逢 甲 大 學

資訊工程學系專題報告

視窗介面實作密碼學

學　　生：呂宜峻（四甲）
　　邱劭華（四甲）
　　蘇淑瑛（四甲）

指導教授：李維斌

中華民國九十二年十二月

目錄

圖表目錄

第一章　導論

1.1　研究動機

1.2　研究目標

1.3　研究目的

1.4　工作進度甘特圖

1.5　工作分配表

1.1 研究動機

在資訊發達的今天，電腦似乎已經成為許多人生活的一部份，無論是工作上或是私人方面，常常必須把重要的資料存放在電腦中，或者是透過網路傳送給另一方。此時存在著一個讓人頭痛的問題，那就是如何防止別人偷窺對自己極重要的檔案資料呢？

網路駭客大量掘起，相信窺視一份沒作保護的檔案是相當容易的，而其帶來的損失更是不可限的，小至個人隱私權受損，大至可能危及國家的存亡，因此資料安全一直以來都非常受矚目，如何保護機密檔案，這是一門非常值得讓人深入探討的學問。對於我們而言，這個專題只是入門，未來這條路仍是非常長遠寬闊的。

1.2 研究目標

1. 藉由一個視窗的介面(如應用軟体 Winzip 或是檔案總管般的視窗介面)，來對檔案進行加密、解密的動作，並且在此介面中可以看出那些檔有加過密或是只是一般的檔案，甚至能夠隱藏使用者存放加密檔案的資料夾。
2. 在 windows 介面上，滑鼠右鍵按下時蹦現出選單能執行使用的功能(如應用軟体 Winzip，在滑鼠右鍵按下時可以對檔案進行壓縮或是解壓縮的動作。)，新增出一個可以對檔案進行加密動作的功能選項。
3. 利用加密之 API 達成加解密的執行。
4. 提供自解檔的功能讓使用者即使在沒有安裝我們的軟體時也能夠享有加解密的服務。
5. 更進一步希望能將加密過的檔案進行偽裝的動作，如加密過的檔案看起來就像是一般的 word 檔案一樣，而執行後開啟的也就如同一般的 word 文件一般。

1.3 研究目的

研究這個專題的最終目地在於提供使用者一個能對重要檔案做保護的方法，讓使用者的隱私有所保障，以避免重要或私密的檔案被窺視後所帶來不可預知的損失。

1.4 工作進度甘特圖

學期	92學年度下學期						93學年度下學期					
月份	2	3	4	5	6	7	8	9	10	11	12	1
尋找老師												
收集資料												
確立目標												
系統分析												
分配工作												
系統實作												
系統整合												
系統測試修改												
撰寫報告												
報告整合校對												

圖1.1 工作進度甘特圖

1.5 工作分配表

姓名 / 工作	呂宜峻	邱劭華	蘇淑瑛
資料收集	yes	yes	yes
視窗程式設計	yes		
Dialog介面設計		yes	
加解密function	yes		yes
滑鼠右鍵蹦跳選單			yes
自解檔		yes	
報告編寫	yes	yes	yes
系統整合	yes	yes	yes

表1.1 工作分配表

第二章　開發背景技術

2.1 什麼是登錄檔

當 Windows 在啟動和運作的過程中，就須要有儲存軟、硬體、系統和應用程式等相關資訊的地方，如使用何種的顯示卡、音效卡、畫面的字型及顏色、視窗的大小、位置等等。而在存放這些資訊的地方，我們就稱之為「登錄檔」。

登錄檔的資料都是分散儲存在 Windows 目錄下的兩個檔案中，分別是 SYSTEM.DAT 以及 USER.DAT，它記錄了種種的系統設定資訊，來提供給 Windows 系統和相關的應用程式，而系統和應用程式若修改了相關的設定，也會同時修改登錄檔內的資料，以能隨時確保登錄檔的正確性，這對系統管理員來說是個非常重要的資訊。

一般來說，除非必要需求，否則盡可能不要自行對登錄檔作變更，因為如果登錄檔作了錯誤的變更，可能會導致系統出現嚴重的問題。因此若登錄檔必需變更時，最好事先作好備份，以預防萬一。

「登錄檔」在 Windows 第一次安裝時就會建立，而當電腦每次重新開機時，系統會偵側電腦週邊，並且和登錄檔中的資料做對照，看看是否有新的硬體加入或是有硬體被變更過。若是有的話，則系統將會自動或提醒使用者改已手動的方式來安裝驅動程式，並更新登錄檔內容。

我們可利用 Windows 提供給我們的程式正常地查看或修改登錄檔的內容，這個程式就在 Windows 目錄下的 REGEDIT.EXE ，只要你在這程式上的圖示上快按滑鼠兩下或在執行的欄位上直接輸入也可以來啟動登錄編輯器。下圖所示在執行欄位直接輸入的例子：

圖 2.1 啟動登錄編輯器

2.2 登錄檔的資料型態結構

登錄檔是一種階層式、樹狀結構化的資料庫。登錄值主要儲存在兩個檔中。正在使用的檔案是以目前的系統組態資料為主，系統會將這些有關電腦目前的使用設定資料記錄於 System.dat 裡;至於所有使用者的設定資料則通通存在 User.dat 裡。

啟動 Windows 內的登錄編輯器(REGEDIT.EXE)後，會開啟一個與檔案總管很類似的介面視窗，視窗中左邊的窗格是登錄檔的樹狀結構，是用來顯示機碼及子機碼的，且有五個以"HKEY_"為開頭的機碼，如下列所示：

1. HKEY_CLASSES_ROOT ： 這個機碼主要包含 OLE（Object Linking and Embedding）、所有檔案的類型、檔案關聯、圖示、副檔名、跟檔案對應配置的資訊。對應資料主要是在選擇一種特定檔案格式時，Windows 可從對應的應用程式來執行或列印該檔，也就是說那個檔案要使用那個程式開啟會被記錄在這個機碼裡。
   ＊ OLE 目地在使各個軟體間的各種 Object，包括聲音、影像、文字等等互相連結。例如在小畫家的圖可拿到 PhotoImpact 去修改。Linking 跟 Embedding 不同在於 Linking 是動態連結，改變新物件的內容也會影響到原來的，Embedding 不會。
2. HKEY_CURRENT_USER ： 這個機碼包含有關系統與程式的使用者特定設定值。其機碼是使用者登錄時，由該使用者在 HKEY _USERS 裡的登錄值資訊所建立的。因此這裡是有關該使用者喜好使用的環境設定值。舉例來說，今天某人將他登入 Windows 時所播放的音樂設定為 Faye.wav，則在他登錄時，這個設定就會列在這個鍵值裡，當然裡面還包括了顏色配置、標題、桌面配置等等。
3. HKEY_LOCAL_MACHINE ： 這個機碼裡詳細的列載了有關電腦、驅動程式及其他系統設定，也就是對機器描述資訊，主要是有關於已安裝的硬體種類、連接埠的對應情形、軟體組態設定及其他資訊。這些資料主要是針對機器而不是使用者，但通常若要破解軟體，可從這裡下手。
4. HKEY_USER ： 這個機碼包含著該台工作站所有使用者的資料。每個使用者的資料都存在這裡。此外還有一個一般使用者設定。這個設定主要是給以新使用者身份登錄工作站時用的參考樣版。而此資料主要是程式、事件規畫、桌面還境等等的設定值。
5. HKEY_CURRENT_CONFIG ： 這個機碼記錄著該電腦實際連接的硬體設定。主要用在當電腦有多重的組態時可使用。而這裡的資料是複製位於 HKEY_LOCAL_MACHINE 裡的組態資訊。

圖 2.3 左邊是 Registry 的樹狀結構，稱作機碼、子機碼(Key、SubKey)， 右邊顯示機碼下記錄的名稱(Value Name)以及資料(Value Data)。名稱前頭會有個圖示表示資料的型態(Data Type)。資料型態主要有三種

1. 字串(String)：最常見的一種。(ex. Windows xp)
2. 二進位值(Binary)：以 16 進位表示。(ex. 7F 31)
3. 雙字元值(DWORD)： 32 位元值, 以 8 進位表示.。(ex. 0x00003613)

以下圖示登錄編輯程式的畫面：

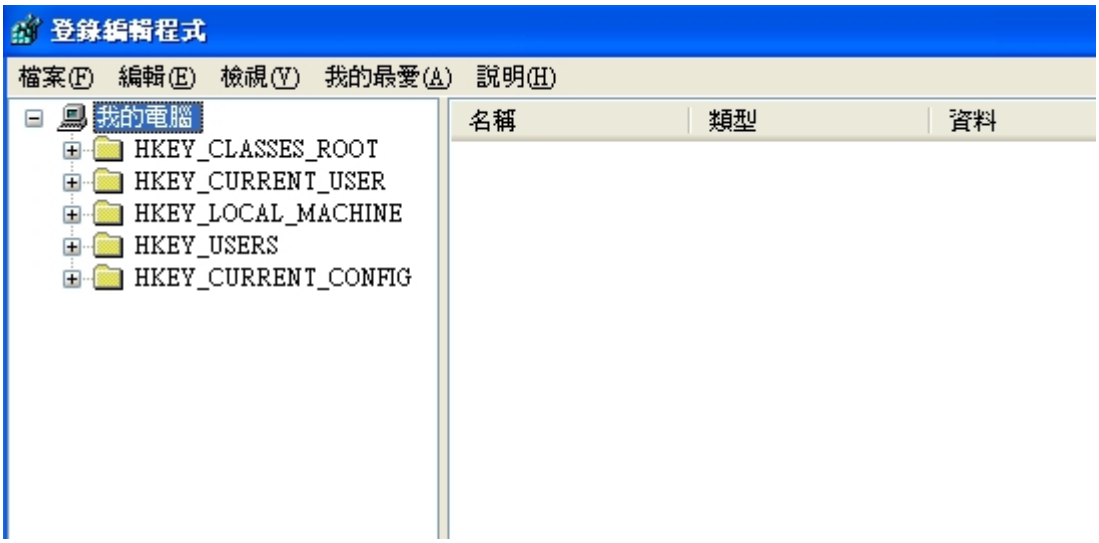


圖 2.2 登錄編輯程式畫面

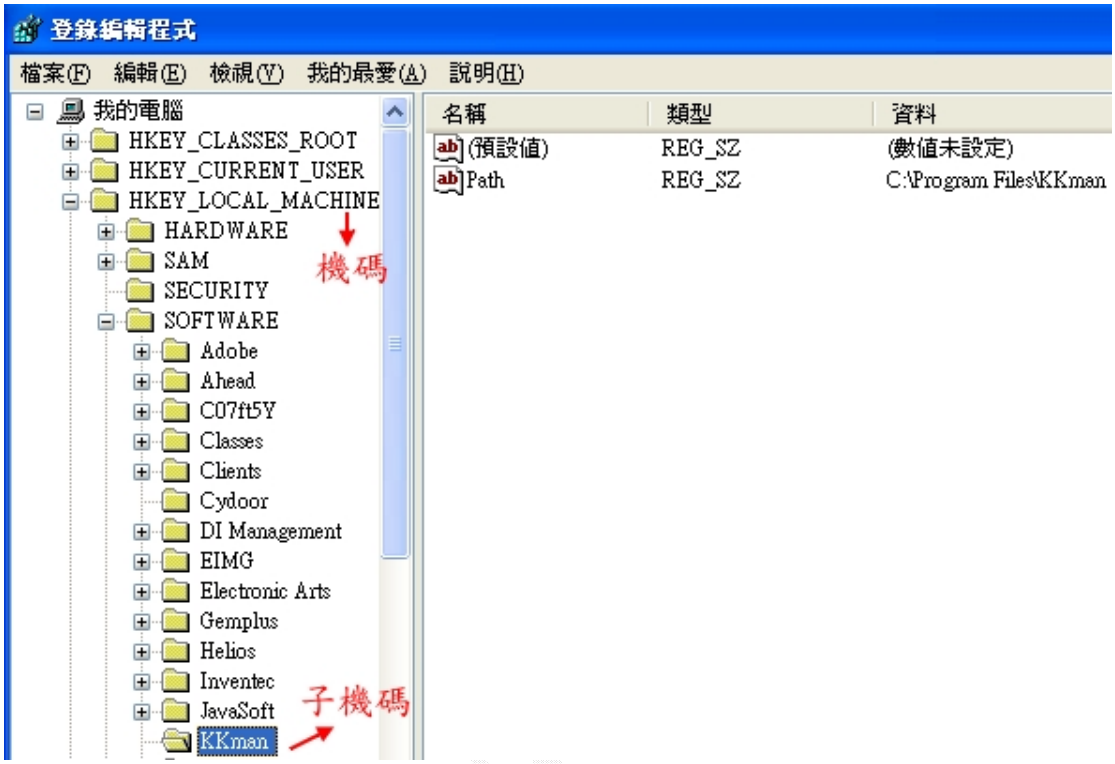

登錄編輯程式
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 說明(H)
我的電腦
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HARDWARE
SAM
SECURITY
SOFTWARE
機碼
Adobe
Ahead
C07ft5Y
Classes
Clients
Cydoor
DI Management
EIMG
Electronic Arts
Gemplus
Helios
Inventec
JavaSoft
KKman
子機碼
名稱 類型 資料
(預設值) REG_SZ (數值未設定)
Path REG_SZ C:\Program Files\KKman

圖 2．3 機碼、子機碼圖示

2.3 密碼學簡介

回顧過去，據說最先有意識使用一些技術來加密信息的可能是公元六年前的古希臘人。他們使用的是一根叫 scytale 的棍子。送信人先繞棍子卷一張纸條，然後把要寫的信息打在上面，接着打開纸送给收信人。如果不知道棍子的寬度（這裡作為鑰匙）是不可能解密裡頭的內容的。

然而在資訊爆炸的今日，隨著網路與科技的不斷演進，不論是身處於何處的你我，都可能面臨資訊安全的衝擊。隨時隨地有駭客或不法之徒在覬覦妳我甚至公司企業所擁有的機密文件。應此如何對公司內部機密或個人電腦中重要的檔案採取適當的防護措施，便是一門極大的學問。

以下將簡介兩大類型的加密演算法，一種是傳統加密（ 對稱式加密演算法 ），另外一種是公開鑰匙加密（ 非對稱式加密演算法 ），對稱加密演算法加解密時使用的是同一把鑰匙，它主要的問題，是如何在網路上安全地被交付這一把加解密的鑰匙？因為這一把鑰匙於網路上傳送交付，也可能被攔截、竄改。而非對稱式加密演算法加解密時使用的是不同的鑰匙，適合於網路上安全地交付加密用的公開金鑰，再以私有的鑰匙去解密，但它的運作方法比對稱式加密法複雜許多，所以在選擇演算法時需多加考量。

2.4—1 傳統加密－古典技術

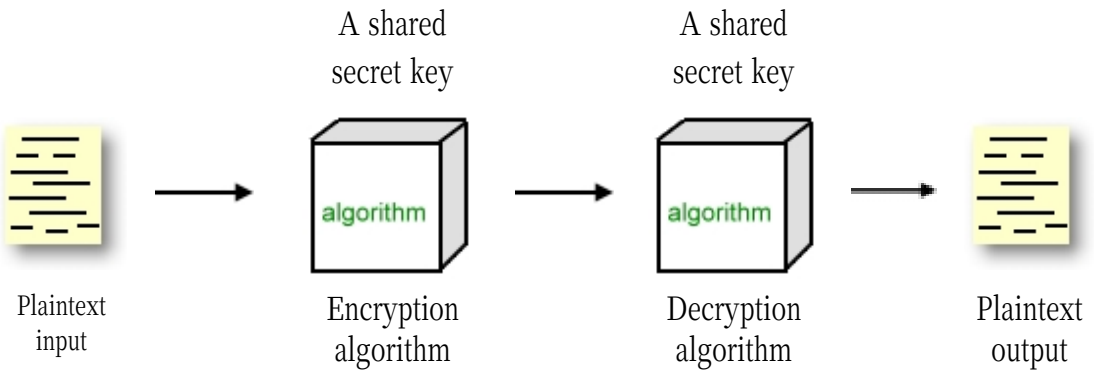


圖 2.4　傳統加密的簡化模型

上圖表示的是傳統加密的程序，原本的資料(即「明文」plaintext)經過加密演算法這個機器運作以後，被轉換成毫無意義

的亂碼(即「密文」cipher- text)。加密的程序是由一個演算法和一把鑰匙所組成的，這把鑰匙是由資料傳送者和接收者所共有，而所謂的鑰匙，其內容是一個和明文無關的數值。這個演算法會根據當時所使用的鑰匙而產生不同的密文輸出。

當密文傳送出去後,收到密文者可使用加密時使用的鑰匙還原成原來的明文。傳統加密法的安全性取決於鑰匙而不是演算法，但前提是其加密演算法要夠強讓人無法解出明文，換句話說，即使讓人知道加密的演算法也無法得知資料內容。

由此可知傳統加密方法被廣泛使用的原因,由於不需要保護演算法，所以在許多方面都可被低成本低應用。

已加密訊息的破解方式

破解方式	破解者擁有的資訊
只知道密文	●加密演算法 ●已經還原的密文
已知明文	●加密演算法 ●已經還原的密文 ●一個或多個透過鑰匙產生的明文－密文組
自選明文	●加密演算法 ●已經還原的密文 ●破解者自選的明文和透過鑰匙所產生的相對密文
自選密文	●加密演算法 ●已經還原的密文 ●破解自選的可能密文與透過鑰匙所解開的相對明文
自選文字	●加密演算法 ●已經還原的密文 ●破解者自選的明文和透過鑰匙所產生的相對密文 ●破解者自選的可能密文與透過鑰匙所解開的相對明文

表 2.1　已加密訊息的破解方式

資料隱藏

資料隱藏是將重要的文件隱藏於一般的文件中讓非法攔截者即使攔取到資料時也把它當成只是一般文件，而去解讀文件的機率也會比較低；這就像大自然生物的本能一樣，它們懂得在危險環境中偽裝自己成為周遭物體的一部份，而這似乎也是原本技術所欠缺保護自我的本能。

一般來說，資料偽裝可以將機密文件偽裝成一般的文字檔案，或者是偽裝成任一種數位煤體的形式，如視訊、影像、聲音等等。在這之間尚存在很多問題，像是資料偽裝後是否會遺失資料或者資料會受損，這些將是重要問題之一。

使用資料隱藏最大的好處是如果使用者有損失的話，一定是有人發現他們正在交換機密，對加密系統而言，訊息本身就很重要、必須加以保密，並且可能被用來辨識傳送者或接收者的身份。

Caesar Cipher 加密法

Caesar 加密法即是固定將每一個字母，位移三個順位替換 如
明文：I am a good student
密文：L DP D JRRG VWXGHQW

如果我們知道密文是由 Caesar 加密法所產生的。那麼我們即可利用暴力法來破解，原因是因為(一)我們已知加解密的演算法(二)我們只需嘗試 25 種鑰匙(三)我們已知明文所使用的語言，並且這語言非常容易辨識。

Monoalphabetic 加密法

這是一個簡單的加密方法，這種加密方法主要是將明文每次位移幾個字母，而成為密文，其位移的字母數要保持私密，解密時再將明文還原回來，但其不甚安全，因常可由出現頻率欄預測其位移的字母數。

Playfair 加密法

playfair 是最有名的多字元加密技巧，這種方法是將雙字元的明文視為單一元素，再將其轉成雙字元的密文。

2．4－2 傳統加密－現代技術

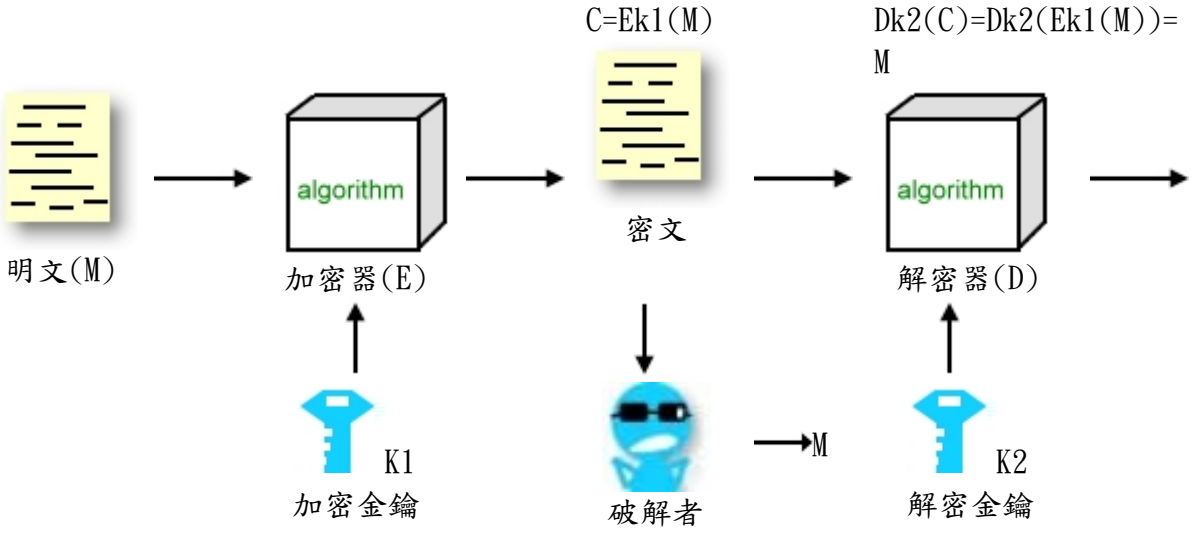


圖 2．5　DES 演算法簡化流程圖

DES

DES 為美國國家標準局 (NBS, NIST 前身)於 1976 年公佈的加密標準，其鑰匙長 64 位元，但因其中每個位元組皆取 1 位元作為同位(parity) 核對，故有效鍵長 56 位元。

DES 是使用 56 位元長度金鑰之區塊加密器 (Block Cipher) ，輸入一固定區塊 64 位元資料，加密後亦輸出 64 位元的密文。 DES 總共經過 16 個回合的處理，每一回合都必須輸入一把不同的子金鑰 (Sub-key)，這些子金鑰是由一把 56 位元長度的母金鑰所衍生出來的。

DES 系統的基本原理是利用多重加密的觀念並利用 Confusion（混淆）與 Diffusion（散佈）等方式，將明文轉換成其他格式，並散佈明文的每一個小部分擴散到密文的各部分以達到加密效果，讓密文無法利用統計方式或其他數學分析技巧將加密後的「密文(Ciphertext)」還原成原來的明文。不過，至今仍有許多人擔心 DES 的鑰匙長度太短，無法抵擋暴力攻擊法，使得其安全性受到質疑 ，此外 DES 的內部架構(S-box)的設計也被列為機密，如此讓使用者相當擔心是否 DES 內部隱藏了某些弱點，讓 NSA 甚至不需要鑰匙就能解開密文，但經過近年來的破解研究顯示 DES 內部似乎是很強固的。

DES 演算法距今也已經二十多年的歷史了。 計算機的速度已經進步得驚人，因此這項標準早就有更換的必要。

NIST 於 1997 年 4 月對外宣布了下一代區塊加密器(Advanced Encryption Standard, AES)徵選的的訊息。

2000 年 10 月 2 日正式宣佈由比利時二位密碼專家 Joan Daemen 及 Vincent Rijmen 二位博士所設計 Rijndael（發音為 Rain Doll）獲選為 AES 之演算法。

IDEA

IDEA 是由瑞士蘇黎世 ETH 在 1991 公佈的加密標準，其鑰匙為 128 位元，基本運算區塊也是 64 位元輸入、64 位元輸出。其 6 4 位元的區塊被切成 4 個均等的區塊，經過 8 次連續的與鑰匙區塊 XOR、相加、相乘、並相交換，獲得最後的加密區塊。IDEA 是近年來用於取代 DES 的傳統加密演算法之一，主要考量在於密碼學上的強度和實作上的便利性。

Triple-DES

Triple-DES 是整合三個 DES 的基本區塊，進一步增進加密的深度。若針對由 Tuchman 所提供的 DES 替代方案，即使用兩把鑰匙的 Triple-DES 來說是相當受歡迎的，這個機制依照加密－解密－加密(EDE)程序來運作，但有許多研究人員認為兩把鑰匙不太可靠，儘管許多破解方法不切實際，但三把鑰匙或許會是更好的替代方案。

Blowfish

Blowfish 是由 Bruce Schneier 所發展出來的加密法。

其特性如下：

(1) 快速的： 在 32 位元的微處理器中，Blowfish 加密每 Byte 的資料只需要 18 個時序，比 DES 還快。

(2) 緊密的： Blowfish 能在小於 5K 的記憶體中執行。

(3) 簡單的： Blowfish 的簡單架構使得它可以很簡單的被實作。

(4) 可變的： Key 的長度是可變的，最長可達 448 位元，這允許我們在執行速度與安全性上作取捨。

Blowfish 並不適用於記憶體受限的應用領域，其與 DES 不同的地方在於 Blowfish 的 S-Box 是與鑰匙有關的，我們不斷引用 Blowfish 本身來產生子鑰匙與 S-Box，如此可以完全的弄亂位元間的關係讓破解變的非常的困難。此外 Blowfish 還有一個特性：每一個回合中的運算都會使用在左右兩段資料上，如此會大幅提昇密碼學上的強度。對於暴力攻擊法來說只要慎選鑰匙的長度 Blowfish 是很安全的，因為其鑰匙長度最長到 448 個位元。

AES

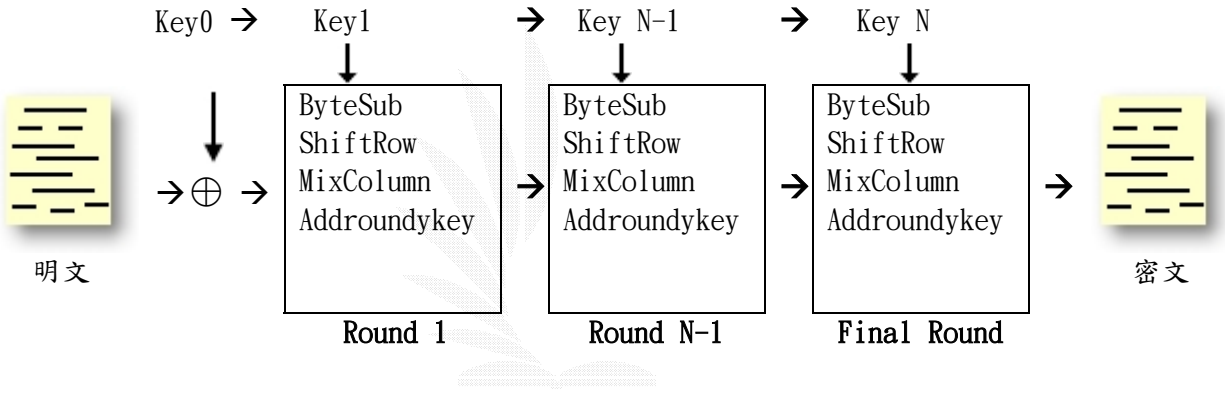


圖 2．6　AES 演算法簡化流程圖

AES(Advanced Encryption Standard)由比利時二位密碼專家 Joan Daemen 及 Vincent Rijmen 二位博士所設計 Rijndael（發音為 Rain Doll)。Rijndael 是一個反覆運算的加密演算法，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，在進行加解密時區塊最小限制為 128 位元而秘密金鑰之長度可以為 128、192、或 256 位元。

就安全性而言比 DES 演算法高出許多，如果一秒可以破解 DES，則仍需要花費 1490000 億年才可破解 AES。

AES 未來可以用來保護美國政府的一些敏感(unclassified)資訊，也可以應用在電子商務交易，自動櫃員機(ATM machine)，無線通訊，虛擬私人網路(Virtual Private Network)等領域上。

關於 AES 的攻擊法，Square attacks 是目前針對 Rijndael 最強大的密碼分析法。這是一種利用密碼器的位元組導向結構，採用與 Rijndal 相似的回合結構的選定明文攻擊法

(chosen-plaintext)。此外，尚有另外一些方法，但是都沒有辦法可以破解原本的 Rijndael,，所以可以說，現在還沒有比窮舉法更快的方法來破解 Rijndael。但由於所花費的時間不切實際，所以在找出攻擊的捷徑之前，我們可以說 Rijndael 是安全的。

AES 和 DES 演算法的比較

	DES	AES
密碼系統	對稱式密碼系統	對稱式密碼系統
年代	1973	2001
發明人	IBM 及美國國安局(NSA)	比利時科學家
加密金鑰長度	56 bits	128,192 及 256 bits
明文區塊大小	64 bits	128 bits
S-Box 設計原理	至今尚未公開	採用抽象代數構
安全度	目前已逐漸不足	高

表 2.2 AES 和 DES 的比較

2.5 公開鑰匙加密

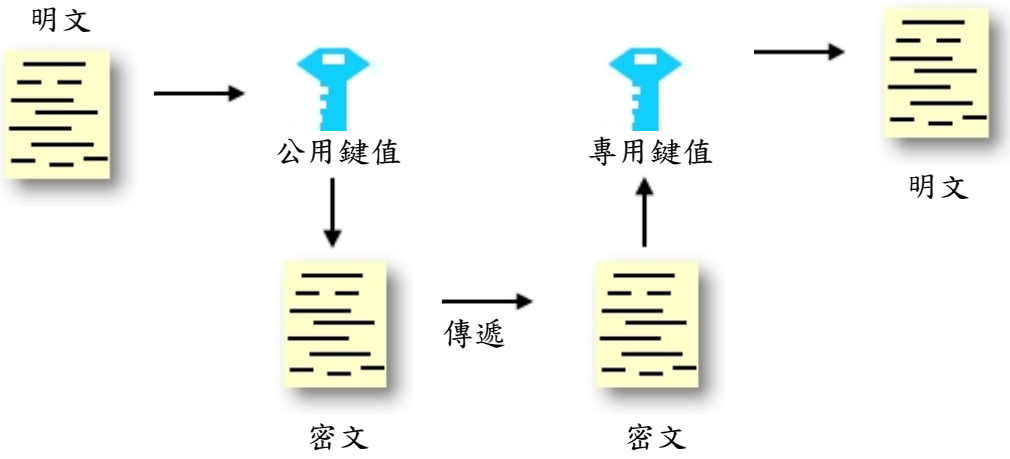


圖 2.7　公用鍵值密碼系統的基本類型

公用鍵值密碼需兩個相關的鍵值，自己保留一個專用鍵值，而另一個可自由分給他人，是可公開的公用鍵值，在傳遞文件時，只要要求對方以公用鍵值加密文件，如此只有自己可以專用鍵值解密文件。此外，公用鍵值有效地減少單一鍵值密碼系統所需的安全頻道。

RSA

RSA 它的鑰匙長度不定，普通是 512 位元 ，其基本運算區塊長度是可變的，但明碼區塊須小於鑰匙長度，暗碼區塊則等於鑰匙長度，由於 RSA 速率較一般的秘密鑰匙演算法(如 DES 、 IDEA)還慢，故較常用於加密較短的資料，或對秘 密鑰匙加密保護、再以秘密鑰匙加密訊息，或用於數位簽名。 關於 RSA 的安全性有三種可能的攻擊法：

(1) 暴力法：即嘗試所有可能的私密鑰匙。
(2) 數學攻擊法：有數種方式效果都等於分解兩質數的乘積。
(3) 計時攻擊法：取決於解密演算法的時間。

RSA 對暴力攻擊法的防衛跟一般的密碼系統一樣，即給予鑰匙一個足夠大的空間，如此能增加破解的難度。

Diffie - Hellman

Diffie -Hellman 這個演算法目的是讓兩個使用者可以安全的交換一把鑰匙，以供後續的加解密，有許多商用軟體都使用這種鑰匙交換技術。而 Diffie - Hellman 的破解關鍵在於計算出一個由很大質數產生的有限場(Finite Field)中的離散對數(Discrete Logarithms)的困難度來決定，無論是 RSA 或是 Diffie-hellman 加密系統，其保密性均是取決於經由公用鍵值推算出的私人鍵值的困難度。就單純編碼加密角度來說，兩者之間的差異並不大，但就目前電腦的處理速度來看，若是鍵值長度漸漸變大時，
Diffie-hellman 演算法的效能上會比 RSA 來的比較有優勢。

RSA 和 Diffie-Hellman 演算法的比較

	RSA	Diffie-Hellman
密碼系統	非對稱式密碼系統	
年代	1978	1976
發明人	Ron Rivest、Adi Shamir、Len Adleman	Diffie、Hellman
破解關鍵	須因數分解一個很大的數	由一很大質數產生的有限場之離散對數的困難度來決定
保密性	取決於經由加密的文字即公用鍵值推算出的私人鍵值的困難度	
效能優勢	就目前電腦的處理速度來看，若鍵值長度漸漸變大時，Diffie- hellman 演算法的效能上會比 RSA 來的比較有優勢.	

表 2.3 RSA 和 Diffie-Hellman 演算法的比較

第三章　系統核心設計

3.1 系統需求

系統需求規格書

功能 一 : 使用者身份驗證

Function	使用者身份驗證
Description	使用本系統時，系統會要求使用者先登入，使用者必須輸入帳號和密碼，系統驗證通過以後才會開放使用者的權限進入本系統 ;若是第一次使用本系統，則必須申請新的帳號及密碼才可使用本系統
Inputs	新使用者申請： 申請新帳號、密碼 使用者登入 ： 輸入帳號、密碼
Source	使用者的帳號、密碼都是由使用者輸入，但使用者帳號、密碼是否存在、正確皆由系統來驗證
Outputs	新使用者申請時，其資料會新增在系統內建的一個記錄使用者資料的檔案內，若其申請的帳號有重複等錯誤發生或者使用者登入帳號密碼有誤時，系統會蹦出錯誤訊息來通知使用者
Destination	目的在保護使用者的隱私
Requires	記錄使用者資料的檔案
Pre-condition	必須安裝本系統
Side-effects	無

功能 二 : 加密與解密

Function	加密解密
Description	通過系統驗證以後，會進入本系統的核心，每個使用者可以對尚未被他人加密的檔案或資料夾加解密
Inputs	對檔案作加密、解密 對屬於使用者的加解密檔案作修改、存取
Source	對任何檔案的加解密都是由使用者來執行，但是使用者是否有權限由系統來判定
Outputs	加解密任何檔案都會產生一個和原檔相關聯的案，而對於他人的機密檔案，系統不開放權限給予使用
Destination	目的在保護機密檔案
Pre-condition	必須先有使用者驗證通過的步驟
Side-effects	無

以下將針對系統的功能性需求和非功能性需求作討論

Functional requirements

1．系統操作流程必須讓使用者能非常流暢的完成，每個按鈕、工具名稱盡可能讓使用者一看就懂，不要用容易讓使用者誤會的文字。

2．系統對於所有他人尚未加密的檔案都能執行加解密的功能。

3．每個使用者都必須有屬於自己登入系統時的帳號(ID)、密碼(PW)，而系統必須將所有使用者的 ID、PW 另外儲存在一個文件內並且保護。

4．系統另提供自解檔的功能，在傳送機密檔案時即使對方電腦中沒安裝本系統也可享有加解密的功能。

Non-Functional requirements

1．作業系統需求　2000pro／ 2000server／ XP
記憶體空間需求大小　3 MB

2．系統開發使用的程式語言是 Visual C++，必需具備視窗程式、加解密以及 windows 中 Registry 等觀念。

3．除安裝本系統來執行相關加解密運作以外，也可利用自解檔的功能。

4．本系統設計不可透露任何和使用者有關的資料。

3.2 系統核心設計

3.2－1 使用者權限問題

在系統設計上，首先，系統會先要求你輸入使用者名稱和密碼。當使用者輸入的密碼和原先不符時，將無法對檔案或資料夾進行加解密。而在實作上，KEY 的產生和使用者的密碼產生關聯性。當使用者輸入密碼後，密碼會經由 hash function 產生一個 hash value，然後再將此 hash value 當作是 key 要產生的一個必備參數來產生 key。所以當一個非法的使用者想要解密一個不是自己所加密的檔案時，將導致加解密失敗。

由於 Microsoft 所提供的加解密函式絕大多數均會和系統登錄的使用者產生關聯性，這將造成的這將造成在 A 電腦中加密的檔案無法移到 B 電腦中解密。最後所採用的做法便是將使用者的 KEY 經由加密後存放到檔案中的某個位置，在解密時先將 KEY 解密，在用此 KEY 來解密。

3.2－2 加密使用之 API

Cryptography API（Crypto API）簡介

微軟最先在 1996 關注於密碼系統問題，之後發展出 Crypto API（1.0 and 2.0）和 CAPICOM 兩套密碼系統。在 windows 系統中藉由 advapi32.dll 提供我們 Crypto API 函式庫，這個函式庫提供應用程式開發者在 Win32 環境下使用加密、驗證等安全服務時的標準加密介面，並且包裝一些複雜的演算法，免除了我們自行實做上的困難；透過 Crypto API，我們可以輕易的保護所需保護的資料。

目前 Crypto API 分為 Crypto API 1.0 和 Crypto API 2.0 2 個版本，描述如下： Crypto API 1.0 提供包含對密碼系統的支援、密鑰交換、管理、Cryptographic Service Providers（CSP）。Crypto API 2.0 則添加了對證書管理的支援、Cryptographic Service Provider 系統結構的可插入結構並可以通過使用不同的 CSP 來擴展不同的安全功能。目前支援 Crypto API 的系統有 Windows 95 OSR2，Windows 98/Me、Windows SP3, Windows 2000/XP。

Crypto API 在系統中的地位

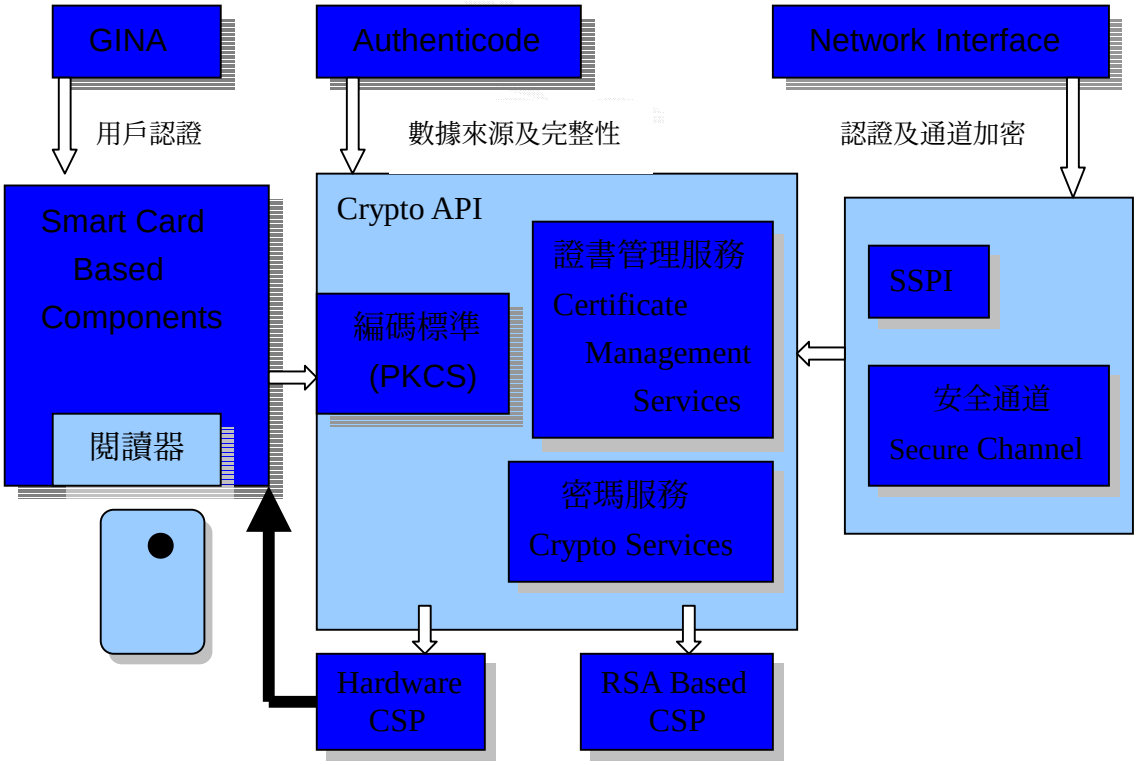


圖 3.1 Crypto API 在系統中的地位

Crypto API 的應用

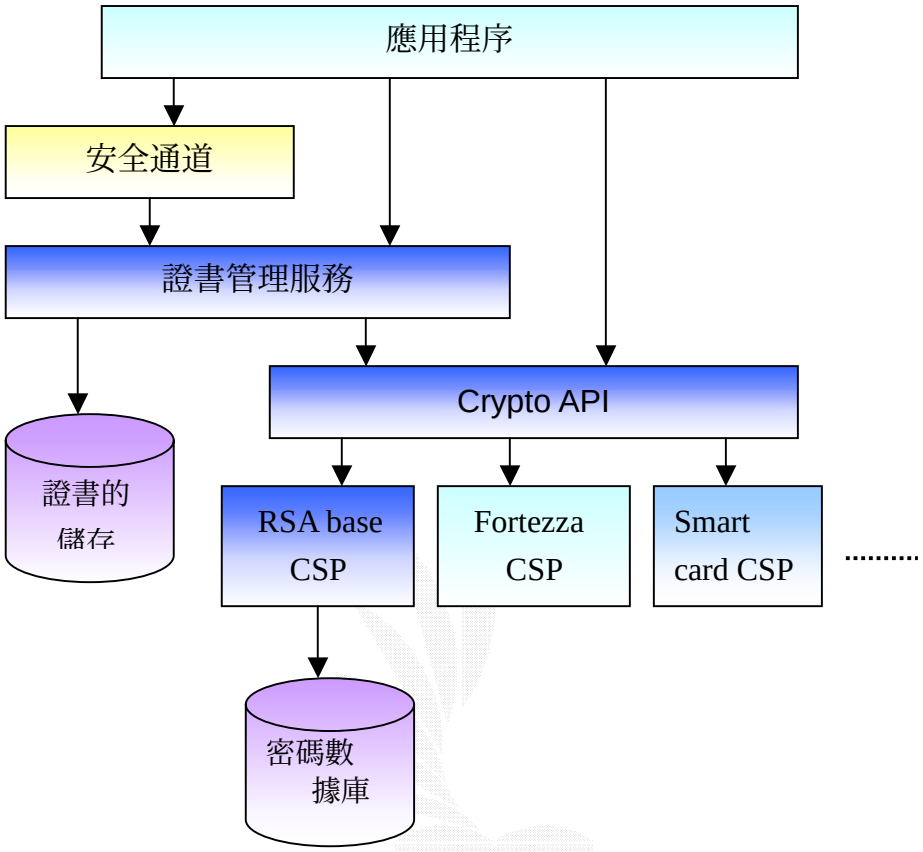


圖 3．2　Crypto API 的應用圖

Crypto API 的結構

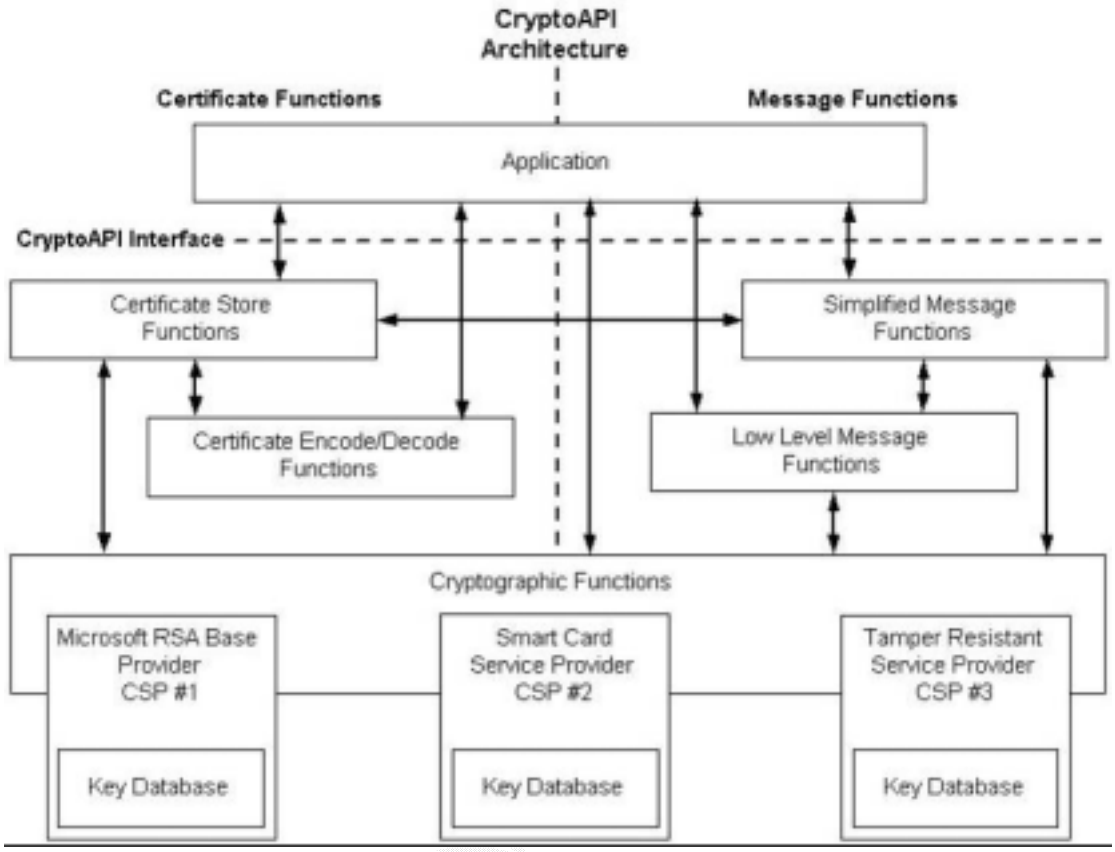


圖 3.3 Crypto API 的結構表

Crypto API 的功能是爲應用程式開發者提供在 Win32 環境下使用加密、驗證等安全服務時的標準加密介面。CryptoAPI 之上是應用程式，之下是 CSP（Cryptographic Service Provider）。CSP 是一個真正執行加密功能的獨立模組，典型的 CSP 有微軟 RSA Base Provider。目前任何一個加密服務提供者若想成爲微軟的合法的 CSP，就必須獲得微軟授予的一個簽名文件，該簽名文件保證了微軟 Crypto API 識別該 CSP。對於 Microsoft 的合法的 CSP，微軟會提供與其 Crypto API 介面的規範，介面的位置在圖中的“- - - -”線處。微軟提供的 CSP 安裝程式會將該 CSP 的各個文件安放到相應的目錄下，並在 Windows NT/98 註冊表中按 CSP 的類型和名稱爲該 CSP 註冊。Crypto API 使用系統註冊表存儲一個 CSP 資料庫，CSP 資料庫中記錄了所有已安裝到一個電腦中的 CSP。

Crypto API 的提供 5 類函數

＊Base cryptographic functions
- •Context functions: 連接 CSP
- •Key generation functions：生成和存儲密鑰
- •Key exchange functions：交換或傳輸密鑰

＊Certificate encode/decode functions

＊Certificate store functions

＊Simplified message functions, used to
- •encrypt and decrypt messages and data
- •sign messages and data
- •verify the authenticity of signatures on received messages and related data

＊Low-level message functions

關於 CSP(Cryptographic Service Provider)

由於編碼演算法很複雜，容易變動又常有專利權的問題，所以 Crypto API 通常以模組的架構來提供應用程式呼叫；支援 Crypto API 的模組又稱之為『編碼服務提供者』。CSP 是真正執行加密工作的獨立的模組，可分為下列來討論：

(1) 物理上一個 CSP 由兩部分組成：一個動態連結程式庫及一個簽名文件。

(2) 簽名文件保證提供者經過了認證，以防出現攻擊者冒充 CSP。

(3) 若加密演算法用硬體實現，則 CSP 還包括硬體裝置。

(4) Microsoft 使用 RSA Base Provider 在作業系統中提供一個 CSP，使用 RSA 公司的公鑰加密演算法，更多的 CSP 可以根據需要增加到應用中。

(5) 每個 CSP 都有一個名字和一個類型。每個 CSP 的名字是唯一的，而類型則不是。

(6) 一個 CSP 類型代表了一個特定的家族，這個家族決定了以下因素：
- 有且僅有一個密鑰交換演算法;
- 有且僅有一個簽名演算法;
- 特定的 Key Blob 格式;
- 特定的數位簽名格式;
- 特定的密鑰推導模式;

- 特定的密鑰長度;
- 特定的分組加密演算法的缺省模式

(7) 若兩個或多個應用間欲交換密鑰和加密的消息，那麼它們必須使用同一類型的 CSP。

下面表格表示預定的 CSP 類型

Provide Type	Key exchange	Signature	Encryption	Hashing
PROV_RSA_FULL	RSA	RSA	RC2, RC4	MD5, SHA
PROV_RSA_AES	RSA	RSA	RC2, RC4,AES	MD5, SHA
PROV_RSA_SIG	None	RSA	None	MD5, SHA
PROV_RSA_SCHANNEL	RSA	RSA	RC4,DES,3DES,+	MD5,SHA
PROV_DSS	None	DSS	None	MD5,SHA
PROV_DSS_DH	DH	DSS	CYLINK MEK	MD5,SHA
PROV_DH_SCHANNEL	DH	DSS	DES,3DES	MD5,SHA
PROV_FORTEZZA	KEA	DSS	Skipjack	SHA
PROV_MS_EXCHANGE	RSA	RSA	CAST	MD5
PROV_SSL	RSA	RSA	Varies	Varies

表 3．1 預定的 CSP 類型圖

關於 CryptoAPI 密鑰庫(Key Storage)

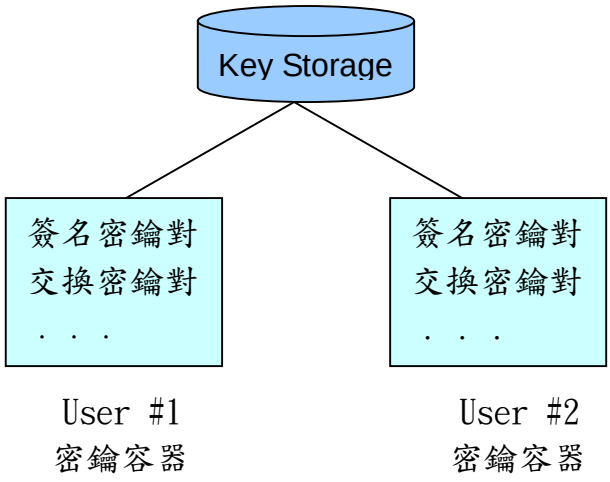


圖 3.4 密鑰庫

每個 CSP 有一個密鑰庫，而密鑰庫用於存儲密鑰。每個密鑰庫包括一個或多個密鑰容器（Key Containers）。每個密鑰容器中含屬於一個特定用戶的所有密鑰對。每個密鑰容器被賦予一個唯一的名字；以這個名字做函數 CryptAcquireContext 的參數，從而獲得指向這個密鑰容器的控制碼。CSP 將永久保存每一個密鑰容器，包括保存每個密鑰容器中的公/私鑰對，但在一個 Session 完後 CSP 將不保存這個 Session 在密鑰容器中的會話密鑰。(注: 一個 Session 是指從使用函數 CryptAcquireContext 到使用函數 CryptReleaseContext 期間的階段。當一個 Session 結束後，所有的使用 hProv 控制碼創建的會話密鑰和 Hash 值都變爲無效)。

關於加解密的密鑰

- 會話密鑰由應用程式用 CryptGenKey 或 CryptDeriveKey 函數創建
- 會話密鑰被安全保存在 CSP 中
- 不同於公私鑰對，會話密鑰一般更換頻繁。應用程式可以以加密的密鑰二進位大物件或 *key blob* 的形式從 CSP 中導出會話密鑰（用 CryptExportKey 函數），以供以後使用或傳輸給其他用戶。

在交換密鑰時，或密鑰需要離開 CSP(即導出密鑰)時，就存在選擇什麼樣的資料結構存儲密鑰的問題。微軟 Crypto API 採用 Key

Blob 資料結構存儲離開了 CSP 內部的密鑰。通常密鑰總是在 CSP 內部被安全地保存，應用程式只能通過控制碼訪問密鑰，而 Key Blob 則例外。當使用 CryptExportKey 函數從 CSP 中導出密鑰時，Key Blob 將要被創建。之後的某一時間，使用 CryptImportKey 函數將密鑰導入到某個 CSP 中(通常是另一個不同的機器上的不同的 CSP)。因此，Key Blob 是在不同的 CSP 之間安全傳送密鑰的儲存體。

Key Blob 有一個標準的資訊頭和位於資訊頭之後的一段表示密鑰本身的資料組成。通常應用程式不訪問 Key Blob 內部，而是把 Key Blob 當作一個透明的物件。由於公/私鑰對的私鑰部分需要絕對保密，所以私鑰要用對稱加密演算法加密。加密 Private Key Blob 時，除了 BLOBHEADER 之外的所有部分都要加密。但加密所用的演算法和密鑰（或密鑰參數）不與該 Key Blob 存儲在一起，應用程式負責管理這些資訊。

【注意事項】

(1) 如果你的機器曾運行過這種資料加密程式，那當然可以正確地運行。如果是第一次才運行此程式，那將會出現問題，錯誤就在 CryptAcquireContext 的使用上。在使用 CryptEncrypt 進行資料加密之前，需要取得當前機器所使用的密鑰容器，而 CryptAcquireContext（ & hProv , NULL , NULL , PROV _ RSA _ FULL , 0 ）就是用於連接所使用的 CSP，如果當前機器未曾設置過此的密鑰容器， CryptAcquireContext 使用就會出錯。因此必須爲機器創建缺少的密鑰容器。創建缺少的密鑰容器也需要調用 Crypto API 進行。下面介紹一種創建缺省的密鑰容器的控制臺程式 InitUser.c【附錄】

【附錄】本程式的功能爲：

1.用你的機器名稱 User Name 創建一個缺省的密鑰容器。

2.在密鑰容器中創建一個數位簽名密鑰對。

3.在密鑰容器中創建一個密鑰交換對。

本程式在一台機器上只須執行一次，執行之後，你將會看到在系統註冊表的 HKEY_CURRENT_USER\Software\microsoft 下多了一個主鍵： Cryptography\UserKeys，並填入一些十六進位的值。之

後，你就可以順利地使用資料加密程式了，甚至可以使用數位簽名了。

(2) 密鑰的物理特性：同時使用同一會話密鑰對兩個序列的資料進行加密或解密時，要物理上製作一份該會話密鑰的拷貝，即一個物理上的會話密鑰不得同時用於兩個操作。因爲每個會話密鑰都包含了內部狀態資訊，若它同時被用於多個操作，會造成混亂。

(3) 其加解密涵式主要為 CryptEncrypt()和 CryptDecrypt()，格式如下：

```
BOOL WINAPI CryptEncrypt(
    HCRYPTKEY   hKey ,
    HCRYPTHASH hHash ,
    BOOL Final ,
    DWORD dwFlags ,
    BYTE*   pbData ,
    DWORD*  pdwDataLen ,
    DWORD dwBufLen
    );
```

```
BOOL WINAPI CryptDecrypt(
    HCRYPTKEY hKey ,
    HCRYPTHASH hHash ,
    BOOL Final ,
    DWORD dwFlags ,
    BYTE* pbData ,
    DWORD* pdwDataLen
    );
```

其他加解密的函式

EncryptFile 和 DecryptFile 函式說明：

在程式開發過程中，經常遇到需要保護用戶資訊和私有資料不被他人竊取的情況，也就是要對資料進行加密。在 Win2000 以前，要對資料進行加密和解密操作，一般要使用 CryptAPI 函數，相當煩瑣。幸運的是，從 Win2000 開始，作業系統從內核的級別上提供了對文件進行加密和解密的函數，而且使用起來相當簡單。

加密的函數是

```
BOOL EncryptFile ( LPCTSTR lpFileName  ) //檔案名
```

參數 lpFileName 可以是檔案名，也可以是目錄名，如果是目錄名，那麼以後在該目錄下創建的文件都將被自動加密。如果函數執

行成功，將返回非零值，否則，返回值爲零，可以通過 GetLastError() 函數得到錯誤代碼。

解密函數是

```
BOOL DecryptFile ( LPCTSTR lpFileName , DWORD dwReserved  )
                    // 要解密的檔案名或目錄名  // 保留參數，必須爲 0
```

參數 lpFileName 如果是目錄名，則該目錄下的所有文件將被解密，並且以後在該目錄下創建的文件不再被自動加密。如果函數執行成功，將返回非零值，否則，返回值爲零，可以通過 GetLastError()函數得到錯誤代碼。

經過加密的文件，在 Explorer 中會顯示爲具有“加密”屬性。其實，你也可以在 Explorer 中，選中文件->右鍵功能表->查看屬性->高級，然後選擇“加密內容以保護資料”，它們實現的功能是完全一樣的

缺點

(1) 這兩個函數實際上是把文件系統和 Windows 的系統帳戶綑綁起來，用戶加密文件後，自己仍然可以隨意查看（不需要先解密）；但具有相同許可權級別的其他用戶不能查看，當他們打開你加過密的文件時，將得到“拒絕訪問”的警告資訊。還要注意的是，比你許可權級別高的用戶（超級管理員），仍然可以隨意查看你的文件。

(2) 只能用於 Windows 2000 Professional，windows 2000 server，Window XP，Windows server 2003。

(3) 此加解密涵式只能用於 NTFS 格式。也就是說非 NTFS 格式不具此功能。

DPAPI(Data Protection API)

我們已經知道,用 Win32API 函數 EncryptFile 和 DecryptFile 可以方便的加解密文件。但是，這兩個函數與系統帳戶聯繫過於緊密，能否解密完全看當前帳戶是否有更高的許可權，用戶的資料實際上並沒有得到真正的保護。下面將介紹一種更爲靈活的資料保護方法。

從 Win2000 開始，作業系統開始提供一個名爲 Data Protection API (DPAPI)的資料保護介面。該介面一共有兩個函數，他們提供了系統級的資料保護服務。這兩個函數存在於 Crypt32.dll 庫中，是 Crypt API 的一部分，但使用起來要比其他的 Crypt API 函數簡單得多。

DPAPI 可以實現資料加密和解密。也就是說我們提供一個密碼用於加密，而其他人只有知道這個密碼才能解密。實際上，DPAPI 上在後臺爲我們完成了相當複雜的加密解密操作，包括密鑰的產生、儲存、使用等。以下為此函式介紹

DPAPI 加密函數

```
BOOL WINAPI CryptProtectData (
    [IN] DATA_BLOB                      *pDataIn,
    [IN] LPCWSTR                         szDataDescr,
    [IN] DATA_BLOB                      *pOptionalEntropy,
    [IN] PVOID                           pvReserved,
    [IN] CRYPTPROTECT_PROMPTSTRUCT      *pPromptStruct,
    [IN] DWORD                           dwFlags,
    [OUT] DATA_BLOB                     *pDataOut )
```

其中*pDataIn 為輸入資料(明文),szDataDescr 為所要描述資訊,*pOptionalEntropy 為額外保護資訊,pvReserved 為保留參數，必須爲 NULL,*pPromptStruct 為提示對話方塊結構,dwFlags 為標誌位元,*pDataOut 為輸出資料(密文)。

輸入資料參數 pDataIn 是一個 DATA_BLOB 結構，該結構的定義如下：

```
typedef struct _CRYPTOAPI_BLOB {
            DWORD     cbData;       //資料的長度
            BYTE*      pbData;     //指向資料的指標
            }DATA_BLOB
```

參數 szDataDescr 是一個字串,可以是關於加密的描述資訊或其他的任何資訊,但不能爲 NULL,該資訊將以明文的方式存在於最終輸出的密文資料中。

參數 pOptionalEntropy 用於額外的密碼保護,本文後面將有詳細的介紹。

參數 pPromptStruct 用於指定一個安全提示對話方塊,在加密解密操作時,將彈出該對話方塊提示用戶正在進行安全操作。如果該參數爲 NULL,則不會彈出對話方塊。後面的程式將把該參數設置爲 NULL;參數 dwFlags 是關於加密的一些選項標誌,一般設置爲 0 就可以了。

參數 pDataOut 是輸出資料,同樣是一個 DATA_BLOB 結構,但應該注意的是 pDataOut->pbData 所指向的記憶體是由系統分配的,在使用完輸出資料後應該用 LocalFree 函數釋放該記憶體。

DPAPI 解密函數

```
BOOL WINAPI CryptUnprotectData (
    [IN] DATA_BLOB                          *pDataIn,
    [OUT] LPCWSTR                           *ppszDataDescr,
    [IN] DATA_BLOB                          *pOptionalEntropy,
    [IN] PVOID                               pvReserved,
    [IN] CRYPTPROTECT_PROMPTSTRUCT           *pPromptStruct,
    [IN] DWORD                               dwFlags,
    [OUT] DATA_BLOB                          *pDataOut )
```

其中:*pDataIn 為輸入資料(密文),*ppszDataDescr 為輸出描述資訊,*pOptionalEntropy 為額外的保護資訊,pvReserved,為保留參數必須爲 NULL,
*pPromptStruct 為提示對話方塊結構,dwFlags 為標誌位元,
*pDataOut 為輸出資料(明文)。

解密函數的參數與加密函數的參數基本類似,唯一需要指出的是參數 ppszDataDescr 在這裏應該被指定爲一個指標變數,在解密完成後,該變數將指向加密時指定的描述資訊,最後應該使用 LocalFree 函數釋放該字串。如果你不想得到描述資訊,可以將該參數設置爲 NULL。

DPAPI 和系統帳戶聯繫仍然十分緊密,他會自動使用當前用戶的系統登錄使用者爲加解密的密碼,這樣一來同一台機器的所有程式都可以解密其他程式加密的資料。爲了防止這一點,函數提供了

pOptionalEntropy 參數,使我們有機會使用自己的密碼。如果提供了 pOptionalEntropy 參數，DPAPI 將使用當前用戶系統登錄密碼和我們提供的額外保護密碼的組合進行加解密操作。如果你不想使用額外密碼保護，則可設置該參數爲 NULL。

缺點

(1)因爲 DPAPI 使用用戶帳戶聯繫的，所以一台機器上加密的資料，一般不能在另一台機器上解密。

SAFEncrypt

加密函式如下：

```
SAFEncrypt.EncryptFile(
[IN]sEncryptionKey，     //為加密所使用的 key
[IN]sInputFile，          //所要加密的檔案名稱
[IN]sEncryptedFile       //加密完後的檔案名稱
)
```

解密函式如下：

```
SAFEncrypt.DecryptFile(
sEncryptionKey，      //為解密所使用的 key
sInputFile，          //所要解密的檔案名稱
sDecryptedFile           //解密完後的檔案名稱）
```

缺點

(1)只能於 Windows XP，Windows server 2003 執行。

3.2－3 滑鼠右鍵蹦出式選單功能

右鍵蹦出式選單功能簡介：當我們在 windows 系統中的任何地方按一下滑鼠右鍵，發現會蹦現出許多的選單功能。或者當你在安裝某軟體後，會發現右鍵選單中亦多了安裝軟體的功能選項加入其中。這些功能是如何做出的，其實就是修改 windows 的 registry 來達到上述所說的功能。

目前我們做出的功能有：

用滑鼠點選檔案後再按一下滑鼠右鍵，即可對檔案做加解密的動作。和用滑鼠點選資料夾後再按一下滑鼠右鍵，即可對資料夾做加解密的動作。

(1) 如何在滑鼠右鍵蹦出式選單功能中顯示對檔案加解密的選項：

打開 windows 登錄編輯器，在 HKEY_LOCAL_MACHINE\SOFTWAR\Classes\Shell\欲顯示在右鍵的名稱\command\欲執行的檔案位置。其中”欲顯示在右鍵的名稱”即是打入你在滑鼠右鍵蹦出式選單中的名稱。而”欲執行的檔案位置”則是打入你想要執行的檔案路徑。或者也可以在原本”欲顯示在右鍵的名稱”這裡新增一個機碼來取代，然後在去修改這個機碼的字串值。此字串值就是你想要顯示在滑鼠右鍵蹦討選單中的名稱。

【注意 1】HKEY_LOCAL_MACHINE\SOFTWAR\Classes\Shell\欲顯示在右鍵的名稱\command\欲執行的檔案位置中所顯示的機碼和在 HKEY_CLASSES_ROOT*\shell\所儲存的內容是一樣的。

【注意 2】此外尚需要在路徑名稱後加一個%1 的參數，如("C:\encrypt_momo.exe" " %1")。加上這個%1 的參數才能取得你目前所選取的檔案。

(2) 如何在滑鼠右鍵蹦出式選單功能中顯示對資料夾加解密的選項：

打開 windows 登錄編輯器，HKEY_CLASSES_ROOT\Directory\shell\欲顯示的名稱\command\執行檔案的位置。其中”欲顯示在右鍵的名稱”即是打入你在滑鼠右鍵蹦出式選單中的名稱。而”欲執行的檔案位置”則是打入你想要執行的檔案路徑。這個機碼和 HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Directory\shell 的內容是一樣的。

【注意 1】HKEY_CLASSES_ROOT\Directory\shell\欲顯示的名稱\command\執行檔案的位置和 HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Directory\shell 的內容是一樣的。

【注意 2】此外尚需要在路徑名稱後加一個%1 的參數，如("C:\encrypt_momo.exe" " %1")。加上這個%1 的參數才能取得你目前所選取的資料夾。

3.2－4 自解檔之應用

為了提高機密檔案的可攜性，在無安裝本軟體的情況下也能對檔案作保護，因此增加了自解檔的功能。而自解檔的製作原理在於了解執行檔的標頭的意義，進而將程式區和資料區的位址移動或者是寫入。

其檔案格式為

執行檔區
記錄區
資料區

自解檔可分成三個區塊，分別為執行檔區、資料區、記錄區。

分別敘述如下：

執行檔區：負責讀取記錄區中的記錄來判斷資料區的位址、檔名及一些額外的處理。

資料區　：存放原本資料的地方。

記錄區　：記錄原本檔案名稱、原檔的長度、檔案經過合併過後的檔案長度。

其製作流程如下圖所示：

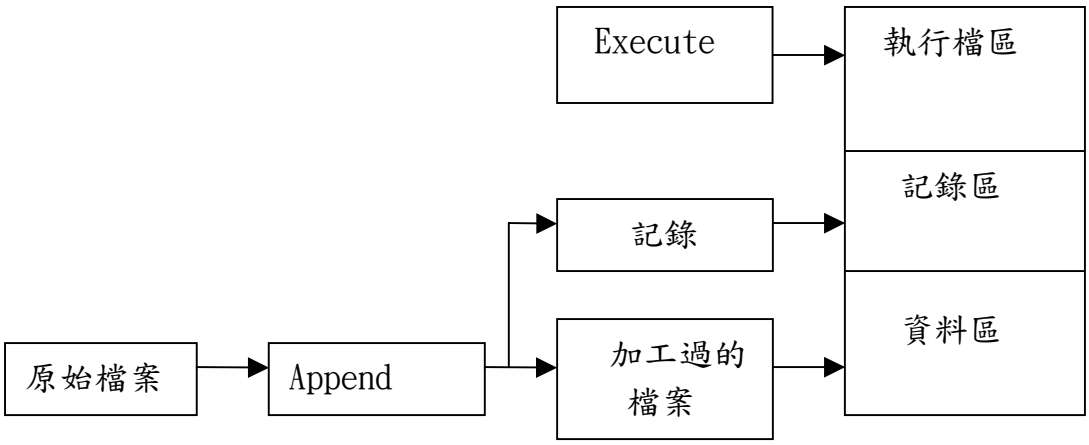


圖 3.5 自解檔流程圖

3.3 系統流程

系統執行加密流程

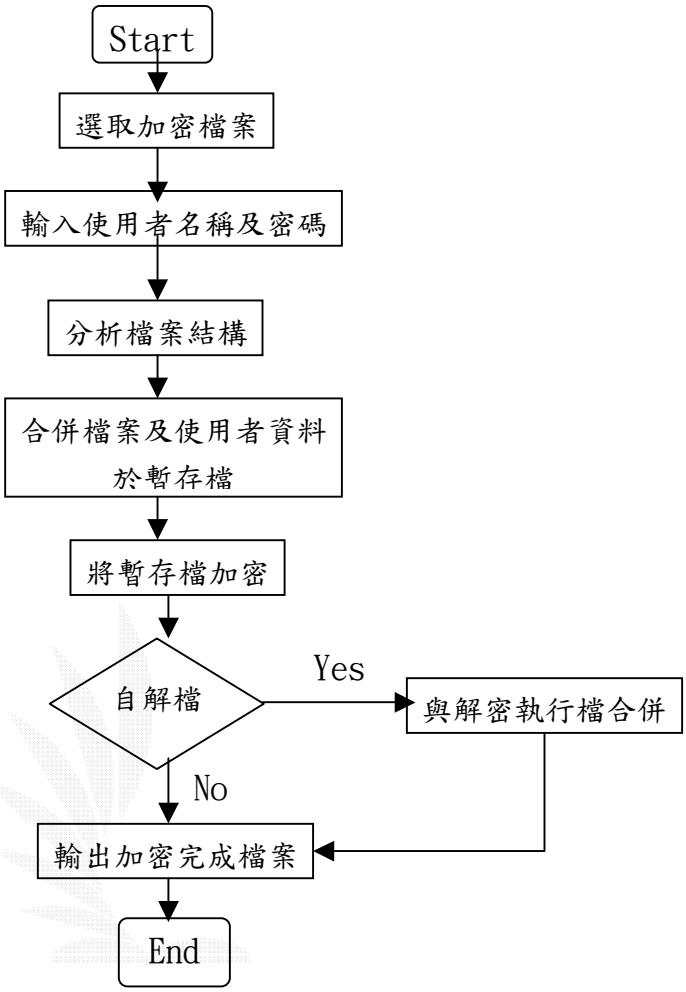


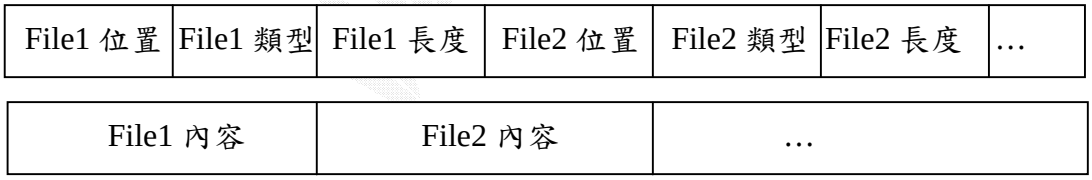
圖 3.6 系統加密流程圖

步驟 1： 在要加密的檔案或資料夾上點選滑鼠右鍵，然後選取選單上的加密選項，進入步驟 2。

步驟 2： 如果要新增一個使用者的話，則點選「新增使用者」的按鈕進入步驟 3。進入加密程式後，輸入使用者名稱及使用者密碼，以及加密後的檔案名稱(預設為加密的檔案.enc)，並且選取是否加密成「自解檔」，按下加密鈕，將輸入的使用者資料與記錄中的使用者資料做比對，如果正確則進入步驟 4，否則回到步驟 2。

步驟 3： 新增一組新的使用者名稱及密碼，輸入使用者名稱及密碼如果欲新增的使用者名稱與舊有所在的使用者名稱沒有重覆，則新增成功，進入步驟 4，否則回到步驟 3。

步驟 4： 分析欲加密的目標的「資料夾結構」、「檔案個數」及「檔案所在的位置」，並將其合併於一個暫存檔案中，如下圖。接著進入步驟 5。



步驟 5： 將合併好的 temp 透過加密的 API，輸出檔案至一個暫存檔中，如果在步驟 2 時有選取加密成「自解檔」則進入步驟 6，否則進入步驟 7。

步驟 6： 將加密好的程式製作成「自解檔」，將解密的執行檔與加密過後的資料合併起來，如下圖。接著進入步驟 7。

步驟 7： 產生加密結果的最終檔案，並且刪除中間過程的暫存檔。

系統執行解密流程

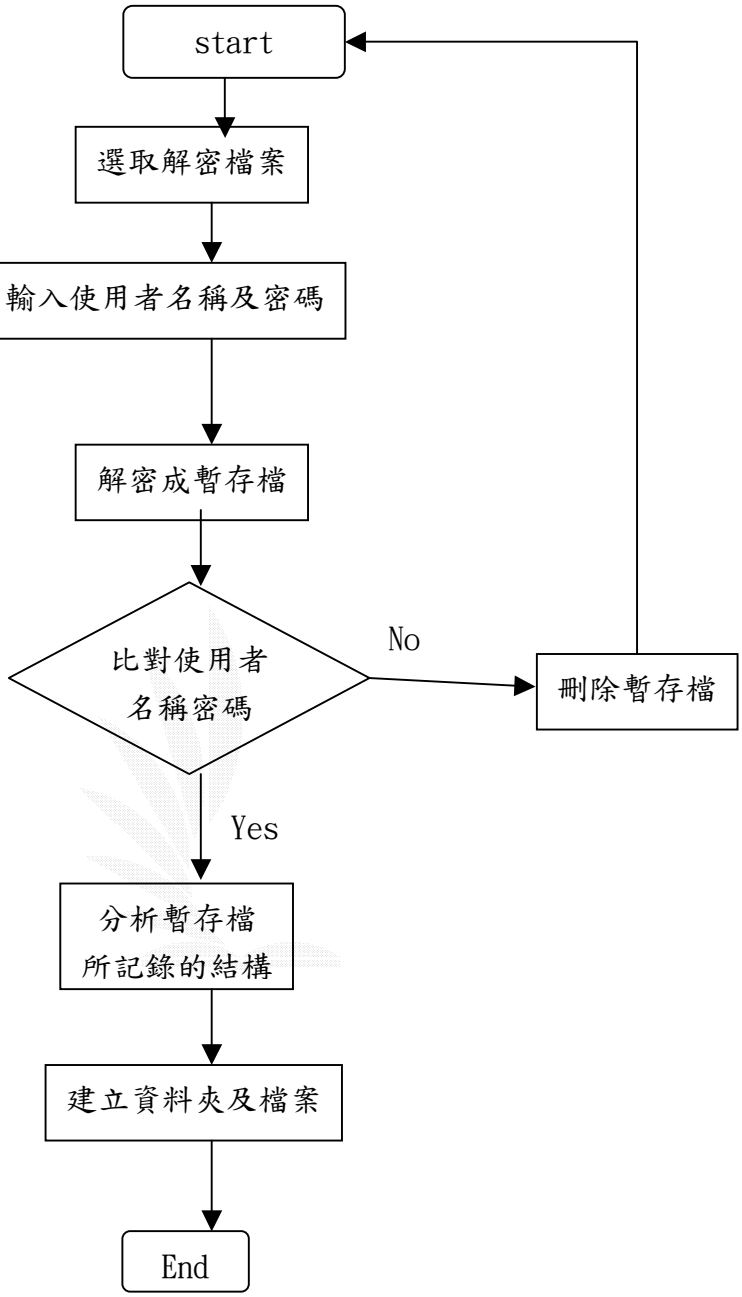


圖 3．7 系統解密流程圖

步驟 1：在要解密的檔案上點選滑鼠右鍵，然後選取選單上的解密選項，進入步驟 2。

步驟 2：進入解密程式後，輸入使用者名稱及使用者密碼，按下解密鈕，將輸入的使用者資料與記錄中的使用者資料做比對，如果正確則進入步驟 3，否則回到步驟 2。

步驟 3：將該檔案透過解密的 API，輸出檔案至一個暫存檔中。進入步驟 4。

步驟 4：分析該暫存檔，檔案格式如下圖，並且記錄下相關資料，以便還原加密前的資料夾結構及檔案內容。進入步驟 5。

File1 位置	File1 類型	File1 長度	File2 位置	File2 類型	File2 長度	…

File1 內容	File2 內容	…

步驟 5：利用步驟 4 取得的相關資料，將資料夾的結構及檔案內容建立出來，並且刪除掉暫存檔。

系統製作解自解檔流程

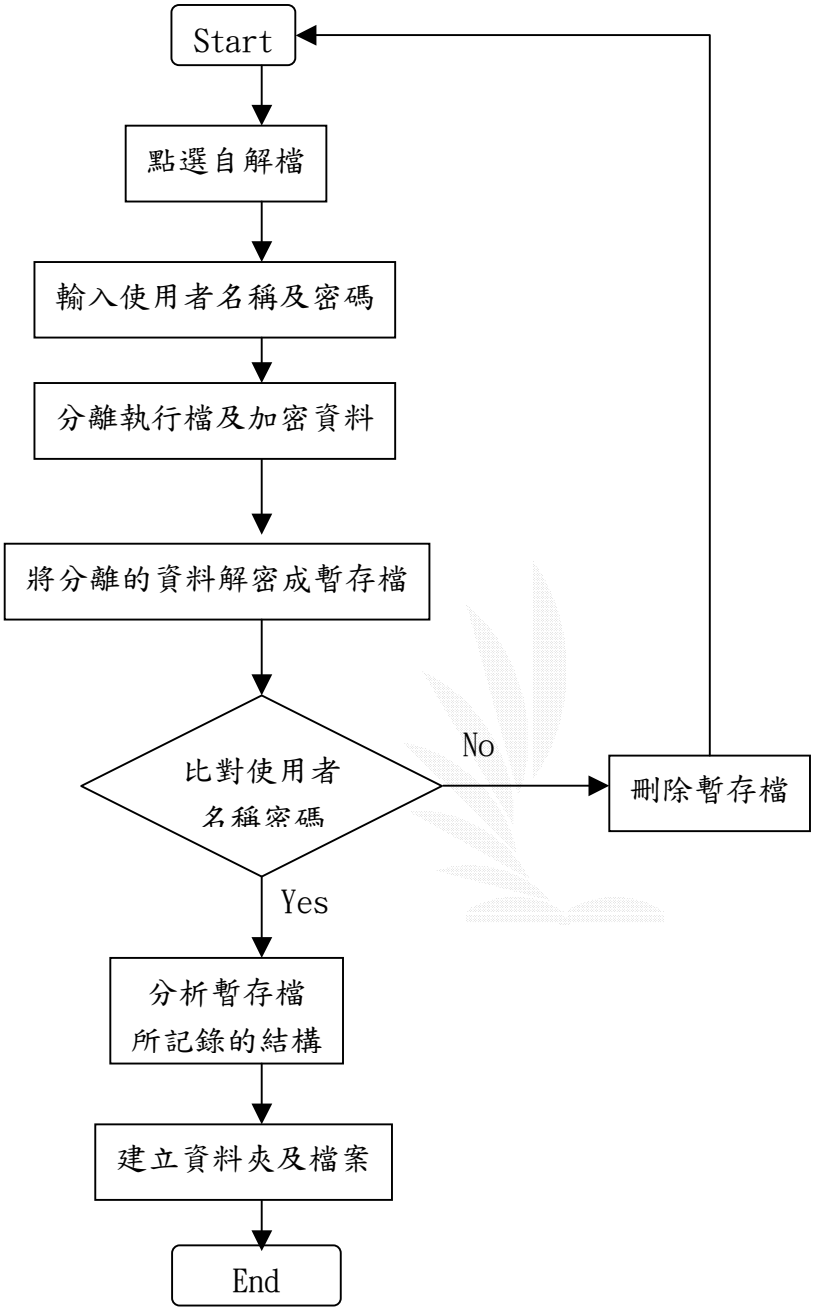


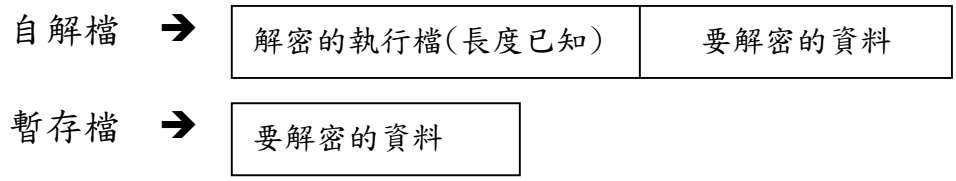
圖 3.8 系統制作自解檔流程圖

步驟 1：點選加密完成的「自解檔」，進入步驟 2。

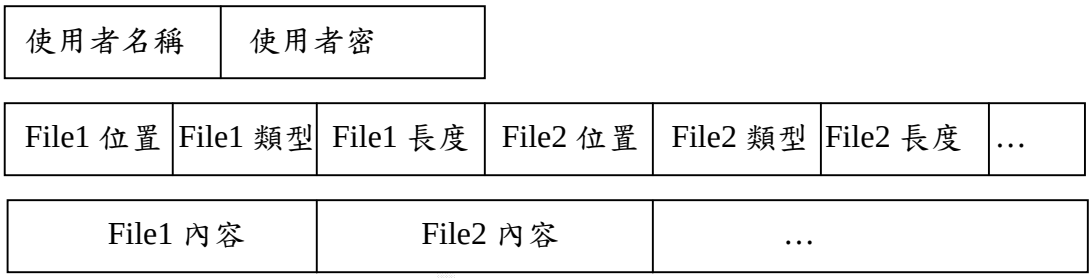
步驟 2：進入「自解檔」解密程式後，輸入使用者名稱及使用者密碼，按下解密鈕，則進入步驟 3。

步驟 3：將解密執行檔與要解密的資料內容分離出來，把資料內容存

在一個暫存檔中，如下圖。進入步驟 4。



步驟 4：分解後的資料格式如下圖，比對使用者資料是否正確，如果正確進入步驟 5，否則刪除掉暫存檔並且進入到步驟 2。



步驟 5：分析該暫存檔，檔案格式如上圖，並且記錄下相關資料，以便還原加密前的資料夾結構及檔案內容。進入步驟 6。

步驟 6：利用步驟 5 取得的相關資料，將資料夾的結構及檔案內容建立出來，並且刪除掉暫存檔。

第四章　系統使用說明與未來發展

4.1　系統使用說明

4.1－1　對檔案加密的執行步驟

4.1－2　對檔案解密的執行步驟

4.1－3　對資料夾加密的執行步驟

4.1－4　加密自解檔的執行步驟

4.1－5　解密自解檔的執行步驟

4.1－6　系統另一執行介面

4.2　系統未來可行性發展

4.1 系統使用說明

4.1－1 對檔案加密的執行步驟

step 1： 在檔案 20030519_20060059786_063148.jpg 上按下滑鼠右鍵選取加密。

圖 4.1 檔案加密(一)

step 2：輸入使用者名稱及密碼，並且輸入加密後的檔名。

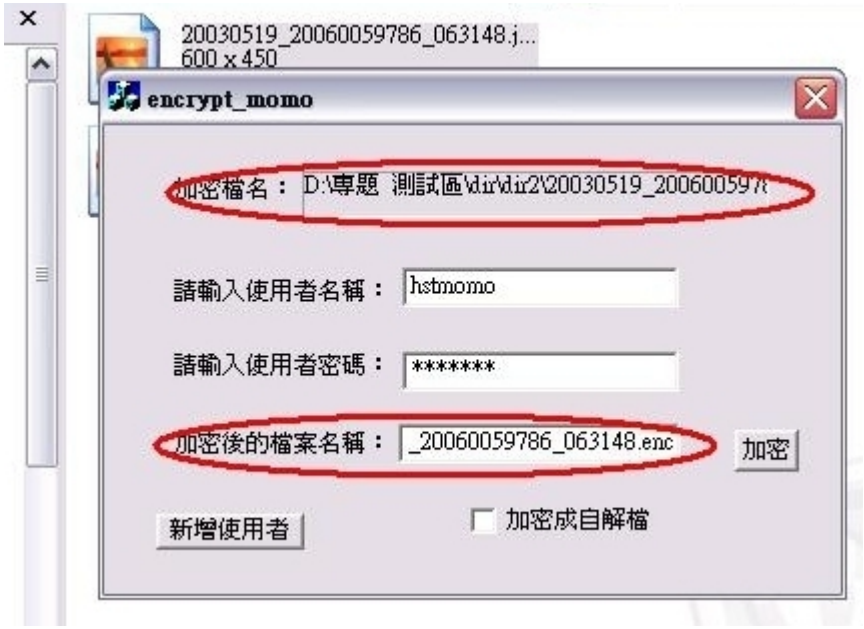


圖 4.2 檔案加密(二)

step 3：點選加密鈕後，會出現加密成功的訊息。

step 4：產生加密後的檔案。

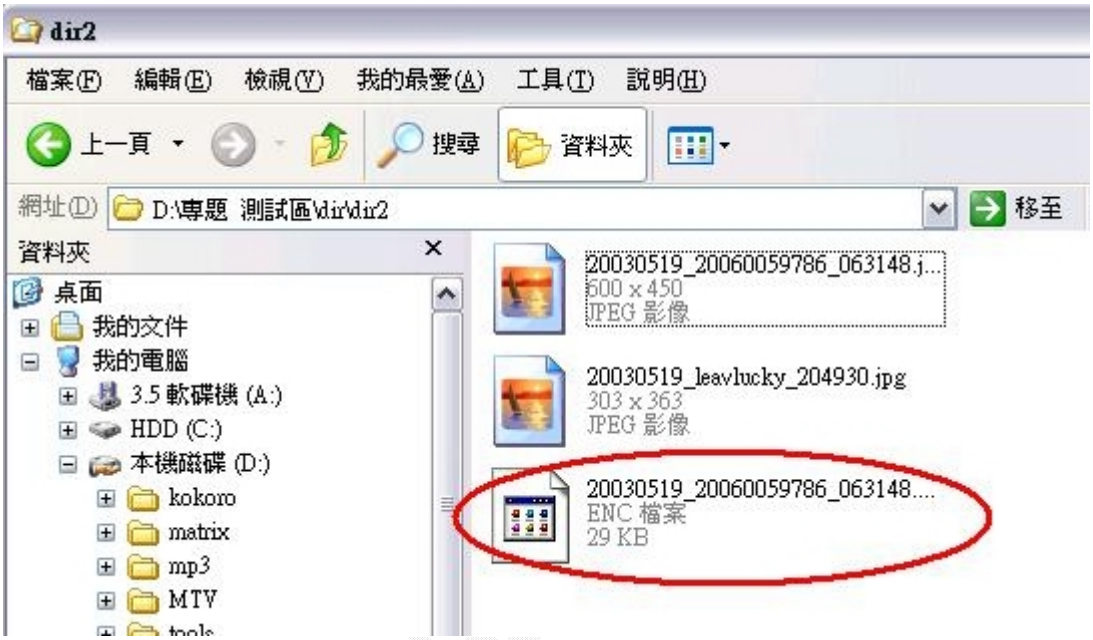


圖 4．3 檔案加密(三)

4.1－2 對檔案解密的執行步驟

step 1 ： 在檔案 20030519_20060059786_063148.enc 上按下右鍵選取解密。

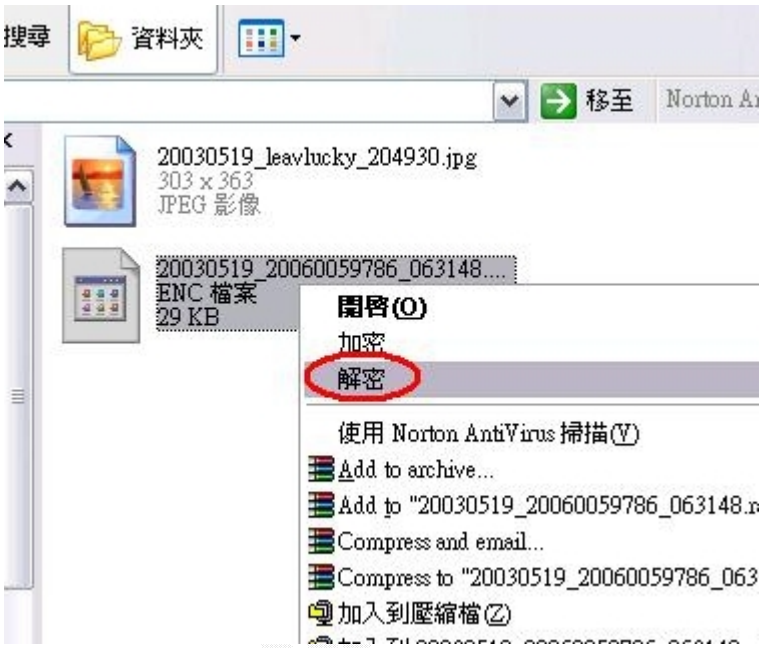


圖 4.4 檔案解密(一)

step 2 ： 輸入使用者名稱及密碼。

圖 4.5 檔案解密(二)

Step 3 ： 點選解密鈕後，出現解密成功的訊息。

圖 4．6 檔案解密(三)

Step 4 ： 產生解密後的檔案。

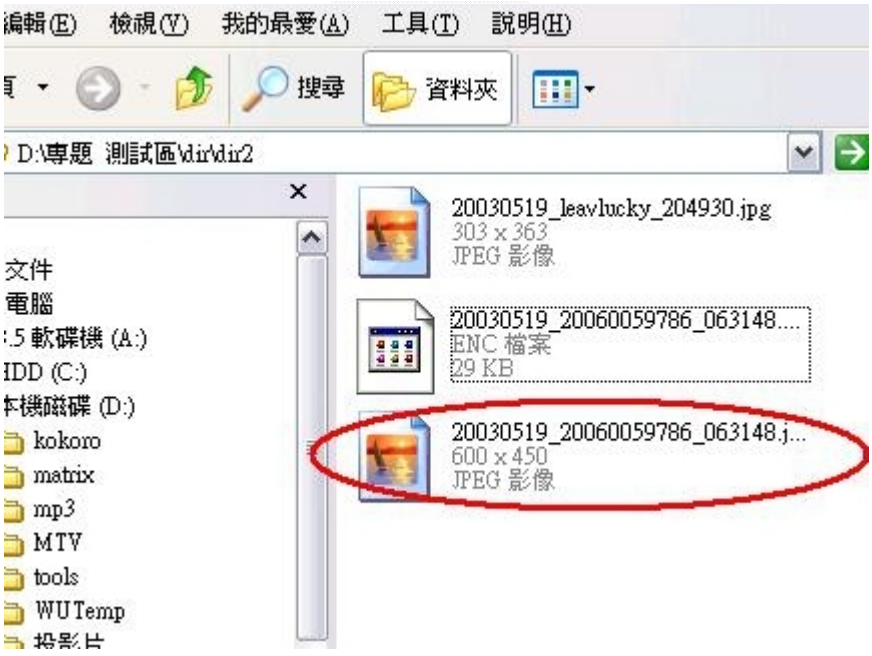


圖 4．7 檔案解密(四)

4.1－3 對資料夾加解密的執行步驟

如同前面的步驟，選擇資料夾後，按下滑鼠右鍵做加解密的動作即可。

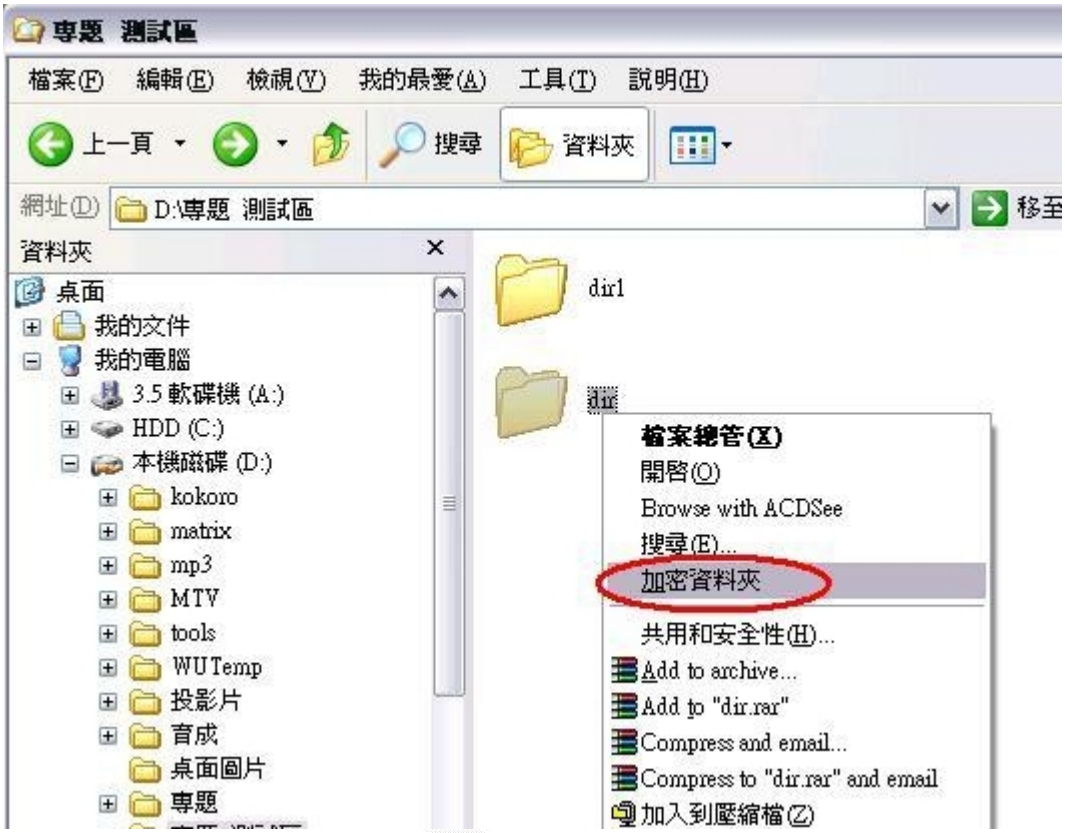


圖 4.8 資料夾加密

4.1－4 加密自解檔的執行步驟

Step 1 :在資料夾 dir2 上按下右鍵選取加密資料夾。

圖 4.9 加密自解檔(一)

Step 2 : 輸入使用者名稱及密碼，並且勾選加密成自解檔。

圖 4.10 加密自解檔(二)

Step 3 ： 點選加密鈕後，出現加密成功的訊息。

圖 4．11 加密自解檔(三)

Step 4 ： 產生加密後的自解檔案。

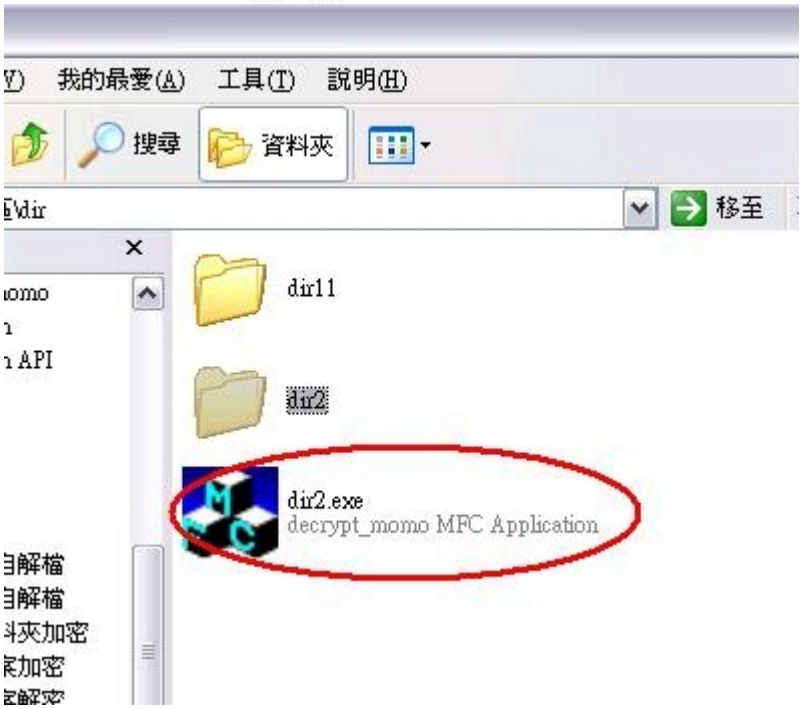


圖 4．12 加密自解檔(四)

4.1－5 解密自解檔的執行步驟

Step 1 ： 點選 dir2.exe。

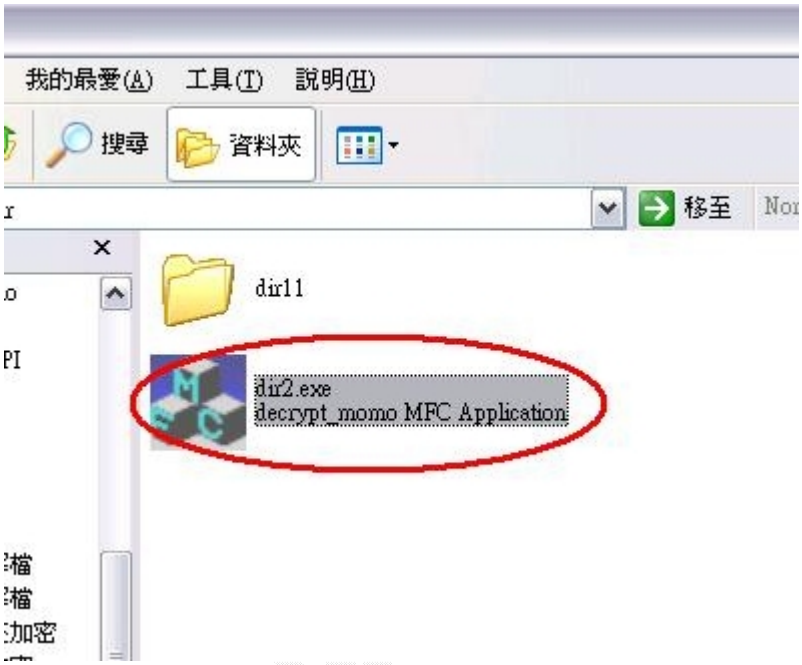


圖 4.13 解密自解檔(一)

Step 2 ： 輸入使用者名稱及密碼。

圖 4.14 解密自解檔(二)

Step 3 ： 點選解密鈕後，出現解密成功的訊息。

圖 4．15 解密自解檔(三)

Step 4 ： 產生解密後的檔案。

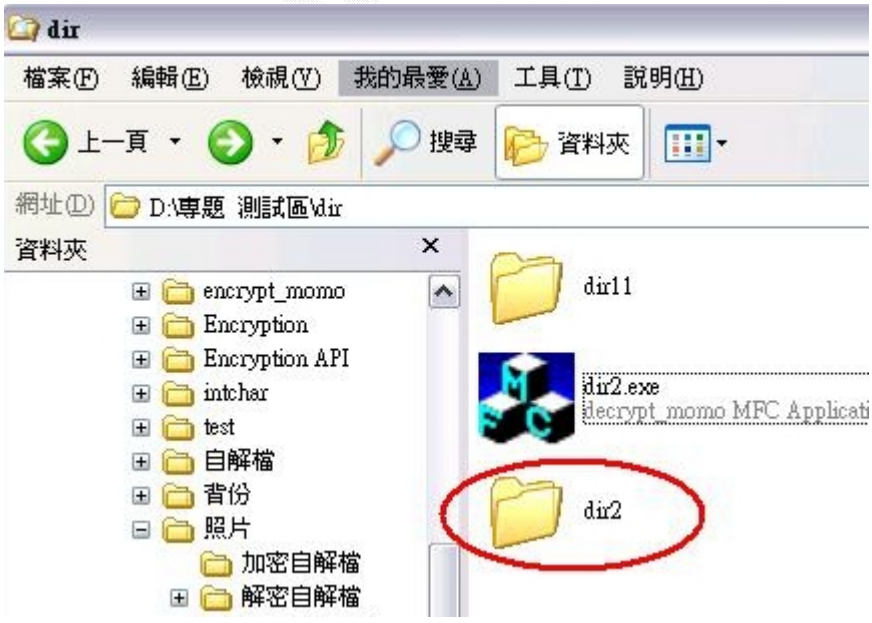


圖 4．16 解密自解檔(四)

4.1－6　系統另一執行介面

下圖為本系統另一個介面，其執行方式和上面所介紹的步驟一樣，唯一不同在於此介面可同時選取多個檔案或資料夾進行加解密等動作。

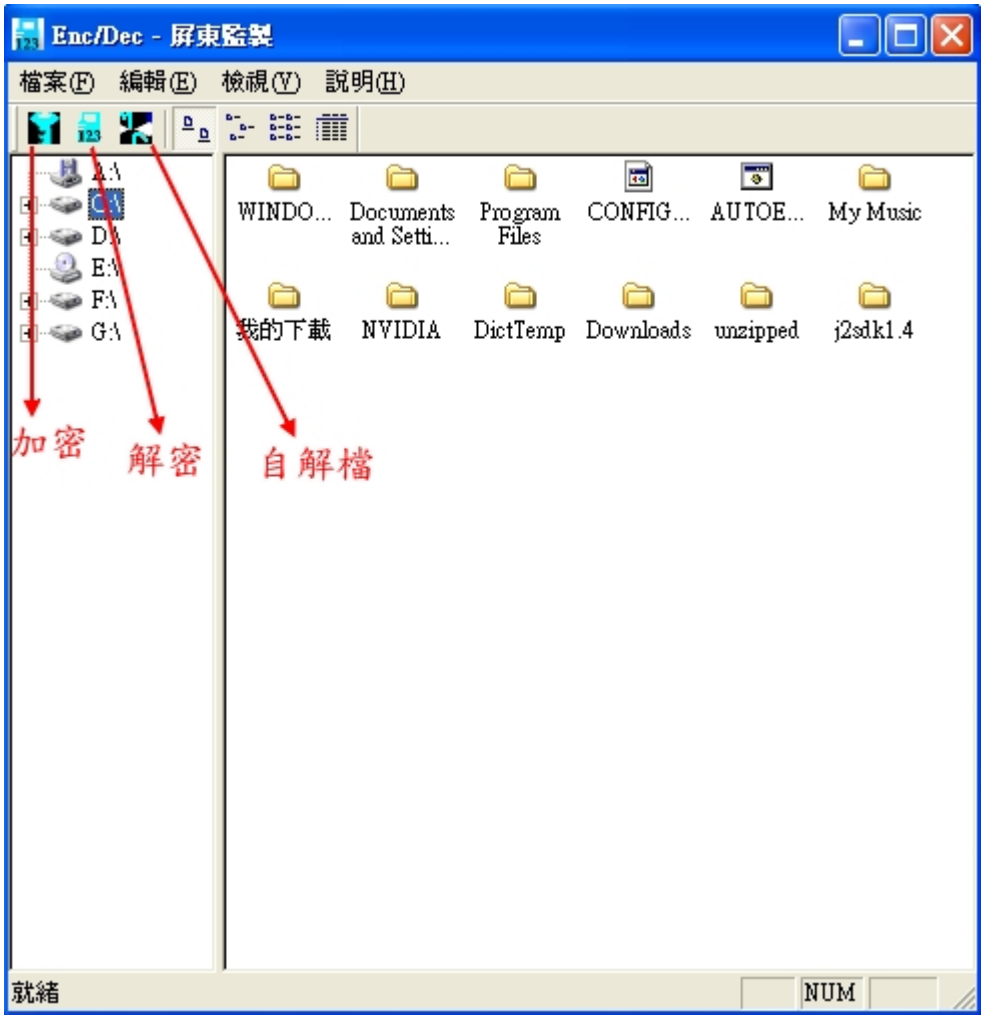


圖 4.17 系統另一介面

4.2 系統未來可行性發展

由於系統開發時碰到一些目前尚無法解決的困難，而且有些地方考慮得不夠周詳，因此所做出來的系統仍有少許不齊全的地方，未來將繼續朝下列方向努力和改良：

1. 加密資料在網路上傳遞時該運用哪些機制來確保資料能安全、不被破壞或修改的傳輸到目的端？

2. 在個人電腦中和公用電腦中所加密的資料該如何做更有效的作區別和保護？

3. 持續改善現有介面，提供一個讓使用者更便利的加密環境。

4. 改善滑鼠的拖曳(drag-and-drop)功能。

5. 結合資料隱藏的技術，增加系統功能的選擇性。

第五章　專題回顧

-組員心得分享

5.1　呂宜峻的心得分享

5.2　邱紹華的心得分享

5.3　蘇淑瑛的心得分享

5.1 呂宜峻的心得分享

這次專題選擇以密碼學來當作題目，對於我們來說可以是一個新的挑戰。起初老師所賦予我們的題目是要求我們去尋找如何在按下滑鼠右鍵後所出現的蹦現選單中加入我們想要的標題。這個技術我們從以前到現在都沒接觸過，於是便開始到各個論壇、討論區發問。最後我們得知這個技術是和 windows registry 有關係，於是我們便開始著手研究有關 windows registry 相關知識和架構（此部份正是我們專題報告的第一部份）。對於 windows registry 的認知我們還不算是很深入，只有將其基本的架構組織弄明白。在此將我們所理解的部分寫成報告，方便以後學弟能從此學到一些東西。

從找資料到學會這個技術約莫花了 2~3 個月，完成後老師便要求我們做一些有用的軟體出來。於是我們便將此技術和密碼學做結合，發展一套關於加解密的軟體，方便使用者對個人電腦或公用電腦中的檔案做保護。然而在大學 4 年幾乎沒有修過有關密碼學或資訊安全的課，於是這又是利一個挑戰的開始。

一開始先找有關於密碼學相關的資料，來認識密碼學的相關領域和原理。有餘加解密程式的演變相當快速，因此老師要求我們不必自己開發新的加解密程式，而是要去找一套安全解被廣泛使用的加解密程式。於是我們選擇使用微軟的 Cryptography API 來當作我們加解密的依據。關於 Cryptography API 的介紹和使用方式我們也在專題報告中做了詳細的描述，希望能幫助以後的學弟便於學習此套 API 的操作。為了要發展出一套較可用的軟體於是我們選擇用 vc++來開發。一開始對於學習 vc++中的 MFC 非常不順利，因為 MFC 是屬於視窗程式設計的開發工具，必須要對物件導向語言和 windows programming 有深厚的認識，因此我們在此摸索了很久。也遇到了很多關於設計視窗程式方面的難題。再透過不斷的和組員討論以及到各大論壇詢問後我們的問題使能夠一一克服。

‘溝通’是除了寫作程式外，也是我們要由此次專題中所學習的一部份。專題是一個 team work 而非是單兵戰鬥，因此良好的溝通將會讓彼此了解所欠缺和所表達的。在專題討論的過程中大家難免會有意見不合的時候，此時如何好好的和組員跟通便顯得很重要。猶記專題剛開始時，老師便一直耳提面命溝通在專題中佔極重要的一部份。在此很感謝我的組員彼此間的相互溝通和包容，每當我太急或彼此間意見不同時，我們總是能夠和彼此的聆聽和採納對方所給的建議，這是能讓我們做完這個專題的重要因素。在此奉勸學弟妹，一個專題的 team 當中一定要有一個 leader，這樣做起事來才不會都茫茫然無目標。此外 leader 也要和組員間有良好的互動，要紓解每個組員中所不滿的

事，並且淨利維持 team 的良好氣氛，這樣專題才能收到事半功倍的效果，也才能體會團隊合作的重要性。

在專業技能的方面，算是有極大的收穫。不論是在windows register、windows programming、密碼學、視窗程式設計等等..每一個領域都是我在大學時期從未接觸過的。很高興我能進入這幾座寶山，而且能夠滿載而歸，從中獲取真正屬於我的知識和體驗。這個部分是我開始做專題時無法預知的。這個專題的歷時不短，中間曾遇到許多瓶頸而停滯不前也曾萌生退意想要逃避轉換題目，進而到重新整理心情克服難關，到專題有成果出來，一切的感覺真是如人飲水冷暖自知。最後要感謝李維斌老師與助教的提攜與教導，還有跟自己一起撐到最後的組員們。

5.2 邱紹華的心得分享

這次的專題題目，選擇借由 windows 介面的滑鼠右鍵功能來加快對檔案加密的速度與方便，就如同 winzip 這套軟体一樣，不過我們的程式功能是在於加密而不是壓縮。老師起先給我們的出發方向是在密碼學這個方面，這幾年以來，加解密的 API 已經有許多人實作出來了，所以實作加密的演算法已經沒有前幾年來的那麼重要了，反而是如何讓使用者來完程加密的過程比較重要，所以我們這次實作的方向是偏重於可以讓使用者利用比較方便的方法來用加密的程式。

於是我們決定先實作出可以讓使用者在 windows 的視窗中直接用滑鼠右鍵來選取檔案來加解密，起初可以說完全沒有方向，因為這個會修改到 windows 內部的設定，但是也完全不知道要修改些什麼，後來透過上網找尋資料及參考一些相關的應用軟体找到我們要使用的功能，這個部份花了我們大部份的時間。

後來開始要做加密的部份，單一檔案來說，其實只要套用加密的 API 就算是解決了，但是對於一個存在多個檔案的資料夾而言，一般的加密 API 並沒有支援多資料夾加密的功能，所以我們後來決定要將一個要加密的資料夾中的所有檔案及內部的資料夾結構記錄起來，並且合併於一個檔案中，如此一來我們只要對這個合併的檔案來加密就解決這個問題了，不過在於合併檔案的部份我們也碰到了需多的問題，如記錄檔案的長度、結構、內容等等的，都是我們平時比較少會去實作的，所以我們也花了不少的時間在研究這個部份。

接著我們還考慮到使用者的加密檔案如果需要攜帶到一台沒有加解密程式的電腦上使用的話，可能會碰到無法解密的問題，所以我們又朝著可以加密成一個可以自行解密的執行檔，如此一來，這個加密過後的執行檔就可以在沒有加密程式所在的電腦上解密了，在這方面我們發現一般的*.exe 檔的檔尾後面可以放入其它不相關的檔案內容，而不會影響到即執行檔的功能，所以我們就把加密後的資料放在執行檔的檔尾後面來解決這個問題。

大致上專題完成的過程是這個樣子，不過這次的專題讓我學會了很多的東西，專業的知識上面的問題都還算是好解決，但是在於一個 team 中，要如何去溝通及分工來順利的完成一個工作，並沒有想像中的那麼簡單，剛開始做的時候，總覺得大家的方向及工作應該都已經大致了解了，所以就比較少有溝通交換彼此的意見，以致於各自去找自己需要問題解答，結果變得一段時間後，聚在一起交換了一下意見，發現原來大家的想法還是有一段蠻大的差距，那麼一來一定會有少許的爭執，使得彼此先前所付出的努力都會有部份的浪費，這樣並不能使一個 team work 達到最大的效果。

合作完成一個工作，一定要有充份的溝充，而且很重要的是 team 中的每一個成員都必需要完完全全的把自己的意見及疑惑表達出來給別人了解，不要一知半解然後就拚血埋頭苦做，這樣會很容易讓接下來的工作走錯方向，使得一個 team 的工作效能下降，嚴重的話也是造成一個 team 的合作破裂。

這個專題目前能夠順利的完成，要非常感謝我們的指導老師李維斌老師與助教的協助與教導，以及一路與我一起完成這個專題的兩位伙伴，也許起初我們還是會遇到老師不斷提起分工及溝通方面的問題，但最後我們還是克服了，也學會了很多專業的知識，所以這次真的是收穫良多。

5.3 蘇淑瑛的心得分享

由於資訊安全這個未知的領域引發了我的好奇心，因此，我想藉由此次的專題研究看是否能作為進入資訊安全這個領域的一個小基礎，不過這個想法太天真了，只知道密碼學中的幾個加密演算法，雖然知道他們的特性也作過了比較，但僅是如此似乎連這個領域的門檻都跨不過。

因此，未來我仍想繼續朝著進入資訊安全這個領域努力。回顧和組員一起努力的這段日子，因為自己實作能力比較差，所以主要致力於資料的搜尋與整合。經由和老師討論專題的可行方向後，對於資料保護方面，做了一般常見的加密演算法之特性、優缺點等比較，這一部份的資料，主要是從書上及網路上統整後所作的報告，希望能提供給閱讀這份報告的人們，對於密碼學有一點點基本的認知，或者能作為研讀的參考。

由於之前並未接觸過密碼學這部份知識，所以剛開始找資料的時後真的毫無頭緒，後來經由老師的導引，我在網路上找到了一些和資料安全有關的免費的軟體，我下載了幾個覺得適合的軟體試用後，將他們的特色寫成了一份報告，然後再和組員們討論，才漸漸的知道怎麼去選擇適合專題的資料。我覺得這樣的一個流程，讓我學到了找資料的技巧，不會盲目的搜尋找不到重點。

不過，我仍然覺得有些許的缺憾，那就是關於資料偽裝和隱藏的技術，在這個專題中，我們並沒有多做研究與討論，所以非常希望未來能應用這樣的技術來增強我們專題的功能選擇性與安全性。

最後，在這樣一個長達兩個學期的專題研究中，我非常謝謝指導我的老師和包容我的兩位組員，我只能說，我能做到的我一定會盡力完成，但是，如果沒有他們的指導和包容，如果在我遇到困難時沒有他們的幫忙，我想今天我一定無法繼續待在這個團隊中，我想專題有一目的也是為了訓練我們如何互助合作維持專題運作吧！以前常聽老師說有一些學長姐會因為專題而反目成仇，當時我怎麼想都想不通，現在我終於了解了，當一個人處於困境而其他人卻不肯伸出援手時，可能就會發生這樣的事吧！

由此可見，團隊工作也是一門值得大家去學習的課程，不管現在的課業或者未來的工作，相信常常會有機會需要大家一起合作的時後，因此，我的體會是:及早認清團隊工作的意義，才能做出對自己或者其他人負責的行為。

第六章　參考資料

[1] William Stallings，密碼學與網路安全原理與實務第二版，曾志光 ， 巫坤品 譯 ，棋峰資訊出版

[2] Douglas R. Stinson，Cryptography：theory and practice，Boca Raton Chapman & Hall/CRC ， 2002 出版

[3] MSDN Library ：

(a) Security->Cryptography->Cryptographic API(CryptoAPI) ->Technica Atricles

(b) Security(General)->Technical Atricles->Windows Data Protection

(c) Security(General)->SDK Documentation ->Cryptography->Avout Cryptography

(d) Security(General)->SDK Documentation ->Cryptography->using cryptography Acquiring a Cryptographic Context and Generating Keys

(e) Security(General)->SDK Documentation ->Cryptography->using cryptography Encryption and Decryption

[4] 位元文化，精通視窗程式設計，文魁資訊股份有限公司

[5] 林俊杰，Visual C++ 6 視窗程式設計經典，碁峰資訊股份有限公司，2003

[6] http://www.codeproject.com，提供關於 MFC 視窗程式設計的 sample 及相關討論

[7] http://www.codeguru.com，提供關於 MFC 視窗程式設計的 sample 及相關討論

[8] 彭建翔，Windows Registry 徹底揭密，金禾資訊股份有限公司