解回原始的值，於是 InvMixColumns 函式也同樣使用字組的乘法，只是 a(x)必須改為其字組的乘法反元素

$a^{-1}(x)$，由 $a(x)\oplus a^{-1}(x)=\{01\}_{16}$，故得到

$$a^{-1}(x)=\{0b\}x^3+\{0d\}x^2+\{09\}x+\{0e\} \qquad (10)$$

(六)Equivalent Inverse Cipher

在(六)中，Inverse Cipher 的轉換是根據 Cipher 的反向運算而得的，故除了使用了相同的金鑰外，資料的函式轉換順序與 Cipher 的架構並不一致。為了結構簡單化、資源共用與設計上的面積考量，ＡＥＳ可以根據以下的兩個特性，並藉由金鑰排程的修正，可讓 Inverse Cipher 擁有與 Cipher 一樣的架構。此種 Inverse Cipher 稱為 Equivalent Inverse Cipher。

InvSubByte 函式與 InvShihtRows 函式可互相交換置換位置，因為 InvSubByte 是進行位元組的非線性置換，並不因函式處理的時機與位置會置換不同的結果。

InvMixColumns 為線性的操作，這樣意謂 InvMixColumns(State XOR RoundKey)可以有以下的變化:

InvMixColumns(State XOR RoundKey)＝ (11)

ImvMixColumns(State) XOR InvMixColumns(RoundKey)

上式的意義在於原先 State 執行 AddRoundKey(即ＸＯＲ RoundKey)的動作，再接著執行 InvMixColumns 函式，可以轉變為先對 State 執行 InvMixColumns 函式，再執行 AddRoundkey 函式的動作，只是此處的 Roundkey 已經修正為新的 RoundKey，即為 InvMixColumns(RoundKey)。

經過這兩個性質的修正，我們可以得到 Equivalent Inverse Cipher 的流程圖，如圖 2-12，其轉換過程如同 Cipher 一樣。而修正後的金鑰陣列可由金鑰擴展處事先作補償，稱為 d W[]。

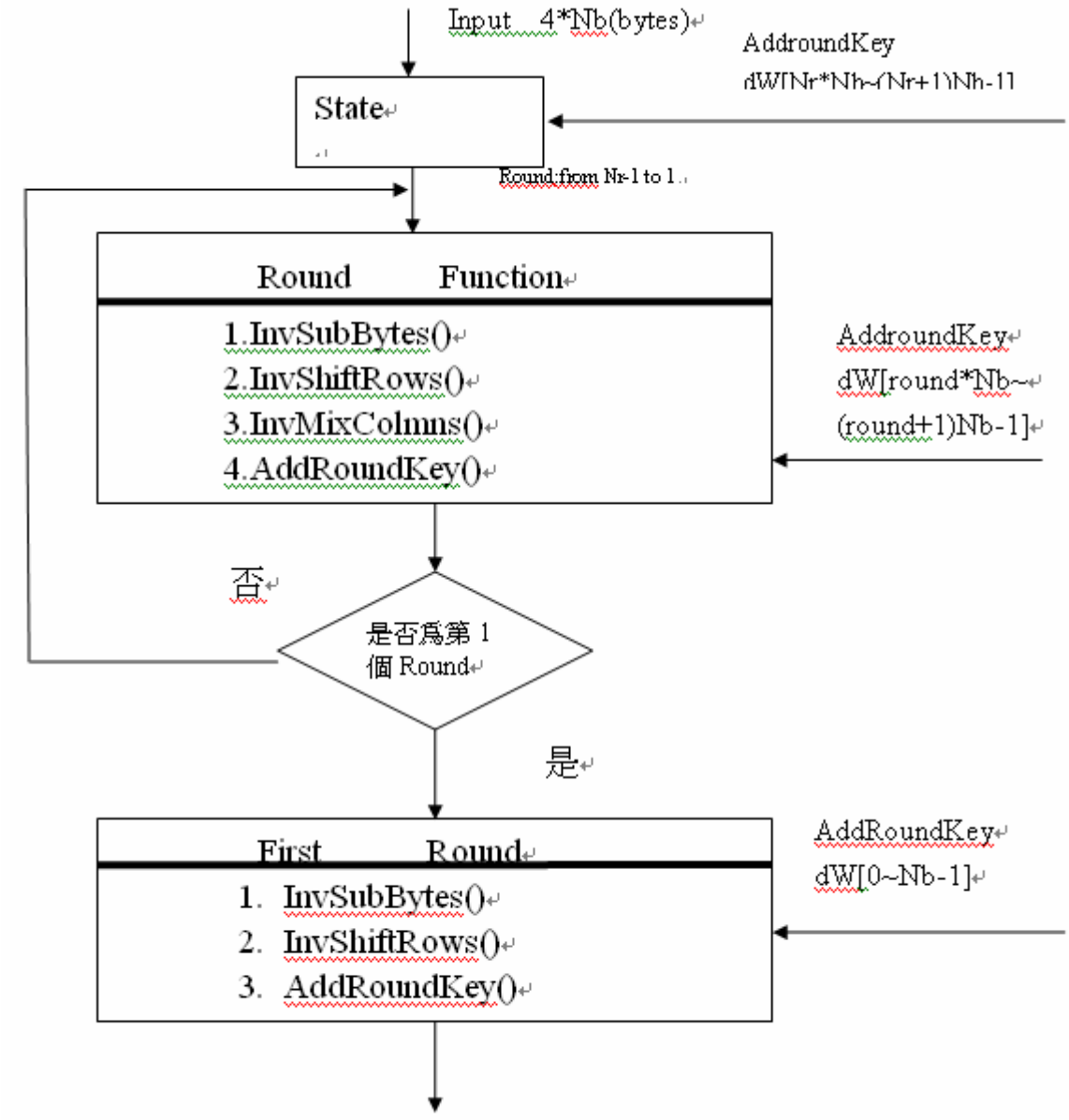


圖 2-12 、Enquivalent Inverse Cipher 的演算流程圖

第六節　資料加密標準(DES)

使用最廣的資料加密標準(DES)在【1997 年被國家標準局(National Bureau of Standard)所採用】，國家標準局現在已改名為【國家標準與技術協會(National Institute of Standards and Technology，NIST)】；而 DNS 就成為 NIST 發布的第 46 項聯邦資訊處理標準(Federal Information Processing Standard 46，FIPS PUB 46)。演算法本身則被稱資料加密法(Data Encrytion Algorithm)。DES 會使用一把 56 位元的鑰匙來對 64 位元的資料區頓進行加密。這個演算法會透過一連串的步驟將 64 位元的輸入轉換成 64 位元的輸出。而同樣的步驟與鑰匙也會被用在解密上。

在 1960 年代末期，IBM 成立一個 Horst Feistel 所主持的關於電腦密碼學的計畫。這個計畫在發展出一個名為 LUCIFER 的演算法之後，於 1971 年結束【FEIS73】。LUCIFER 被賣給勞伊船舶保險公司(Lloyd's of London)；它被用在同樣是由 IBM 發展出來的自動提款系統上。LUCIFER 是一種 Feistel 區段加密法；其輸入是 64 位元的資料區段，而使用的鑰匙長度為 128 位元。因為 LUCIFER 帶來了大好前景，所以 IBM 又著手研發具有市場性的商業加密產品，並且希望這個產品能做在單晶片裡。這件工作就由 Walter Tuchman 與 Carl Meyer 來主導，並且除了 IBM 的研究人員之外，其成員還包括 MSA 的技術指導與諮詢人員。

這個計畫的研究成果就是新版的 LUCIFER。它變得更難破解，並且為了能夠做在單晶片內，其鑰匙長度縮短為 56 位元。

國家標準局(NBS)在 1973 年公開徵求國家加密標準的提案。IBM 就將 Tuchman 與 Meyer 的計劃結果提交出去。這項結果是提交出去的演算法中最好的一個，因此在 1977 年被採用而成為資料加密標準(DES)。

DES 在被採用成為標準之前，曾遭到極為嚴苛的批評。批評者砲轟的原因有兩點。第一，IBM 原來發展的 LUCIFER 用的是 128 位元的鑰匙，但是提交的版本用的卻是 56 位元的鑰匙，鑰匙長度大幅縮減的 72 位元。反對者擔心這樣的鑰匙長度太短了，可能無法抵擋暴力攻擊法。他們關切的第二點是 DES 將內部架構(S-box)的設計策略列為機密。這樣一來，使用者無法確定 DES 是不是還有一些隱藏的弱點，讓 NSA 可以不需要鑰匙就能解開密文。後來的一些事件顯示 DES 的內部架構應該是很強固的。此外。根據參與研究的 IBM 人員所述，他們提交的版本只更動了 S-box。他們是根據 NSA 的建議來更動 S-box 的，這樣可以去除一些在評估的過程中所發現的弱點。

DES 加密程序

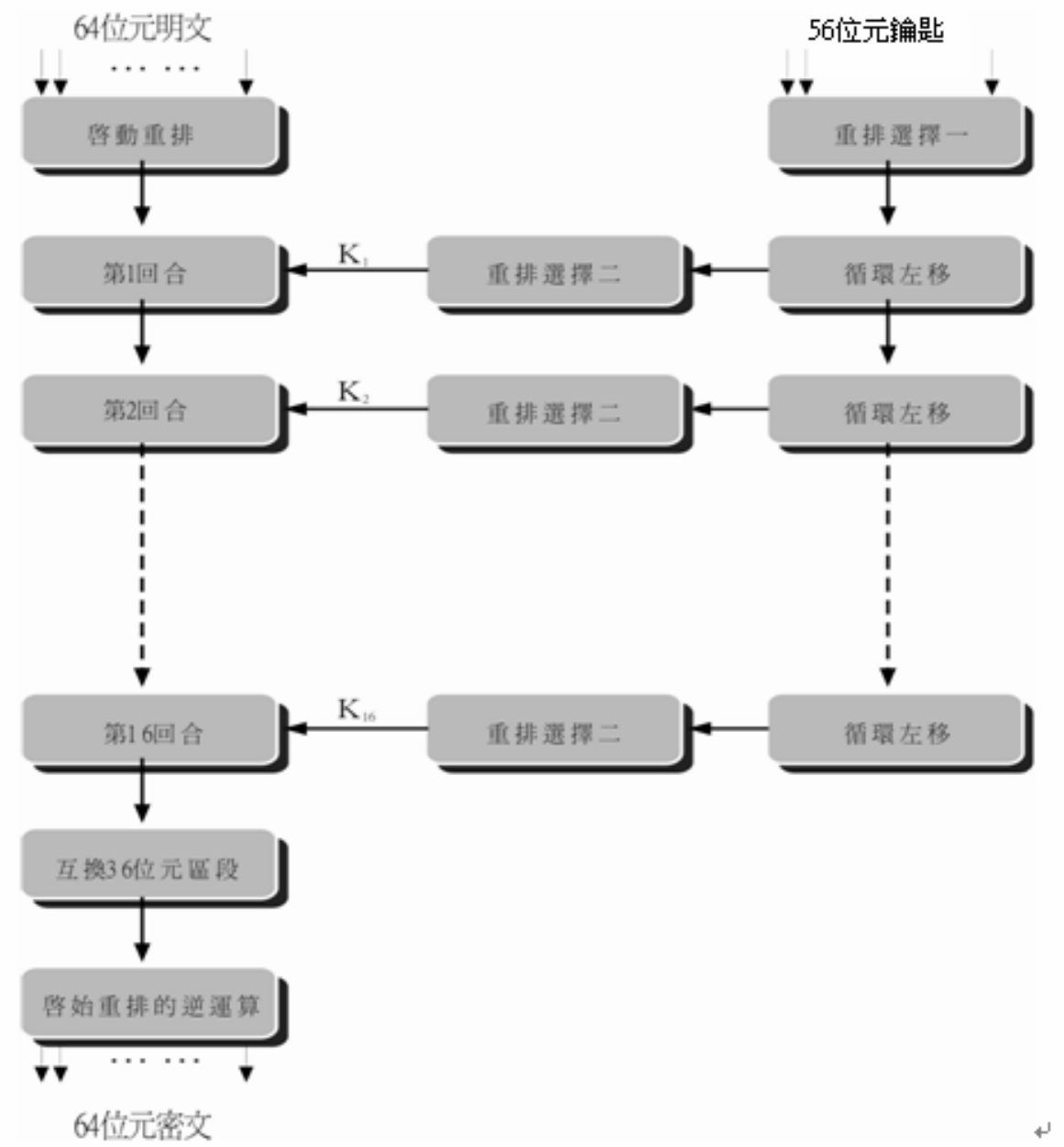


圖 2-13、DES 加密程序圖

(圖 2-13)是 DES 的整體加密架構。與所有的加密架構一樣，這個加密函數也需要兩個輸入；待加密的明文與鑰匙。在 DES 中，明文的長度是 64 位元位元，而鑰匙的長度是 56 位元。

我們可以看到密文的處理分為三個階段。首先，64 位元的明文區段會通過一個啟始重新程序(IP)，IP 會重新組合其位元，藉此產生「重新排列的輸入」。下一個階段是由功能相同的 16 個回合所組成的，其中包含了重新與取代這兩種運算。最後一回合(第 16 回合)所產生的 64 位元是根據明文與鑰匙所產生的，其輸出的左右兩邊交換過來之後就形成了”前期輸出”(preoutput)。最後，這個前期輸出會經過一個重新程序(IP^{-1})來產生 64 位元的密文。IP^{-1} 是啟始重新程序(IP)的反函數。如同除了一開始與最後的重新程序外，DES 與 Feistel 加密法具有相同架構。

圖 2-13 的右半邊顯示了 56 位元鑰匙的使用方式。一開始，這把鑰匙會通過一個重排程序，接下來，在這 16 回合中的每一個回合，都會用左旋移位以及重排的方式來產生一把子鑰匙(K_i)。每個回合所用的重排程序都是一樣的，但是因為鑰匙的位元被重複地移位，所以得到每一把子鑰匙都是不一樣的。

啟始重新排序

啟始重新排序與其逆運算式是由表格來定義的，這兩個程序分別定義在表格 8 與 9 中。表格的說明如下。

表 8、啟始重排程序(IP)表表

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

表 9、啟始重排程序逆運算(IP^{-1})表

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

表格是由編號 1 至 64 的 64 位元所組成。其中的 64 個項目代表了 1 到 64 的某種排列方式。每個項目代表某個輸入位元的輸出位置；此程序的輸出也是 64 位元。

單一回合的運作細節

顯示了單一回合的內部架構。我們將 64 位元的中間值的左右兩半視為兩個獨立的 32 位元資料，並且分別標示為 L(左)與 R(右)。

與任何標準的 Feistel 加密法一樣，每個回合中的處理程序都可以用下列的方程式來表示：

$L_i=R_{i-1}$

$R_i=L_{i-1}\oplus F(R_{i-1}，K_i)$

表 10、擴充式重排程序(E)表

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

每個回合中所使用的子鑰匙 K_i 是 48 位元。R 的大小是 32 位元。R 會先被擴充成 48 位元，擴充的方式是透過表格來定義的(表 10)。這個表格定義了一個重排運算與一個擴充運算；這個擴充運算會複製 R 中的某 16 位元。接下來，所產生的 48 位元會被拿來與 K_i 做 XOR 運算。

運算結果會傳給一個取代程序，這個取代程序產生 32 位元的輸出。最後，這 32 位元的資料會依照表格 11 所定義的方式重新排列。

表 11、重排函數(P)表

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

表 12-1、DES 的 S-BOX 定義表

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

表 12-2、DES 的 S-BOX 定義表

S5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

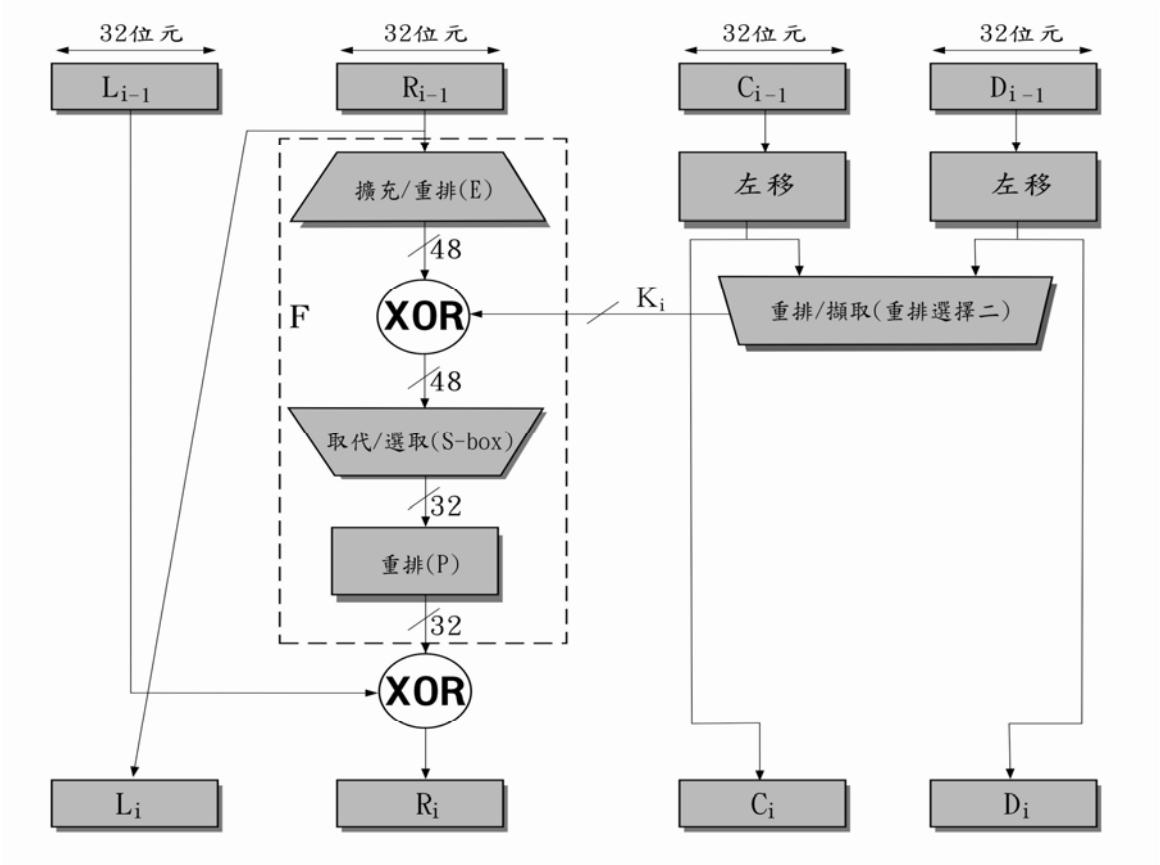


圖 2-14、DES 演算法中的單一回合

圖 2-14 所顯示的是在函數 F 中 S-box 所扮演的角色。整個取代運算是由 8 個 S-box 所組成，每個 S-box 會接受 6 位元的輸入，而產生 4 位元的輸出。表格 13 定義了這 8 個 S-box 的運算方式。它們的操作方式如下：我們將 S_i 這個盒子的輸入的第一個與最後一個位元組合成一個 2 位元的二進位數值，我們用這個數值來決定要選取 S_i 表格中的哪一列。中間的 4 位元所形成的數值就被用來選取表格中 16 行中的某一行。由此行列所選出的十進位數值就會被表示成 4 位元的二進位形式，這就是 S_i 的輸出。

例如，就 S1 而言，若輸入為 011001，則被選取的列就是 01(第 1 列)，而被選取的行就是 1100(第 12 列)。位於第一列第 12 行的數值是 9，因此 S1 的輸出就是 1001。

S-box 的每一列都定義了一個可逆的取代法則。(圖 2-14)可能有助於您瞭解這種對應關係。圖中所顯示的取代法則就是 S1 的第 0 列。

S-box 的運算值得我們進一步來討論。我們先不管鑰匙(Ki)造成的影響。您會發現原來的 32 位元輸入被分成一組一組的，每一組大小是 4 位元；然後每一組再被擴充成 6 位元，擴充的方式是重複位於兩組邊界字元。例如：某段輸入的字元是

. . .　efgh ijkl mnop　. . .

就會變成

. . .　defghi hijklm lmnopq　. . .

每一個外側的兩個位元會被用來選取四個可能的取代法則中的其中一個(就是 S-box 的其中一列)。然後，每一組中的某四個位元(就是中間那四個位元)就會被一個 4 位元的輸出所取代。

因此，這 8 個 S-box 就會產生重新排列過的 32 位元的輸出，所以這些輸出位元就會在下一個回合中儘可能地影響其他的位元。

鑰匙產生程序

讓我們回到(圖 2-12)與(圖 2-13)演算法的其中一項輸入式 64 位元的鑰匙。我們將鑰匙的位元從 1 到 64 編號；每 8 個位元跳過不管，如同表格 13 中所表示的。此鑰匙會先經過一個重排程序。這個重排的行為是由一個名為「(重新選擇→(PC1))的表格所決定的(表格 14)。接下來，產生的 56 位元鑰匙會被分成兩段 28 位元的資料，我們分別將它們標示為 C0 與 D0。在每一回合中，Ci-1 及 Di-1 都會分別被循環左移(或左旋轉)一個或兩個位元。至於到底是一洞一個或兩位元。被移動過的值就成為下一回合的輸入。這些被移動過的值同時也會是「重新選擇二(PC2)」的輸入(表格 15)。重新選擇二產生 48 位元的輸出，而這個輸出是函數 $F(R_{i-1}$，Ki)的輸入。

表 13、輸入的鑰匙表

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

表 14、重排選擇依(PC-1)表

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

表 15、左移的清單表

回合	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
旋轉回合數	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES 解密程序

與 Feistel 加密法一樣，除了子鑰匙的使用順序顛倒過來之外，DES 的加解密用的是同一個演算法。

第三章PDA 上的加密演算法

第一節 系統架構

圖 3-1、系統傳輸過程圖

(一)系統功能簡介

(1)AES 加密解密功能操作(如圖 3-2)

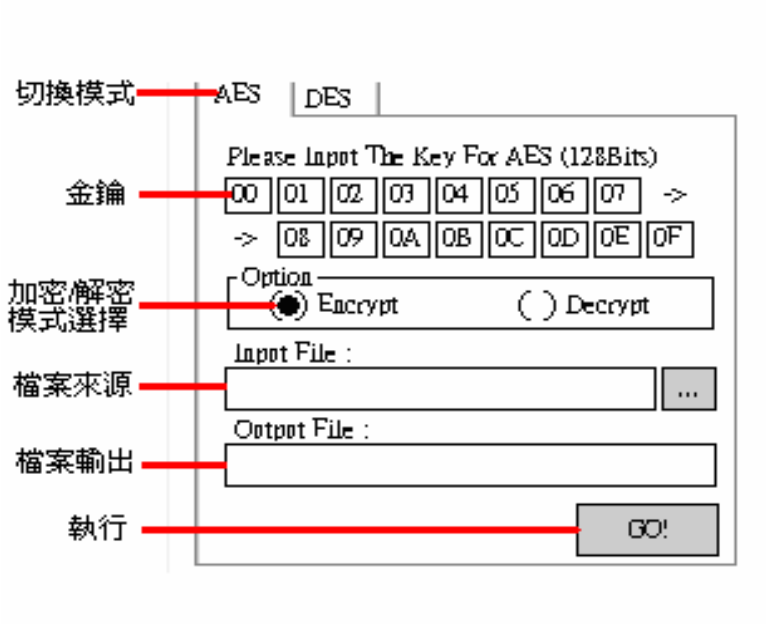


圖 3-2、按鍵說明圖

1. 切換模式:切換 AES/DES 演算法。
2. 金鑰:(範圍 00~0F)發送端可設定一組金鑰(AES 是 128 個 bit)接收端收到資料時再由發送端取得這組金鑰方可解開這組資料，發送同時若是遭人攔截，攔截者沒有金鑰也無法開啟資料。
3. 加密/解密模式選擇:發送端要對資料做加密的鍵。
4. 檔案來源:輸入要加(解)密的檔案。
5. 檔案輸出:輸出加(解)密後的檔案。
6. 執行:開始執行。

(2) DES 加密解密功能操作(如圖 3-3)

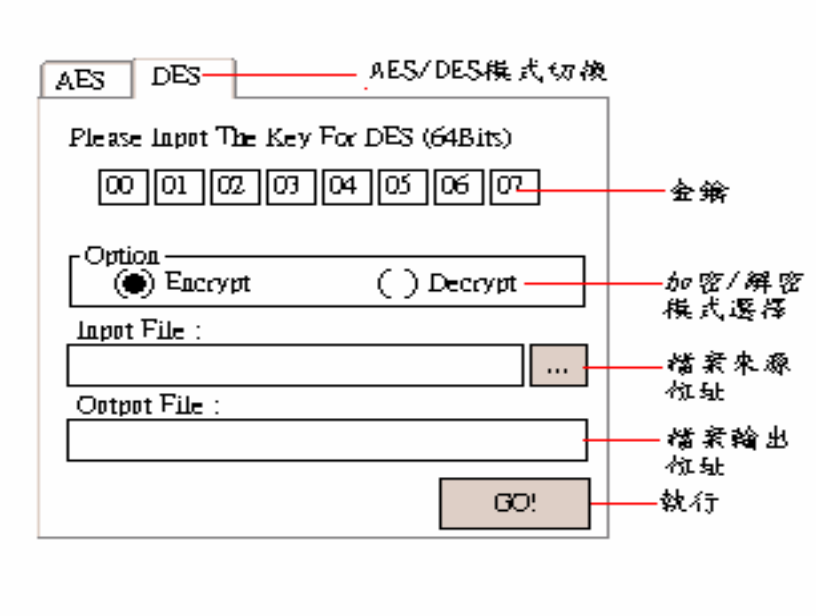


圖 3-3、按鍵說明圖

1. AES/DES 模式切換:選擇 AES 或者 DES 其一。
2. 金鑰:(範圍 00~07)與 AES 相同但是它只有 64bit。
3. 加密/解密模式選擇:選擇要加密還或者解密。
4. 檔案來源位址:輸入要加(解)密的檔案。
5. 檔案輸出位址:輸出加(解)密後的檔案。
6. 執行:開始執行加密或解密。

第二節　系統操作流程

開啟 PDA 系統畫面時，在程式集裡面開啟檔案總管，除了檔名以及金鑰要自行輸入之外，開啟文件以及變換加解密演算法只需動動筆就可以完成了。

依照順序將功能依照使用者的需求來呈現:

1. 加密(Encrypt):操作(Encrypt)功能，可以對欲加密的檔案作加密無需用手輸入。
2. 解密(Decrypt):操作(Decrypt)功能，可以對欲解密的檔案作解密一樣無需用手輸入。
3. 金鑰(Key):傳送端若是發送機密資料擔心半路遭人擷取可設定一組金鑰(Key)，當接收端收到資料時再跟發送端討取這組金鑰方可解開資料，若是遭不明人士盜取沒有金鑰還是解不開。

第三節系統特色

本專題為 PDA 上的加密演算法，主要特色在於 PDA 攜帶方便，若是突然有重要文件要發送而身旁剛好又沒有電腦時，即可以使用 PDA 來做傳送。但是又擔心資料在傳輸過程中遭人擷取，所以我們構想出若能直接在 PDA 上執行加密解密的話，應能降低在網路上傳送資料十，個人資料遭人擷取的風險。PDA 的優點是方便攜帶、能隨時紀錄訊息或查詢所需資料，且具有無窮的發展性，同時又不失功能的強大，缺點是屏幕過小，且電池續航能力有限。PDA 通常採用手寫筆作為輸入設備，而存儲卡作為外部存儲介質。在無線傳輸方面，大多數 PDA 具有紅外線傳輸和藍牙傳輸的介面，提供大眾能享有無線傳輸的便利性。許多 PDA 還能夠具備 Wi-Fi 連接以及全球衛星定位系統 GPS。

第四章系統呈現

第一節 系統效能

本系統效能是行動的安全性傳輸系統，系統中我們以 Microsoft Visual Studio 2005 開發系統。提供了 AES 跟 DES 的加密演算法。

在介面來說:使用者在開啟欄位上開啟檔案，然後選擇加密或是解密，系統就會顯示加密完成或是解密完成。

例如:今天我們加密一張圖片，只要點選 Encrypt(加密如圖 4-1)鍵入一個金 鑰執行加密動作。之後將加密過的檔案傳送給接收端，接收端收到這個資料再鍵入金鑰然後點選 Decrypt(解密)就可以取得原始的圖片。

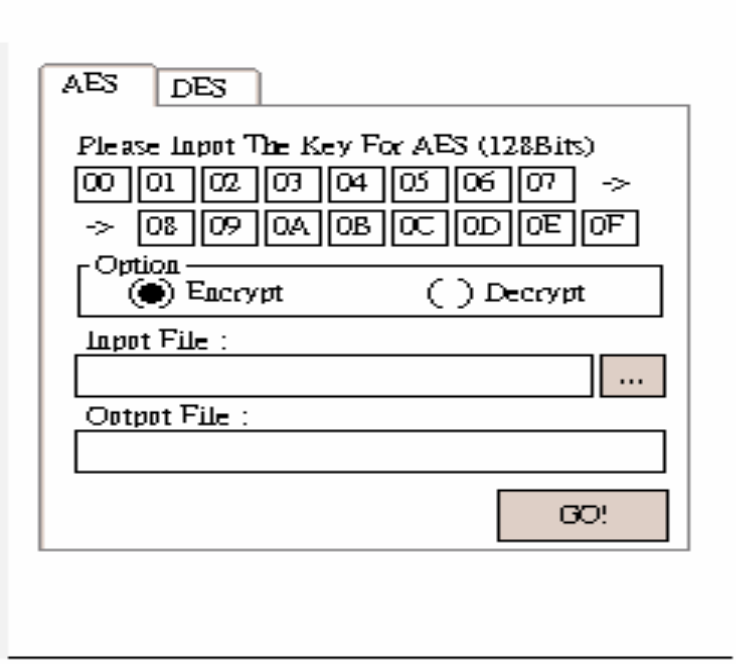


圖 4-1、介面圖

第二節 系統畫面

（一）ActiveSync4.0

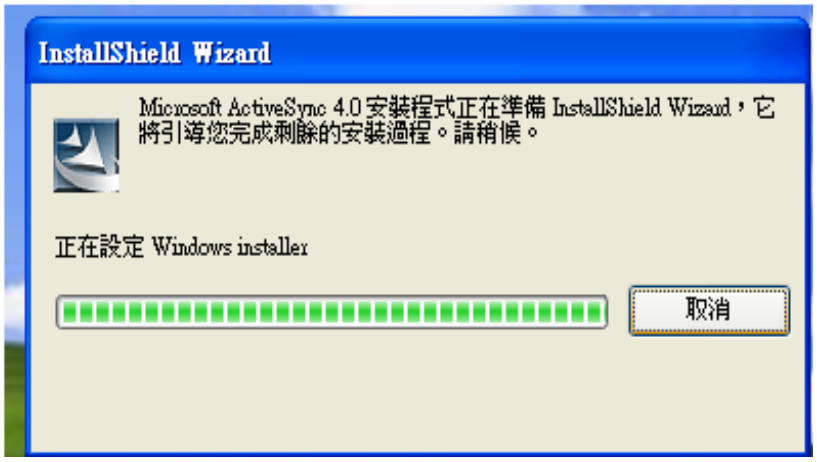


圖 4-2、安裝過程一圖

(二) 下一步

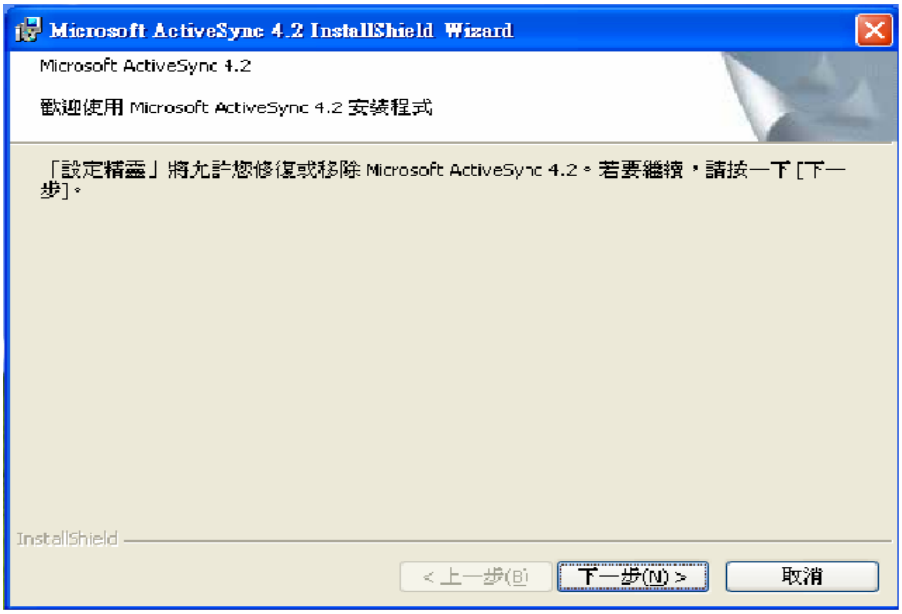


圖 4-3、安裝過程二圖

(三)ActiveSync4.2 軟體授權條約

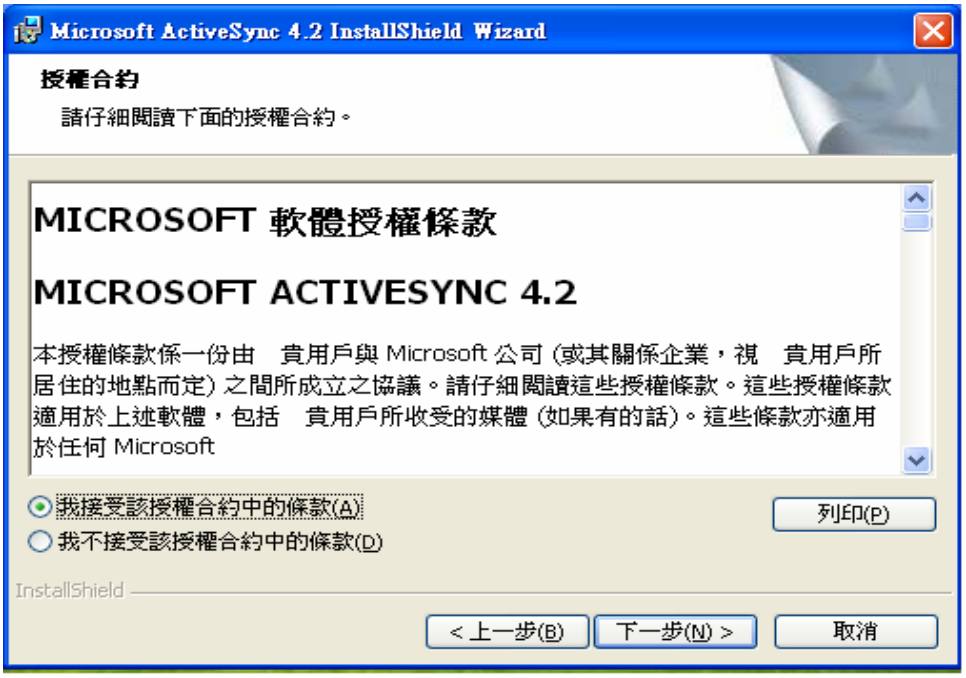


圖 4-4、安裝過程三圖

(四)可自行輸入您要使用者的名稱和組織

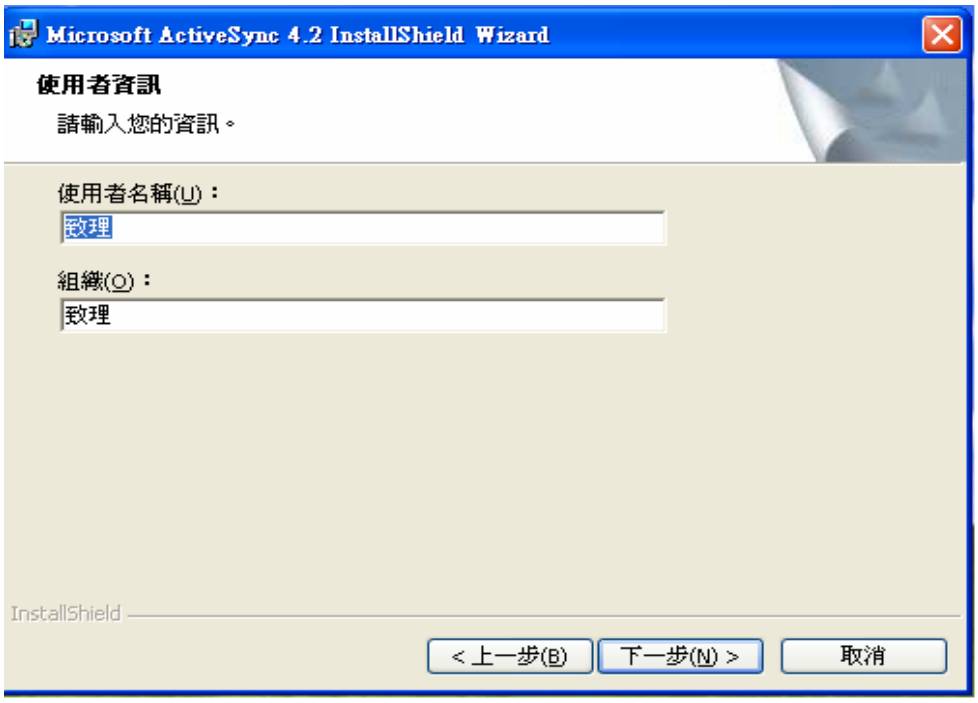


圖 4-5、安裝過程四圖

(五)選擇目的地資料夾

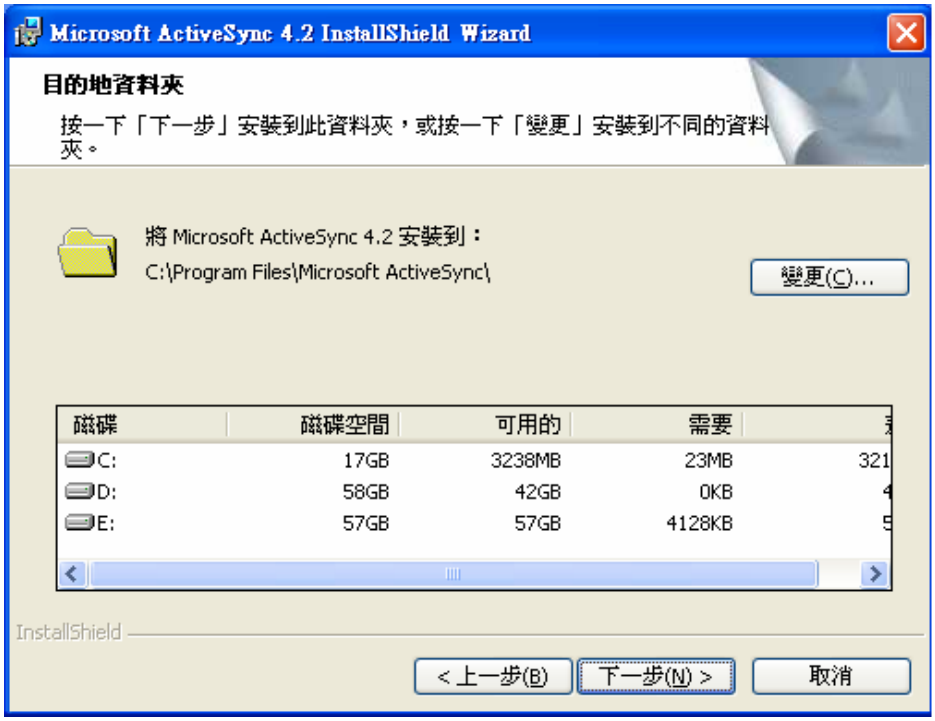


圖 4-6、安裝過程五圖

(六)ActiveSync4.2 安裝

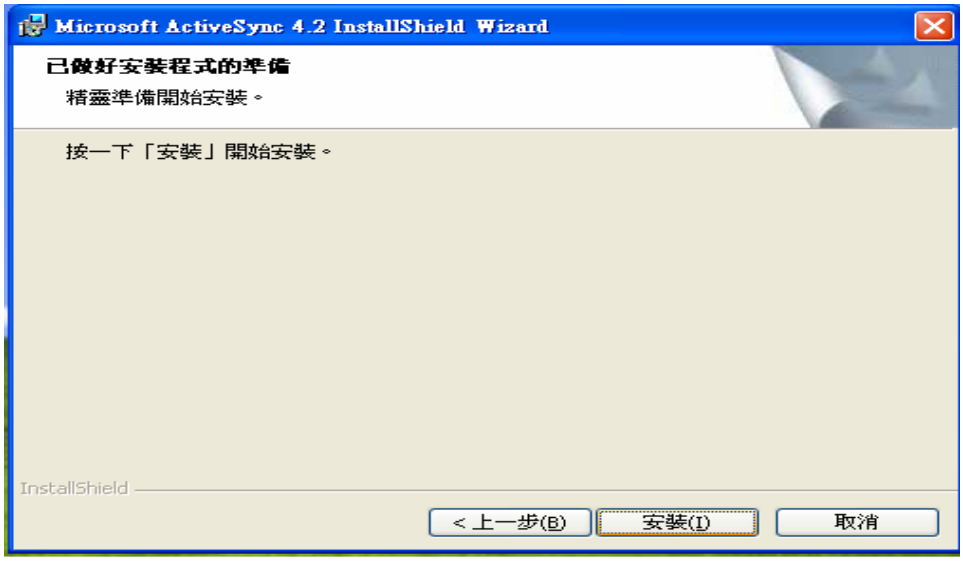


圖 4-7、安裝過程六圖

(七)安裝完成

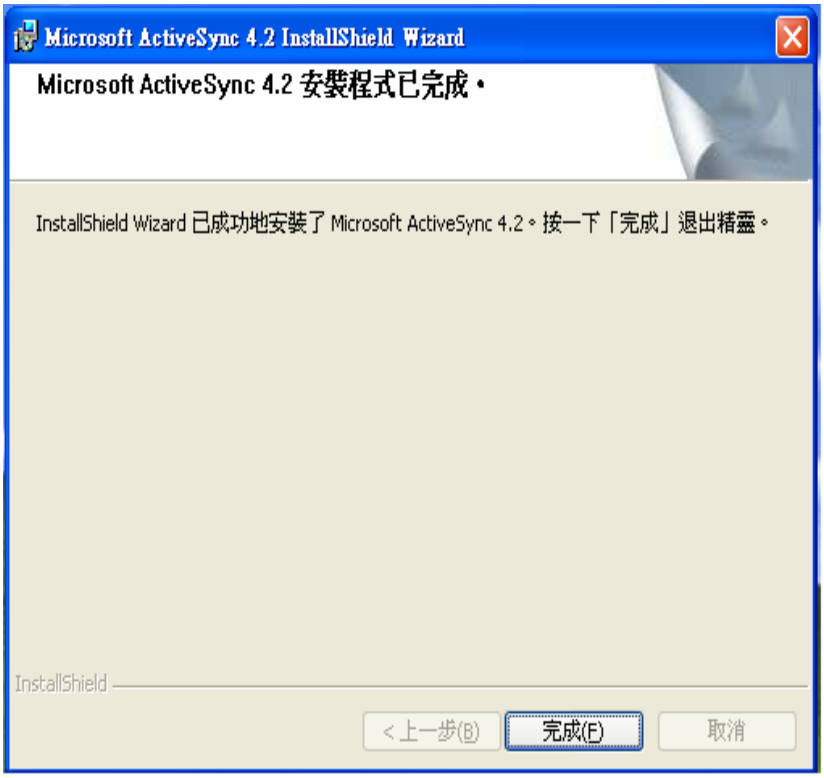


圖 4-8、開始使用 ActiveSync4.2 圖

(八)電腦開始要跟 PDA 轉換資訊

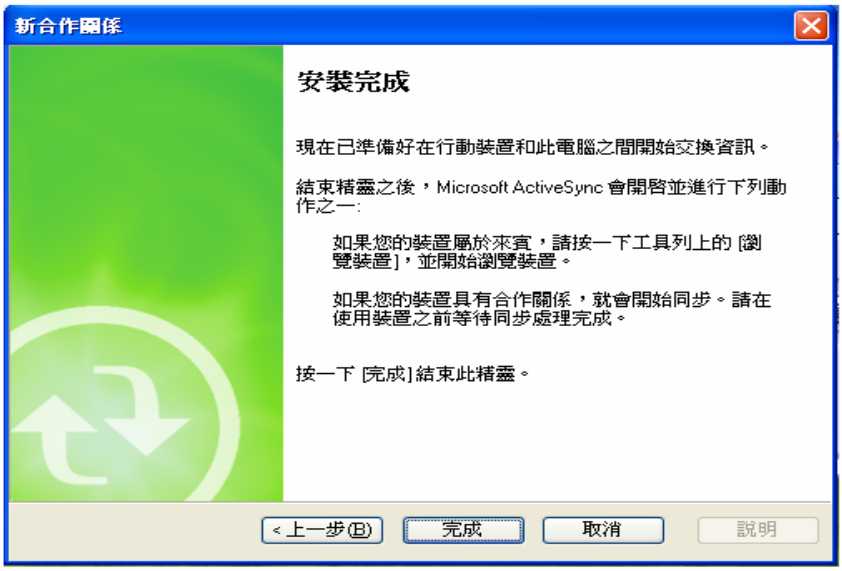


圖 4-9、使用 ActiveSync4.2 與 PDA 連接圖

(九)建立合作關係

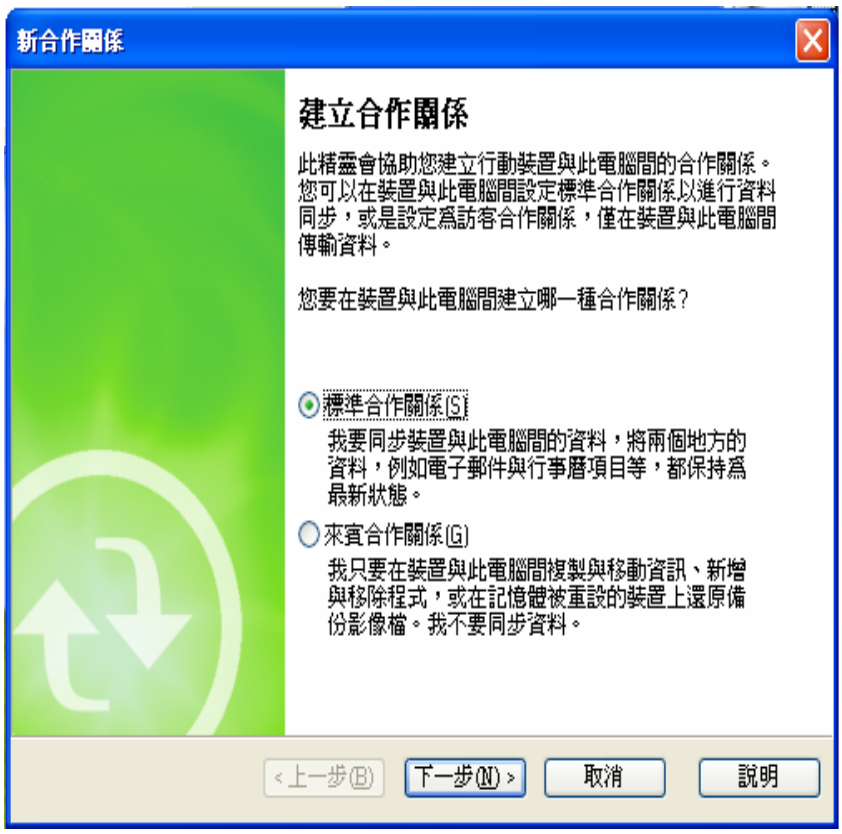


圖 4-10、選取標準合作關係圖

(十)取得連線

圖 4-11、選擇要連線的裝置圖

(十一)選取連線裝置

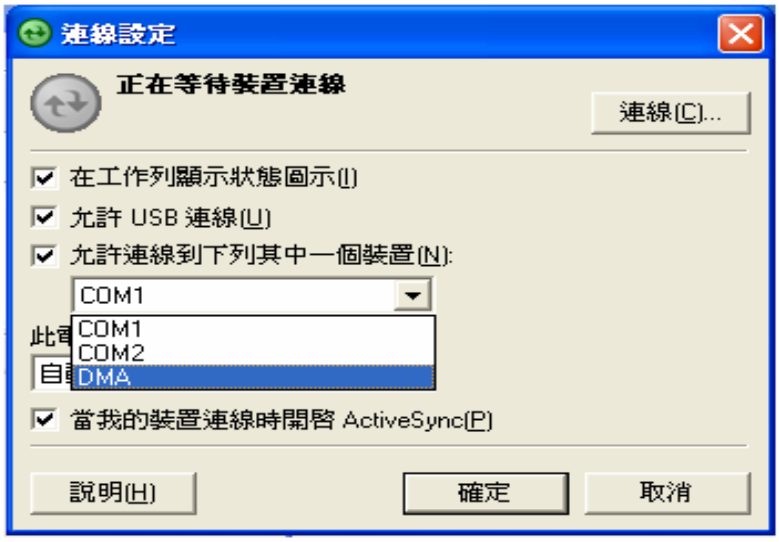


圖 4-12、選取 DMA 才能跟 PDA 做連接圖

(十二)開始搜尋

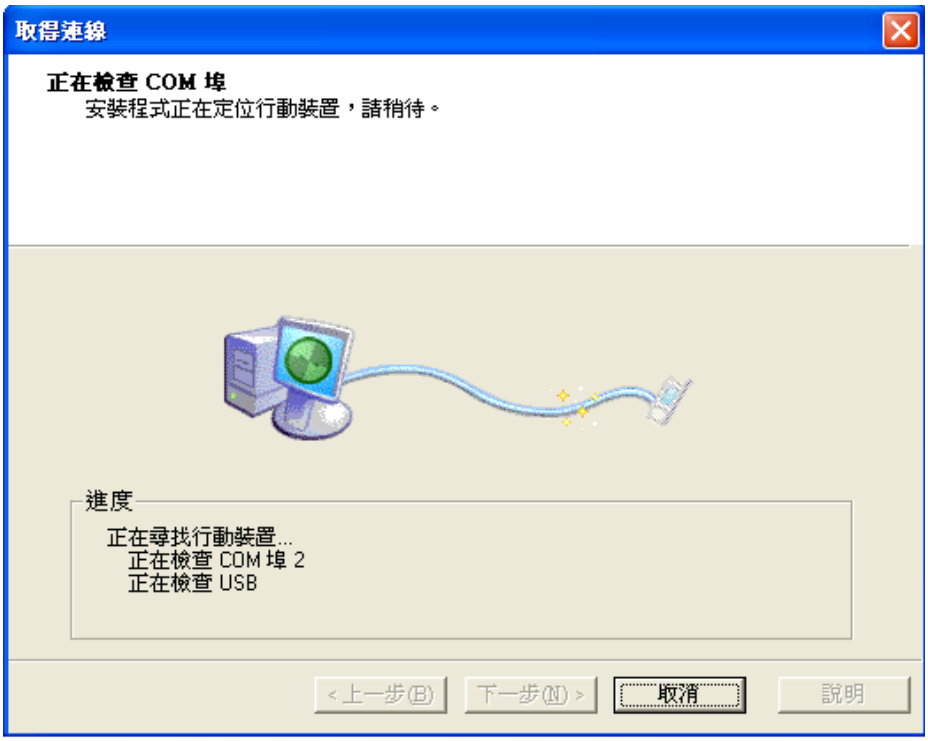


圖 4-13、搜尋裝置圖

(十三)開啟 PDA 模擬器

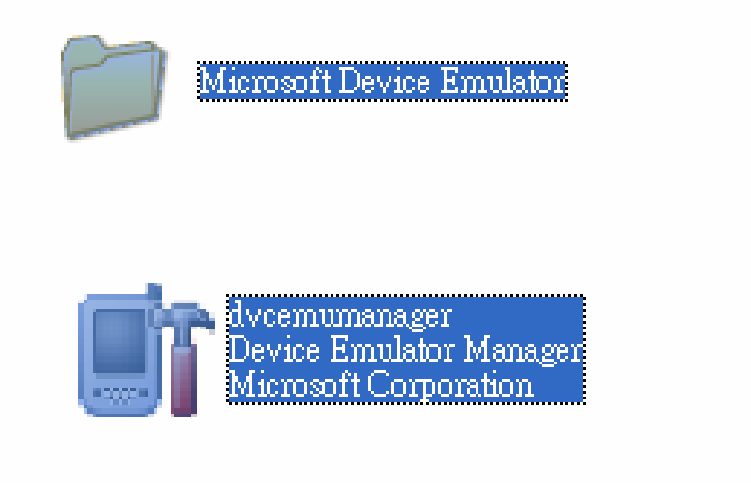


圖 4-14、在 C 槽裡的 C:\Program Files 裡圖

(十四)PDA 開啟

圖 4-15、PDA 的畫面圖

(十五)與 PDA 同步連接底座與 ActiveSync4.2 同步

圖 4-16、PDA 同步連接底座圖

(十六)當 PDA 與 ActiveSync4.2 同步後

圖 4-17、紅圈處的箭頭就會變成雙向的圖

(十七)選取 ActiveSync4.2 的流覽裝置

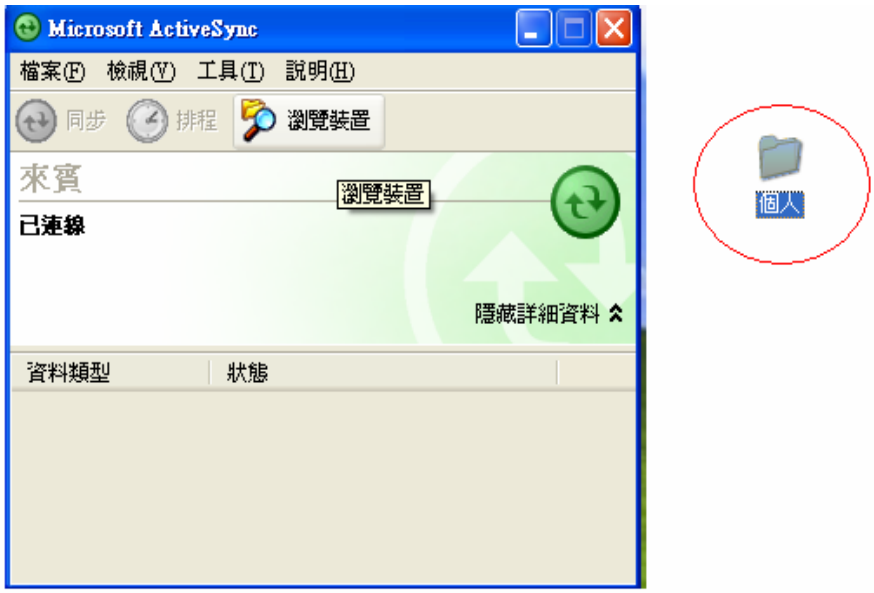


圖 4-18、然後選取裡面一個「個人」資料夾圖

(十八)匯入檔案如圖

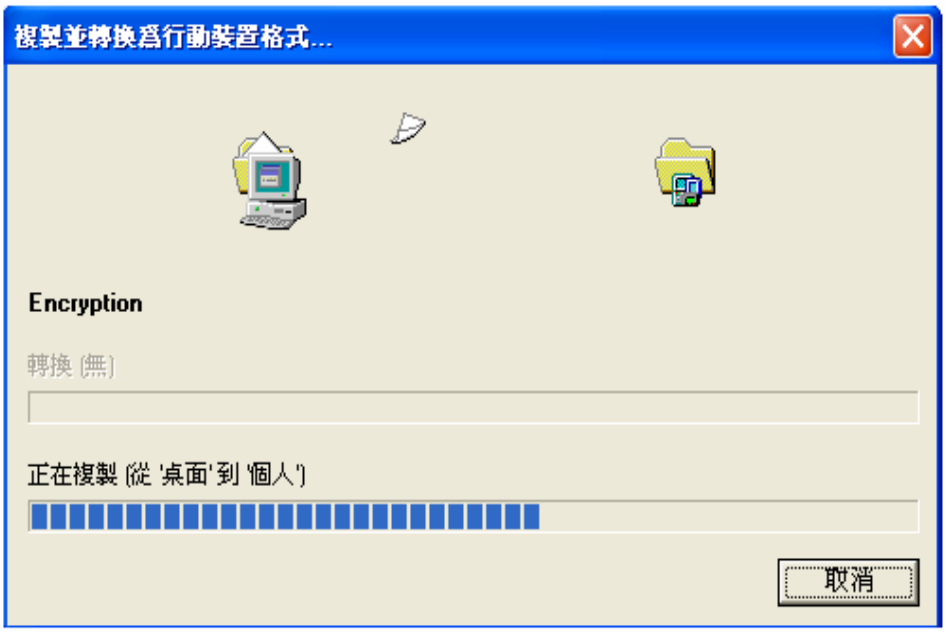


圖 4-19、匯入我們的專題 Eneryption 圖

(十九)點選檔案總管

圖 4-20、開啟反藍處圖

(二十)選 My Documents

圖 4-21、點選反白處圖

(二十一)再選取“個人”的資料夾

圖 4-22、反白處圖

(二十二)接著匯入檔案到個人如圖

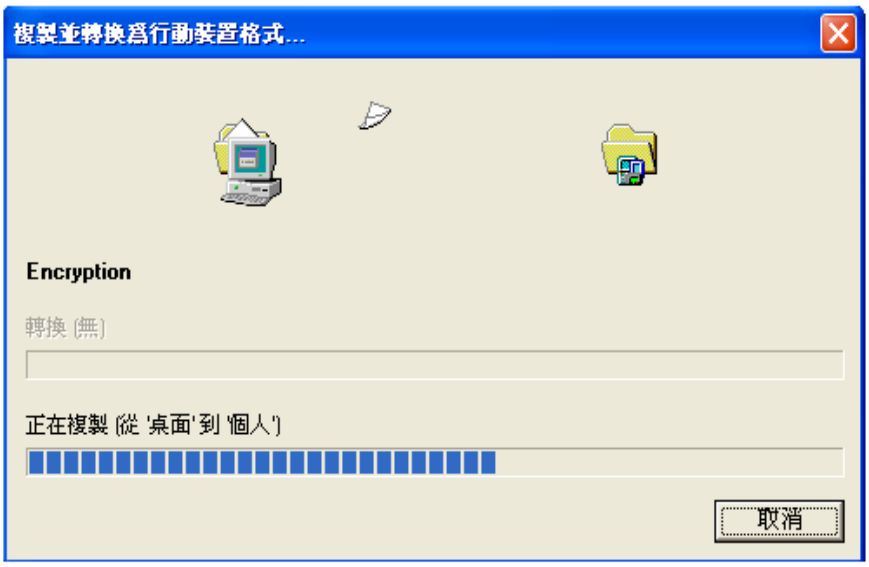


圖 4-23、匯入主程式圖

(二十三)再選取我們的檔案

圖、4-24 在個人的資料夾裡面開啟圖

(二十四)開始對密文加密

圖 4-25、用 PDA 的 WORD 做加密圖

(二十五)進行加密

圖 4-26、密文加密中圖

(二十六)加密過後

圖 4-27、密文加密後變亂碼圖

(二十八)進行解密

圖 4-28、密文進行解密圖

(二十九)解密成功

圖 4-29、出現此圖代表解密成功圖

(三十)解密後的結果

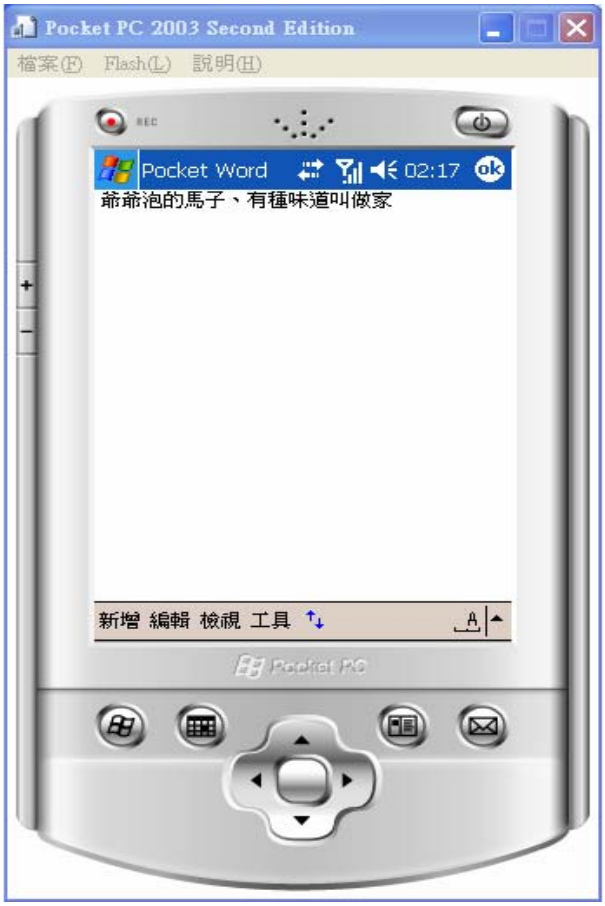


圖 4-30、解密後的文件與原始文件一樣圖

結論

在資料安全性越來越重視的今天，安全與可靠的加解密標準已成為保護資料不可獲缺的工具，新一代的對稱式加解密演算法 AES 因此產生。我們於研究 AES、DES 的相關文獻，已成功可以將程式執行於 PDA 上透過加解密功能可以將不想讓一般大眾看到的文件加以鎖起來。

(一)、使用了 AES 與 DES 這兩種不同的加解密演算法能確實的提供某種程度的安全傳輸環境。

(二)、確實的提供了檔案適當的保護。

(三)、雖無法做到不被第三方任意截取檔案卻可以使其無法輕易得到明文。

參考文獻

[1] AES Questions and Answers
http://www.nist.gov/public_affairs/releases/

[2] http://asia.playstation.com/tch_hk/index.php?q=psp/hardware

[3] http://www.hakuyu.com.tw/nds/spec.htm

[4] 張宇超，2006，Visual C# 2005 資訊安全程式設計，文魁出版。

[5] Stallings 著，巫坤品、王青青 譯，2006，密碼學與網路安全原理與實務，第三版，碁峯出版。

[6] 維基百科 http://zh.wikipedia.org/wiki/Wiki

[7] 蘇持平、吳誠文，2004，密碼處理器之設計與測試，國立清華大學電機工程學系博士論文。

[8] 洪嘉隆、吳誠文，2004，使用即時金鑰產生器之 AES 密碼晶片設計，國立清華大學電機工程學系碩士論文

[9] 王仁傑，2003， XL2 演算法之實作，國立中山大學資訊工程學系碩士論文。

[10] 楊中皇，2002 年 6 月，密碼學演算法於 IC 卡上的具體實現，資訊安全通訊，第八卷第 3 期，頁 8~17。

[11] 楊中皇，2005 年 1 月，橢圓曲線密碼系統軟體實現技術之探討，資訊安全通訊，第十一卷第一期，頁 15~25。