國立交通大學

資訊工程學系

碩 士 論 文

利用 DiskOnKey 的金鑰建構企業安全的電子郵件

Construct Enterprise E-Mail Security with PKI Keys in DiskOnKey

研 究 生：林祐廷

指導教授：劉振漢　教授

中 華 民 國 九 十 三 年 七 月

利用 DiskOnKey 的金鑰建構企業安全的電子郵件

Construct Enterprise E-Mail Security with PKI Keys in DiskOnKey

研 究 生：林祐廷　　Student：Yo-Tin Lin

指導教授：劉振漢 博士　　Advisor：Jenn-Hann Liou

國 立 交 通 大 學

資 訊 工 程 學 系

碩 士 論 文

A Thesis

Submitted to

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

In Partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

July 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年七月

利用 DiskOnKey 的金鑰建構企業安全的電子郵件

研究生：林祐廷　　指導教授：劉振漢 博士

國立交通大學資訊工程研究所

摘　　要

公開金鑰基礎建設已廣泛使用於網路交易、身分認證等領域，透過私密金鑰電子憑證在網路上提供解密、簽章、身分鑑別與不可否認性，如同網路上的個人身分證，因此私密金鑰的保護，一直是大眾所關注的議題。DiskOnKey 是一種連接在 USB 介面的隨身硬碟，可以安全的存放小量的資料，本研究利用 DiskOnKey 裝置具有隱藏區的特性，將使用者私鑰加密儲存在隱藏區裡，使用者是看不到私密金鑰的，只有應用程式能夠讀取，使用者只需帶著 DiskOnKey 接上使用者電腦，即可匯入公鑰憑證和私鑰，Outlook Express 上的郵件就可以加密或簽章。本研究另外實作了 CA 自簽憑證用來簽發 DiskOnKey 使用者憑證，另外，利用 CA 憑證建立安全的加密連線，提供沒有 DiskOnKey 的使用者傳送簽發憑證要求，用這三個部分建立一個安全電子郵件的小型區域網路，如同一般的企業網路。

Construct Enterprise E-Mail Security with PKI Keys in DiskOnKey

Student：Yo-Tin Lin Advisor：Dr. Jenn-Hann Liou

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION ENGINEERING

NATIONAL CHIAO TUNG UNIVERSITY

Abstract

Public Key Infrastructure has been widely utilized in fields of e-commerce and identity authentication. private key certificate, which is similar to real personal identification on the network, provides decryption, signature, authentication and non-repudiation. Therefore, the protection of private key has been largely concerned. DiskOnKey is a removable USB flash disk. It can save a small amount of data. This thesis made use of the hidden area characteristic of DiskOnKey device to store the encrypted user's private key in the hidden area. Users could not see the private key. Only the application program could read and recognize it. Users merely need to carry the DiskOnKey and connect it to the computer, and public key certificate and private key can be imported. Thus, the e-mails in Outlook Express can be encrypted or signed. In addition, the thesis implemented CA selfsigned certificate for signing DiskOnKey user's certificate. CA selfsigned certificate could also construct secure connection to assist users who do no have DiskOnKey to send certificate sign request. Therefore, we can use the above methods to construct local area network of secure e-mail, which is resemble to general enterprise network.

目　　錄

圖　目　錄

表　目　錄

第一章

緒論

自從網際網路在 1992 年解除商業限制後，開始了其發展的高峰，而且隨著 Internet 在全球各地不斷普及，與資訊安全相關的事件層出不窮，對資訊安全的威脅不斷加深，在目前的網路環境中，每個人都無法真正看到對方，全憑個人識別碼(ID)來識別通訊的對象。自從 RSA 演算法提出之後，網路個人身分鑑別變的簡單許多,利用 PKI 架構下的電子憑證,達成身分的不可否認性與資料隱密性。

隨著電子簽章法的通過，凡經由使用者簽章過後的文件，已具有法律上不可否認的地位，因此私鑰的保密程度更受到重視，利用DiskOnKey USB隨身碟的C-ookie特性，儲存加密後的私密金鑰，提高了私密金鑰的隱密性。當連接上電腦後即可匯入公鑰憑證和私鑰，提供了電子郵件加密和簽章功能，匯入電腦的憑證，無法匯出PKCS #12 格式的憑證檔，只能匯出.cer檔，即私鑰無法再從電腦匯出，私鑰只能儲存在DiskOnKey裡，且私鑰經由使用者輸入的密碼加密，密碼長度不限、密碼資料也不侷限於數字，增加了暴力破解法的難度。儲存在 DiskOnKey的私鑰，其相對應的公鑰憑證可以由網路上的公信機構發行或由自定的CA簽發，加上一台本研究自行建構的CA，和安全的網路連線，即可建構出一個提供安全電子郵件的區域網路。

第二章

背景知識介紹

第一節　密碼學的分類

密碼系統主要分為對稱式金鑰密碼系統和非對稱式金鑰密碼系統，最原始也最早期的加密技術是把字元位移，早在古希臘羅馬時代就有這樣的例子，現在網路通訊為開放架構，因此採用公開的密碼演算法加上金鑰的概念，也就是資料的加密與解密需要鑰匙配合進行計算，為了確保演算法是安全、無法破解的，金鑰的保護就顯的相當重要。金鑰是一連串 0 與 1 的組合，若長度為 N 位元，就可能有 2 的 N 次方種不同的變化組合，若使用暴力法破解，就得嘗試各種不同 0 與 1 的組合，因此金鑰的長度(位元數)越長，理論上安全性越佳。

1.1 對稱式金鑰演算法

資料加密標準(Data Encryption Standard)演算法最初由IBM公司的研究人員 W.Tuchman 和 C.Meyer 兩人於1971年研發成功，1977年由美國國家標準局(Nati-onal Institute of Standards and Technology)頒布為國家標準，為對稱式密碼系統的代表。DES是一種對稱式金鑰系統(單金鑰)，所以收發雙方都必須擁有同一把秘密通訊金鑰，其加密及解密演算法是對稱的，在發表初期，對資料的安全保密性、資料完整性(不被竄改)具有一定程度的保密效果。在加密的過程中，塊狀密碼器先將明文切成許多同樣大小(64位元)的區塊，金鑰的長度則是56位元，表面上金

鑰長度為64位元，其中只有56位元會用來加密和解密，其餘的8位元則是作為同位檢查位元(Parity Bit)之用，DES密碼系統架構可分為「初始排列運算」、「金鑰產生」「重複運算」及「反初始排列運算」四個主要部分。以下圖為例:

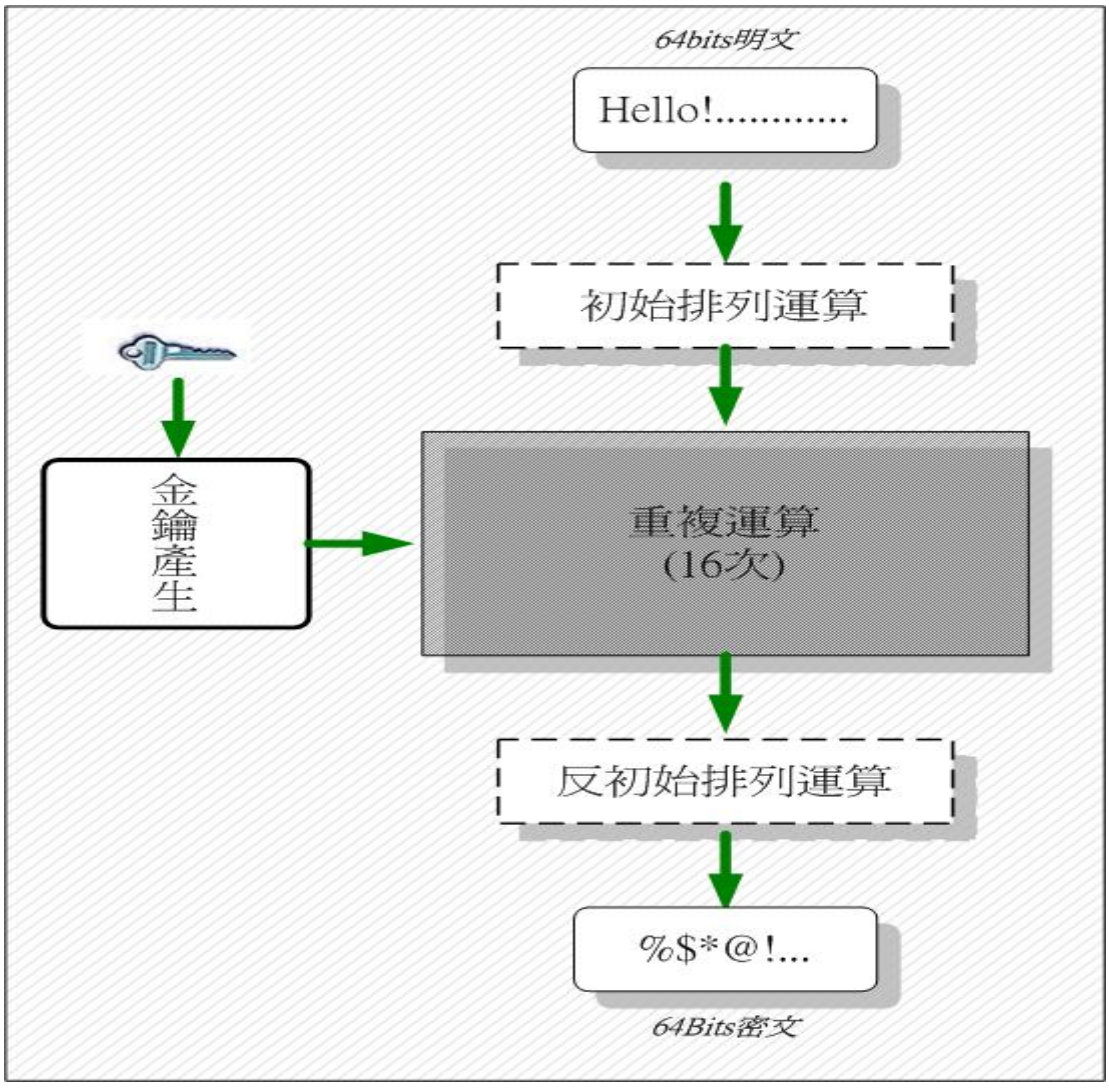

圖2-1.DES密碼系統架構

(1)初始排列運算:

將分割的64bits明文區塊進行初始排列運算Initial_Permutation()，並將運算結果分成左方32bits(L_0)及右方32bits(R_0)。

(2)金鑰產生

將56bits金鑰進行一次permutation_0()，得新的56bits金鑰LK_0，在透過16個不同的排列運算permutation_1()、…、permutation_2()、permutation_16 ()對LK_0

進行運算，得到16個56bits之金鑰LK_1、LK_2、…、LK_{16}，在經由縮減排列運算compression_p()變成16個供重複運算的48bits金鑰K_1、…、K_2、K_{16}。

(3)重複運算

將「初始排列運算」的結果(32bitsL_0、32bitsR_0)與「金鑰產生」的結果K_1、…、K_2、K_{16}進行重複運算。重複運算規則:1. 原始右方32bits R_0變成下一次左方32bits L_1，即【$L_1=R_0$】;2. 原始左方32bits L_0加上R_0與第一個金鑰K_1進行運算的結果【$R_1=L_0+f(R_0,K_1)$】;以此類推直到完成16次重複運算。

(4)反初始排列運算

將重複運算的結果L_{16}與R_{16}左右對調，並對該64bits進行反初始運算Inverse_permutation()，運算結果即得到最後64bits密文。

對稱金鑰系統運算速度快且單純，但由於其執行加解密時使用相同的秘密金鑰，因此在傳送訊息給接收者時，也必須將秘密金鑰傳給對方，在資料傳輸過程中，如何將秘密金鑰安全地傳給對方，始終是DES的一大問題。本法不只用於加解密，也可用在訊息鑑別，以保護訊息之真確性。DES演算法具有雪崩效應的特性，所謂的雪崩效應(The Avalanche Effect)就是只要明文裡經一點點的變更，或者在金鑰上做變化時，則最後的祕文就會產生重大的改變，所有的加密演算法都具有不錯的雪崩效應，一般來說若是雪崩效應不明顯，那麼演算法被破解的可能性就大為增加。

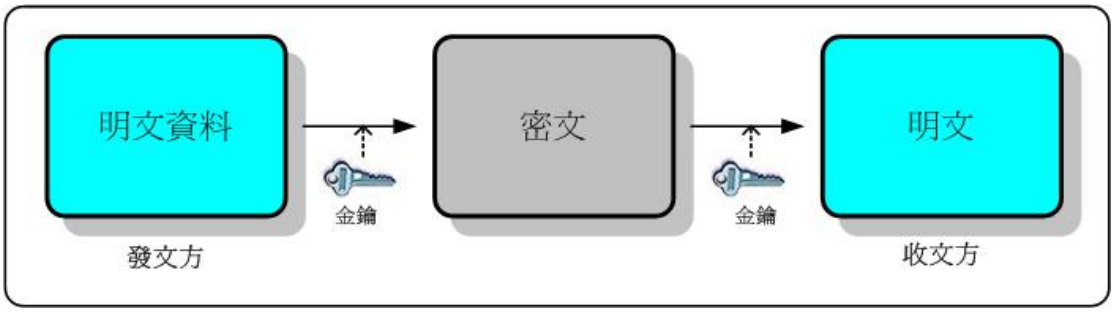

圖 2-2. 對稱的加解密方法

自 DES 演算法問世之後，從密碼分析的角度並沒有一個成功的數學解析解存在，亦沒有發現數學上證明可行的破譯攻擊，但隨著電腦計算技術的不斷發

展，DES 演算法所面臨的密碼分析衝擊日益嚴峻。

對稱性演算法的例子如：IDEA、RC2、RC4、DES、DES3、河豚演算法與鑽石演算法。本篇論文接下來會採用 RC4 作為預設的對稱式資料加密演算法，RC4 為一種串流加密器（stream cipher），由 RSA Data Security 公司的 Ron Rivest 所發展，金鑰長度 128 位元，它將短的鍵值展開成為無限制的虛擬亂數鍵值串。發送者使用這個鍵值串與明文資訊做 XOR 運算處理並產生密文。XOR「exclusive or」（互斥或）邏輯運算指的是是一種二進位系統的邏輯運算子，若兩個運算元不同，則結果為１，當兩個運算元相同時，其結果為０。依照這項原則，接收者則使用鍵值產生適當的鍵值串。並且對密文做 XOR 及鍵值串演運之後，就可以得到原始的明文了。目前使用 RC4 技術的公司有 Microsoft、Netscape、Oracle 等。由於這項技術嚴密，IEEE 802.11 委員會亦將 RC4 列入無線區域網路 WEP (Wired Equivalent Privacy)標準中。

1.2 非對稱式金鑰演算法

為解決交換金鑰的問題，便有公開金鑰密碼系統之產生。公開金鑰密碼系統改善了對稱式金鑰密碼系統的缺點，其加密金鑰與解密金鑰不是同一把，每一金鑰對包含兩把相互對應的金鑰，一把是可以公開之金鑰，另一把是不可公開之私鑰。

1976 年Diffie和Hellman首先提出了非對稱式金鑰的觀念，1978 年，三位麻省理工學院教授 Rivest、Shamir和Adleman三人共同提出RSA非對稱式金鑰演算法，利用不同的兩把金鑰來解決對稱式金鑰會遇到的問題。非對稱式金鑰之觀念如下：

- 每人擁有經運算產生之兩把金鑰稱金鑰對，一為公鑰(Public Key)，一為私鑰(Private Key)。公鑰將公佈給所有人知道。
- 私鑰將由持有人自己持有，不能公開。公鑰加密的資料只有該金鑰對之私

鑰能解密，私鑰加密的資料只有該金鑰對之公鑰能解密。

- 以送方私鑰加密資料，收方利用送方之公鑰解密，可達到確認該加密資料一定為送方所送且可散發給任何人。此種方法使用在電子簽章。
- 送方以收方之公鑰加密資料，收方以自己的私鑰解密，可達到該加密資料一定只有收方才可解讀，而任何人均可送給收方。此種方法使用在文件加密。
- 因公鑰為公開，故理論上可利用公鑰反運算得到私鑰。但此反運算將耗時過久而變得不切實際，因此得以保證私鑰之無法取得。但因此也使得正常之加解密運算時間也較長。
- 一般以RSA 運算來加解密。

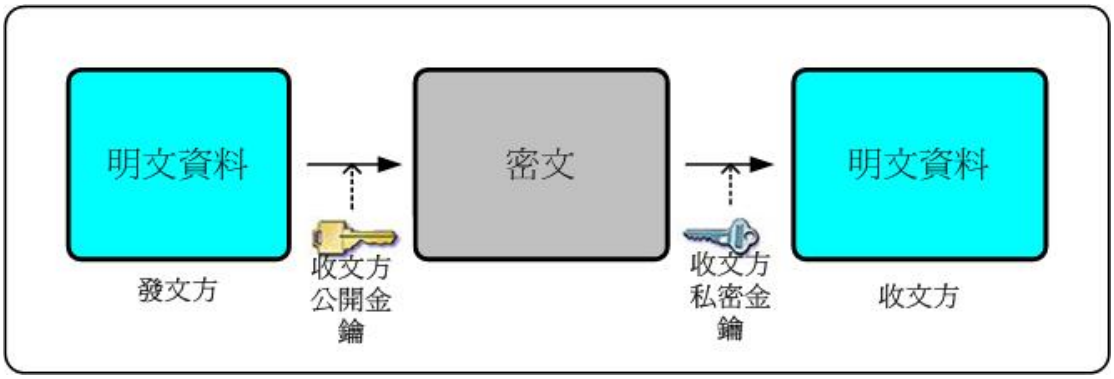

圖 2-3. 非對稱的加密方法

公鑰的加密系統是目前唯一能讓位於兩地的雙方安全分享資訊的方式，使用者不必擔心傳送的密碼會被竊聽者用來攔截雙方的對談內容。因此，公鑰加密系統已成為電子商務和虛擬私有網路〈VPN〉的運作基礎。非對稱式金鑰演算法以 RSA 公開金鑰演算法為代表，以下為 RSA 演算法之步驟:

1. 收報者取兩個相異的大質數 p、q 和另一個與(p-1)(q-1)互質的數 E，且 E<w 令 $w=(p-1)(q-1)$, $E<w$, $n=pq$。

2.（公開）告訴發報者 E 與 n。

3. 發報者將他的信號分成許多段，每段含 k-1 位數（十進位）（若 k=3（即 p、q 均為不小於二位的數），則信號 331414320001 應分段成 33，14，14，32，00，01 ，一個一個的發出，設發報者的信號之一為 x（k-1 位數，即上面之 33，或 14，或 32，…），則他將它作成 $c = x^E \pmod n$ 發出。

4 收報者收到 c 之後，即可把原有的 x 求出來，因 E 與 w 互質，我們可找到二整數 d、e；d>0 使得 ad+we=1，令 $y \equiv c^d \pmod n$ ，則此 y 即發報者之 x。

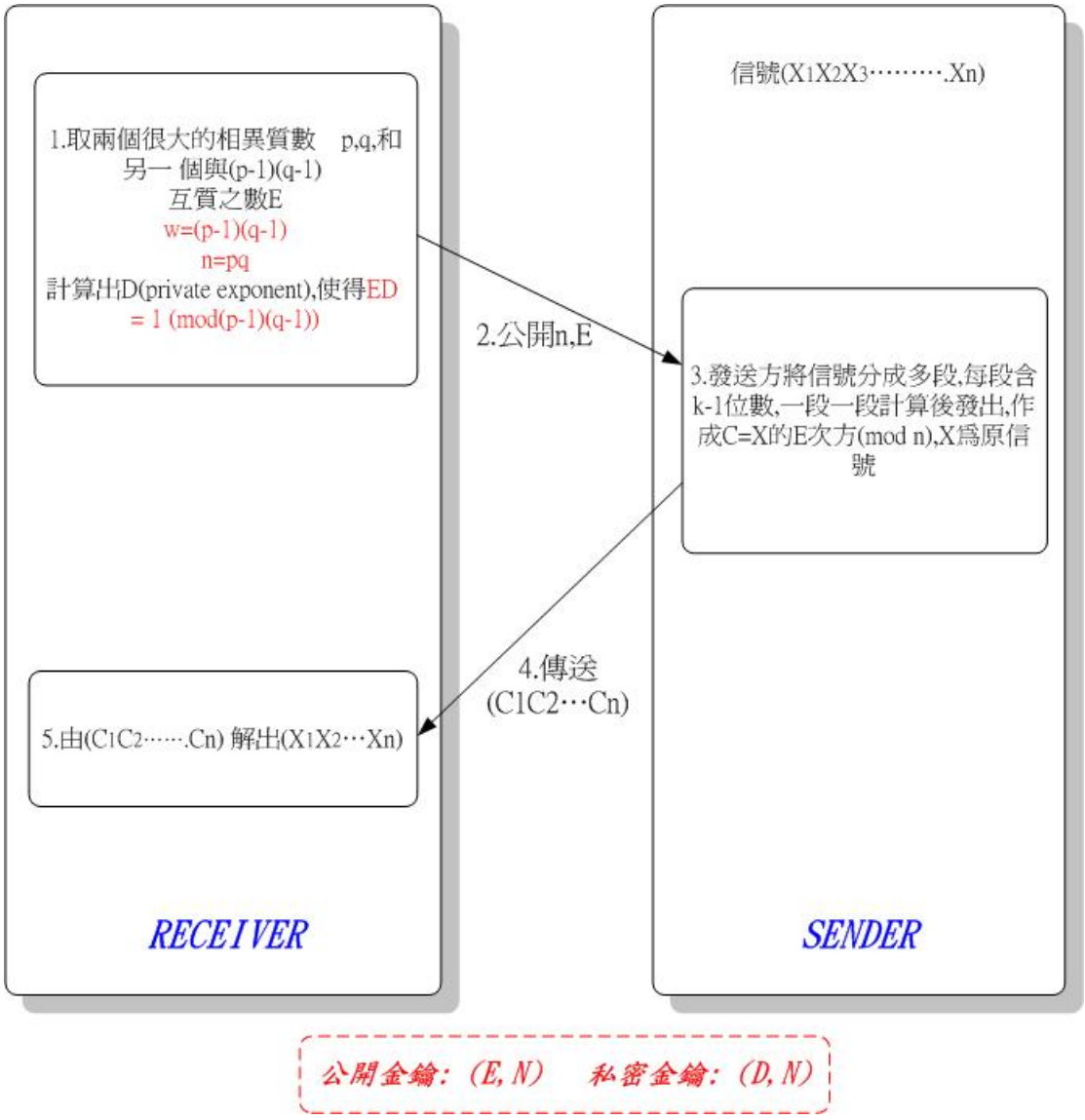

圖 2-4. RSA 演算法流程

這種密碼之關鍵在於 p、q 為兩大質數時，分解 m 成為 p、q 為一件極費時工作，若分解不開 m，則找不到 w 與 d，因此就無法從 c 解得 x，在不久以前，要分解

一個數的因子仍停留在近乎硬試的階段，即要從 2，3，5，7，…，一直試到根號 n 附近才停止。若 n 是 50 位數而 p、q 均近 25 位數，則分解 m 要除約 10^{25} 次，若以電子計算機以每秒 10^6 次的高速運算，這仍是一個 10^{11} 年的工作，目前由於大家對這方面的重視，分解一個 50 位數的時間已可縮短至 10^{10} 次運算。

1.3 對稱式與非對稱式金鑰演算之比較

密碼系統分為對稱式(Symmetric)密碼系統及非對稱式(Asymmetric)密碼系統，其特性、優缺點如下:

1. 對稱式密碼系統

(1) 特性: 在網路上建立安全通道是一大問題，且每一把密鑰必須由共用該把密鑰的通訊雙方同時秘密保存且不可外洩，因此當一方可能與多方通訊時，必須與各方先協商產生各自的私鑰並安全送達，故金鑰不易管理。

(2) 優點:於處理大量資料之加解密具有良好之效率。

(3) 缺點:
- 金鑰必須被分享，安全性較差。
- 金鑰分配問題。
- 金鑰數目眾多，產生金鑰管理問題。
- 無法達到不可否認性。

2. 非對稱式密碼系統

(1) 特性:
- 公鑰與私鑰雖然具有數學上對應關係，但其產生方法是不可逆，即無法由公鑰推得其相對應之私鑰，因其為複雜之數學關係。若利用暴力破解法，所需的時間太久。
- 支援數位簽章。
- 通常需要建置PKI架構。

(2) 優點:
- RSA之安全性取決於質因數分解之複雜度，要將很大的正

整數N，因數分解為P與Q之相乘，是很困難的，故可以達到機密性。

- 簡化金鑰分配及管理問題。
- 使用數位簽章可達到不可否認性。

(3) 缺點: • 加解密運算複雜且速度較慢。

對稱式與非對稱式密碼系統各有其特性及優缺點，故目前一般均採用各自的優點混合使用，例如資料加密即利用對稱式金鑰速度較快的優點將明文加密，再配合非對稱式金鑰的加密機制來傳送對稱式金鑰。

第二節 公開金鑰基礎建設(Public Key Infrastructure)

2.1 公開金鑰基礎建設(PKI)與電子憑證

公開金鑰基礎建設(Public Key Infrastructure，PKI)是以公開金鑰密碼學技術為基礎而衍生的架構，公開金鑰密碼技術為非對稱式演算法，以 RSA 演算法為基礎，提供訊息的身份鑑別(Authenticity)、資料完整(Integrity)、不可否認性(Non-repudiation)與機密性(Confidentiality)等資訊安全四大需求功能。資料加密與資料解密使用不同的金鑰，以接收方的公開金鑰(Public Key)對傳送訊息欲隱密的部分，執行 RSA 演算法，將訊息加密，接收方收到訊息後，以自己的私密金鑰(Private Key)對加密訊息執行 RSA 演算法而解密。

然而，公開金鑰密碼系統有一個缺點，就是公鑰在網路傳輸過程中，可能會被非法第三者給竄改或是破壞，為了解決這個問題於是產生了以憑證為基礎的公開金鑰密碼系統(Certificate-base Public Key Cryptosystem)，也就是藉由公正的第三者(CA)發給合法的數位憑證(Digital Certificate)，在公開金鑰基礎建設模型中，有以下四個主要元件:

1. 憑證使用者(Certificate User):使用者自己產生一組金鑰對之後，分別交給註冊中心與憑證中心以取得一份憑證，或者在需要與另外一位使用者溝通時，向憑證儲藏中心要求查詢這一位使用者的憑證。
2. 註冊中心(Registration Authority，RA):使用者將金鑰對交給註冊中心驗證之後，註冊中心會產生一份經過認可但尚未簽署的憑證，交由憑證中心來完成簽署。
3. 憑證中心(Certification Authority，CA):憑證中心在憑證上簽章，完成簽署手續，並且存入憑證儲藏中心，讓憑證可以經由查詢而得到，如果憑證出現問題，或者依使用者要求，憑證中心可以依照一定的程序來廢止憑證。

4. 憑證儲藏中心(Certification Repository):這裡儲存憑證中心簽署過的憑證，和一份憑證廢止清單(Certificate Revocation List，CRL)，上面列出了所有遭到憑證廢止中心公告廢止的憑證，讓使用者可以隨時查詢得到最新的結果。

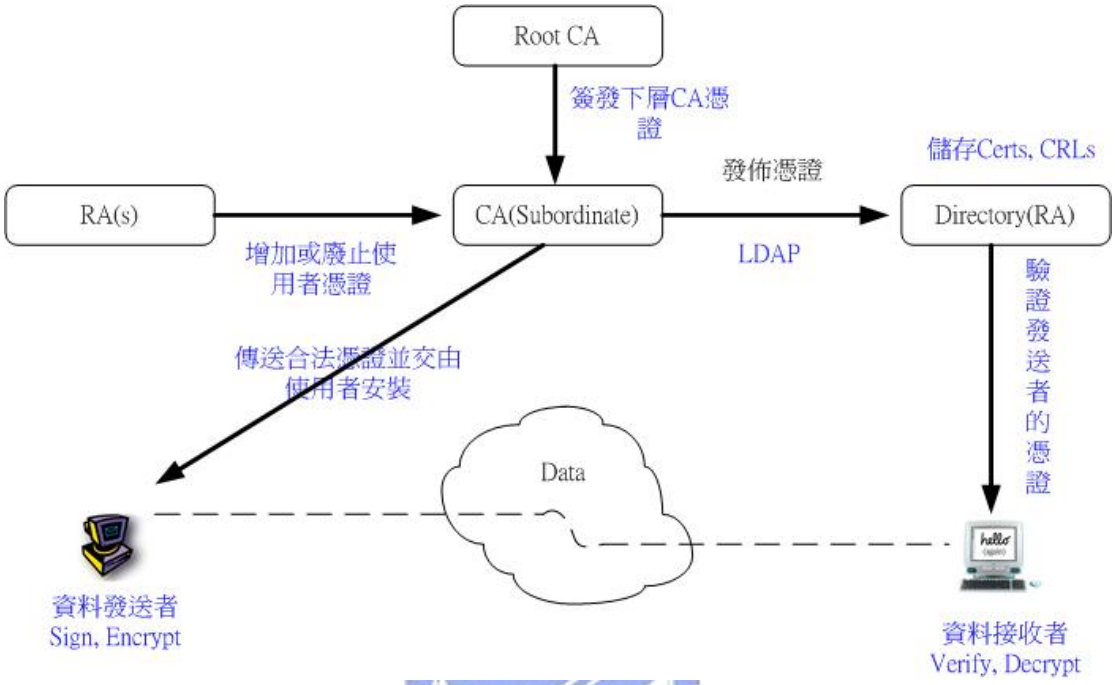

圖 2-5. PKI 的架構

在公開金鑰基礎建設中扮演最重要的角色就是憑證，憑證中最基本的資料包含公開金鑰、使用者(Subject)的身分、憑證序號、憑證有效期限、簽發憑證的憑證中心、及憑證中心所簽署的簽章，目前憑證最通用的標準是X.509 v3(ITU-T X.509 version 3)，此標準提供最好的通用性及延展性。

2.2 X.509 憑證服務

X.509是國際電信組織(ITU-T)所制定的X.500目錄服務(directory serv -ice)系列文件的其中之一，X.509 公開金鑰憑證規格歷經三次沿革，目前的版本為版本三，三個版本間的差異性如下。

- X.509 版本一：1988 年提出，為X.500目錄服務中的一部份，定義出標準的憑證格式。

- X.509 版本二：1993 年提出，在憑證中增加了兩個欄位，可以用來支援作目錄存取控制。
- X.509 版本三：1996 年6 月提出。此版本在版本二中加入了預備的額外擴充(extension)欄位，特殊的擴充欄位型態可能會被標準化或是被任何公司組織所定義和註冊。ITU-T 定義了X.509 版本三的擴充欄位型態如：主體識別(subject identification)、金鑰屬性(Key attribute)、策略映射(policy mapping)與憑證路徑限制(certificate path constraints)等。

在TBS (To Be Sign)憑證中發行者與主體唯一識別碼只有在X.509 版本二後才支援，故若有此兩欄位，則憑證中版本欄位必是版本二或三，而擴充欄位則是在版本三後才支援。

圖 2-6 X.509 三版本公開金鑰電子憑證比較

上圖說明了X.509v3憑證的組成欄位，以下說明各欄位的意義:

- Version: X.509憑證版本，其中第三版較前兩個版本多了一個擴充欄位(Exten

-sions)。

- Certificate serial number: 憑證中心所發出的憑證序號，此序號在該憑證中心是唯一的。
- Signature Algorithm: 用來簽署這份憑證的簽章演算法代號。
- Issuer name: 憑證中心的身分，以X.500來表示。
- Period of validity: 包含兩個日期，利用”Not before”和”Not after”兩個欄位來區隔憑證的有效期限。
- Subject name: 擁有這份憑證的使用者的身分。
- Subject’s public key info: 憑證擁有者的公開金鑰資訊，包含公開金鑰，演算法代號等。
- Issuer unique name: 憑證中心的唯一識別身分。
- Subject unique name: 憑證擁有者唯一識別身分。
- Extension: 其他擴充欄位。
- Signature: 整份憑證的簽章是由憑證中心來簽署的。

CA 發行憑證時採用公開金鑰演算法，利用CA 的私密金鑰依照憑證中指定的簽章演算法對憑證的TBS 作簽章，故驗證憑證也是依照相同的程序，將簽章值用憑證CA的公開金鑰解密便可以得到TBS憑證的雜湊值，此時再對自己所收到憑證的TBS憑證欄位作雜湊，比較兩者所得的值是否相同，若相同代表憑證沒有被竄改過。使用者在拿到由憑證中心所簽署的憑證後，任何使用者都能以憑證中心的公開金鑰來驗證憑證裡面的使用者公開金鑰，且任何第三者對這份憑證的修改都能夠被偵測出來，憑證無法被任意的變造，故可以放在公開的伺服器上供人查詢，由於使用者的數量非常龐大，因此X.509憑證中心採用階層式架構，來建立彼此間的信任關係。

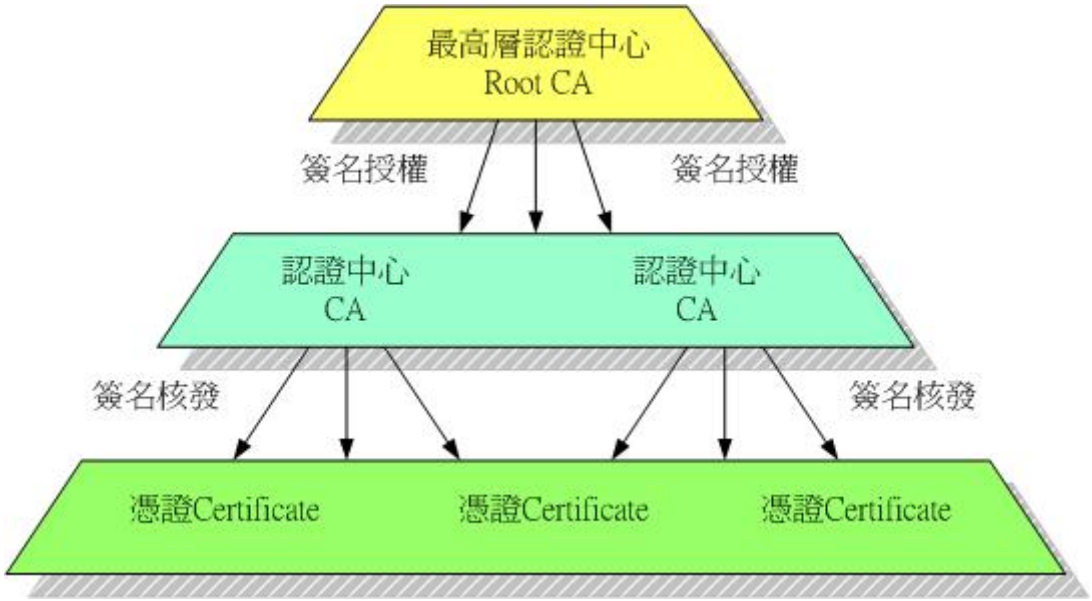

圖 2-7 x.509 的金字塔制度

在 X.509 中，每一個合格的憑證上，都會有一個簽名。最下層的憑證上，會有一個認證中心（CA）的簽名，表示這個認證中心（CA）檢查過，確認所有者資料無誤。中間的認證中心（CA）上，也會有管轄它的最高層認證中心（Root CA）的簽名，表示最高層認證中心授權給它，可以簽發別人的憑證。只有最高層認證中心上，因為它已經是最大，沒有再上層可以給它簽名了，所以只好自己簽自己，憑證上的簽名是自己簽的。程式自己會認得幾家可靠的認證中心(CA)碰到不認得伺服器的憑證（Certificate)時 ，只要那個憑證上，有自己認得的認證中心（CA）簽名保證過，那個憑證就沒有問題。

第三節　雜湊函數(Hash)

雜湊函數(Hash)加密法，又稱為「訊息摘要」(message digest)，其主要是將任何長度的資料透過函數轉換為固定的訊息長度。此類加密法與其他加密法最大的不同在於其轉換過程為不可逆，亦即無法從轉換出來得資料訊息還原成原來的內容。

由於不同的資料透過雜湊函數轉換之後所產生的訊息都不同，因此可以將訊息當作是資料的「訊息指紋」(message fingerprint)。當再網路上傳送資料時，為避免資料在傳送過程中遭到竄改，可以事先將資料透過雜湊函數轉換出訊息摘要，在傳送資料時一併傳送給接收者，當接收者收到資料後可把資料透過相同的雜湊函數轉換出訊息摘要並與傳送者所傳送的訊息做比對，若數值相同代表資料是正確的。

單向雜湊函數(One Way Hash)亦即一般的雜湊函數，具有以下特點:

(1) 可把任意長度的資料映射到固定長度的訊息

(2) 其過程為不可逆，因此無法從固定長度的訊息摘要還原成原來的資料內容

(3) 產生碰撞的機率微乎其微

(4) 所產生的訊息可以用來證明資料的正確性

(5) 無法防止擁有相同雜湊函數的第三者竄改資料

雖然接收者可以利用轉換出來的訊息摘要來證明資料的正確性，但假使第三者擁有相同的雜湊函數，在竊取傳送者的資料與訊息摘要後，若將資料內容加以更改並利用相同得雜湊函數轉換出對應的訊息摘要一起送給接收者，則收件者即使利用訊息所產生出來的訊息摘要進行比對資料，也無法確定資料的真確性。

3.1 MD5 演算法

MD5 此訊息摘要(Message Digest)演算法是由提出RSA演算法中的MIT教授Ron Rivest 所發展出來，其輸入任何長度的訊息產生輸出一128 位元的"finger

-print" 或" message digest" ，輸入訊息會被分成好幾個512 位元的區塊來處理。雖然較長的訊息摘要長度可以減少雜湊值碰撞的機率，但相對的運算時間會較長。其演算法描述如下：

假設有一b 位元的訊息輸入要找出其訊息摘要，b 位元是一任意非負值的整數也可以為0 值，並不需要是為8 位元的倍數且可為任意大值，此位元值可視為：m_0 m_1…..m_{b-1}，下面五個步驟說明如何計算出信息的訊息摘要。

(1)加上填充(Padding)位元

在訊息之後填充一些位元使得它的位元長度在求得512 位元的餘數後會等於448 位元（即訊息長度 mod 512≡448），換句話說，填充過後的訊息長度會比512 位元的倍數減少64 位元，因此一定要在訊息尾端後填充64入位元使位元長度符合512 位元的區塊以供處理，填充方式為先填入一個1 位元後再用0 位元填補到所需的位元長度。

(2)加上長度

將步驟一原始的訊息長度表示成一段64 位元的資料，並填充在先前產生的訊息末端，在最不可能的狀態下原始長度若大於2^{64}，則只有低序的64位元被取用，這些位元被以兩組32 位元的Word 來加入且低序的Word 先配給先前的訊息末端，所以此時欄位的內容即為原始訊息長度取2^{64}的餘數。經由前兩個步驟之後訊息長度變成為512 位元的倍數，相對地，這訊息長度也是16 個32 位元Word 的倍數，假設M[0….N-1]是訊息結束中的Word，則N 會是16 的倍數。

(3)初始化MD 緩衝器

使用一組4 個Word 的緩衝器(A、B、C、D)來計算訊息摘要，A、B、C、D 每一個皆是一32 位元的暫存器，使用下列16 位元的數值來初始化這些暫存器一樣是低序位元組優先。

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

(4) 處理在16 個Word 區段中的訊息

首先定義四組輔助函式，每一組皆為輸入3 組32 位元的Word，而產生一輸出為32 位元的Word。

F(X,Y,Z) = XY v not(X) Z

G(X,Y,Z) = XZ v Y not(Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X v not(Z))

F 函式中每一個位元的執行條件為：if X then Y else Z。F 函式中的每一個位元都是獨立且公平的，函式G、H、I 是類似於F 函式，只不過其基本邏輯函數不同。四個函數共需執行四個回合，每一個回合分別處理512 位元的區塊和4個Wo-rd 的暫存器ABCD；而這四個回合會分別更動這些暫存器的內容，利用sin 函數來建構64 個元素表T[1…..64]。用T[i]來表示表中的第i 個元素，其值等於2 32 (abs(sin(i))的整數部份，這裡的i 表示為弧度(radians)。

(5) 輸出

訊息摘要產生的輸出是A、B、C、D，開頭用A 的低序位元組，末端用D 的高序位元組，成為128 位元的訊息摘要。

3.2 SHA-1演算法

安全雜湊演算法(Secure Hash Algorithm)，是由美國國家安全局(NSA)所制定，並由美國國家標準局(NIST)於1993年公布，SHA-1則是SHA在1994年的修定版，修正一些SHA未經公佈的缺陷，SHA-1之設計理念與MD系列相似，但可以處理較長(160位元)的雜湊值，亦可以處理小於2^{64}位元的文件，處理速度比MD5稍慢一些，但其雜湊值的長度較長，在地毯式搜尋及逆向攻擊(Inversion Attack)下具有較高的安全度。SHA-1運作方式描述如下:

(1)添加附加內容

對於遇執行雜湊運算的訊息，先添加一字串，除第一個位元為1外，其他皆為0，而此字串的長度加上原上原來訊息位元的長度除以512於48。在將原本的長度，以64位元表示法，附加在最後，如此可以確保所有要輸入的訊息長度為512個位元的整數倍。

(2) 開始計算雜湊值

SHA-1的運算程序與MD5相似，可以由下圖看出其做法，都是上一次輸出為下一次壓縮函數的輸入。

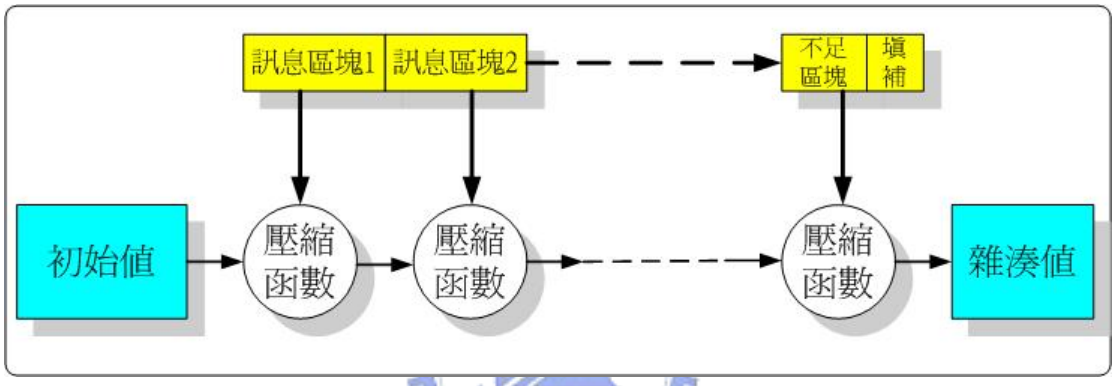

圖2-8 雜湊函數運算示意圖

將訊息M以512位元為單位，分成若干個訊息區塊，依序如下圖所述的方式輸入壓縮函數，執行雜湊運算，可得最後雜湊值

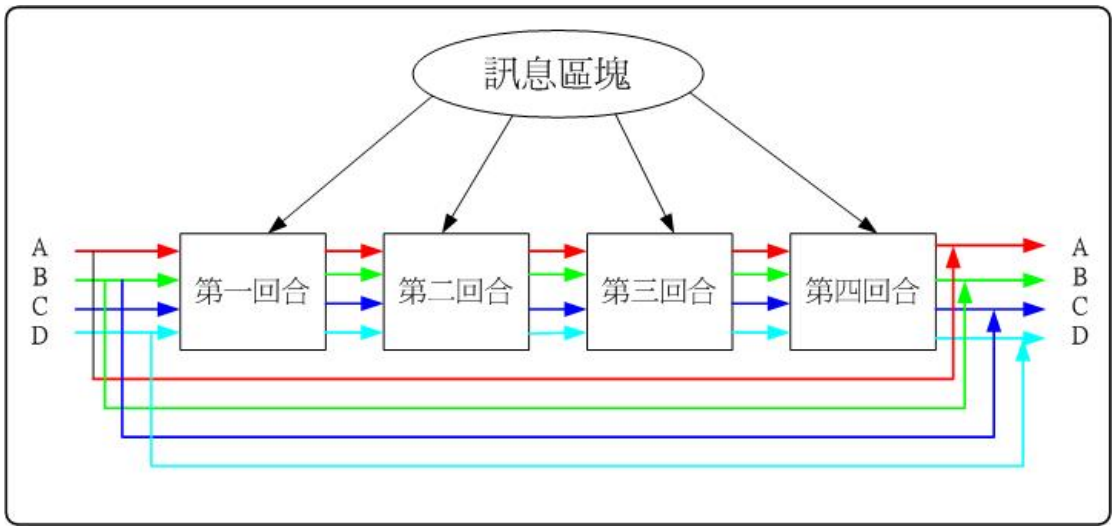

圖2-9 壓縮函數運算流程圖

將512位元的區塊分成16份32位元的子區塊。在此分別以M_0，M_1，...，M_{15}表示，

在每一個區塊有四個初始值

A=0x67452301

B=0xefcdab89

C=0x98bddcfe

D=0xc3d2e1f0

每一次執行SHA-1的運算，需執行80個步驟的運算，最後將abcd ABCD相加(此法為mod 2^{32} 的加法)，可得到整個訊息雜湊運算的結果。

第四節 數位簽章(Digital signature)

在現實生活中與他人進行交易與往來行為，可以依靠交談、出示身分證明等方式，提供他人辨識而產生信賴關係。然而就網路世界而言，雙方當事人可能素未謀面，如何能確認彼此真正身分來進行交易，實在是一大難題。由於網路上通訊及交易內容均以電子檔案進行傳輸，如何在資料傳送中不致遭第三人惡意竄改或擷取監看，以及如何避免交易完成後產生「事後否認」情形，均為推展電子商務時必須解決的重要課題。數位簽章（Digital Signatures）為電子簽章的一種，由於非對稱式公開金鑰密碼系統的編碼與解碼速度較慢，因此在製作數位簽章時，通常並不直接對整份訊息編碼，而是先利用「雜湊函數(Hash Function)」將訊息原文濃縮轉換為一固定長度的「訊息摘要(message digest)」，雜湊函數為單一方向性，即經雜湊函數運算後，無法反運算得到原訊息，然後再利用發文者的私密金鑰，對該訊息摘要加密，以做為該訊息原文的數位簽章，實際作業流程如下所示。

M：本文(Message)

S ：簽章(Signature)

C ：憑證(Certificate)

d ：發文方製作之訊息摘要(Digest)

d´：收文方製作之參考訊息摘要

SKs ：發文方私鑰

PKs ：發文方公鑰

H( ) ：雜湊函數運算。

ESKs（ ）：以發文方私鑰加密。

DPKs（ ）：以發文方公鑰解密。

1. 將欲簽章之本文(即明文*M*)以雜湊函數*H( )*運算，得到一固定長度之資料 *d=H(M)*，稱為訊息摘要 。

2. 使用RSA演算法將訊息摘要以發文方之私鑰加密形成此發文方之數位簽章 S = $ESK_S(H(M))$。

3. 將本文加上數位簽章及發送方之公開金鑰憑證(M & S & C)傳送至接收方。

4. 接收方先以發送方憑證欄位中之憑證中心簽章來驗證發送方之憑證是否有效並確認發文者之身分。

5. 如確認憑證無誤，則以發送方憑證內的公鑰將簽章 S 解密，得到發文方之訊息摘要$d = DPK_S(ESK_S(H(m)))$

6. 收文方將收到之本文 M 內容以雜湊函數(Hash)運算，得到參考訊息摘要 $d' = H(m)$ 。

7. 再將此訊息摘要d'與發文方之訊息摘要 d 比對，若 d´=d ，則表示來文於傳輸過程中未被修改，收文方以此確認發文方之身分未被造假；反之d´ ≠ d ，則代表來文已被篡改，。

相同文件由不同發文者簽署時，簽章不同，相同發文者簽署不同文件時，簽章也不同，簽章檢驗容易，可由任何第三者來執行。驗證簽章內容一致時可以確定:

1. 發文者身分（身分不可否認性）
2. 明文內容完整性（內容未被竄改）

但未加入保密機制，故其他人可以看見明文內容。

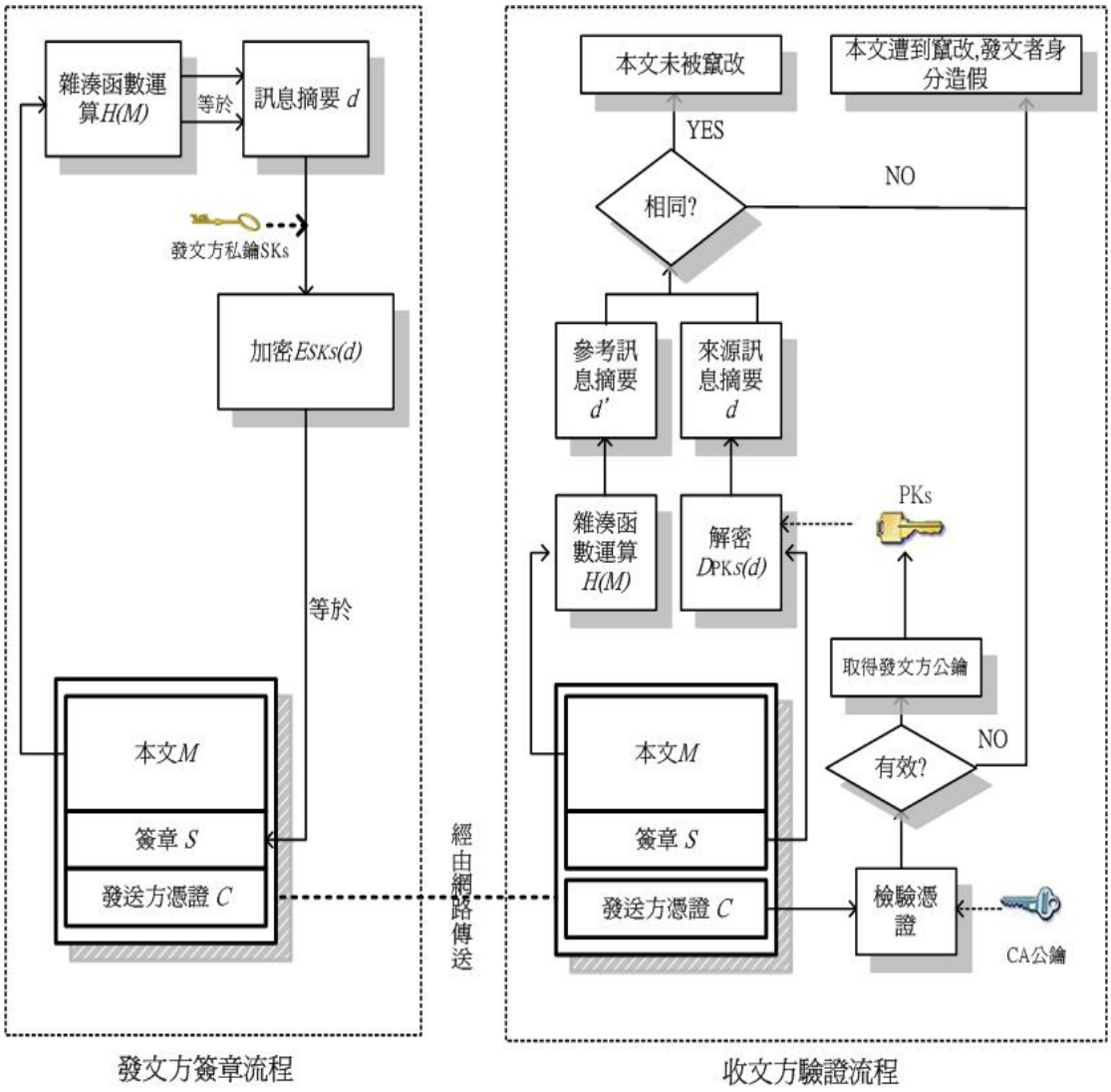

圖 2-10 數位簽章流程圖

第五節 Secure Socket Layer(SSL)

SSL 為Secure Socket Layer 的簡稱，最早在1994 年Netscape 所提出的標準草案，經多次的改進升級，目前最新版是1996 年11 月提出3.0 IETF(Internet Engineering Task Force)，其工作位於應用層的HTTP協定與會議層的TCP協定之間。SSL 已廣泛被業界所接受, 成為 Internet 的通訊安全國際標準，除 Netscape 的產品外, 包括微軟的IE 等很多其他產品均支持SSL 的通訊協議。因為SSL 的安全與方便性，目前網路上絕大多數有關電子商務的網站皆是利用SSL 的方式來加密交易資料。SSL通訊過程可以分為兩個部分:

1. Handshake

   SSL通訊是一種綜合使用對稱金鑰與非對稱金鑰加密技術的安全通訊協定，Handshake的過程可以分為以下幾個步驟:

   (1)提出安全連線請求

   由用戶端提出安全連線請求，在這一步驟中，用戶端將告訴伺服器，他可以支援多少位元的加密，有哪些演算法可以使用。

   (2)伺服器回應請求

   伺服器收到用戶端的請求之後，就檢視管理員設定SSL連線規則，如符合基本連線要求，則會選擇用戶端可以接受的演算法，回應用戶端並向用戶端提供自己的公開金鑰。

   (3)用戶端收到伺服端資料後，就確定了本次SSL加密所需使用的具體演算法，並且驗證伺服器金鑰的真實性，如果憑證沒有問題，用戶端程式會隨機產生一個金鑰(Session Key)，最後使用伺服器提供的通開金要加密後回傳給伺服器。

   (4)伺服器確定加密傳輸

   伺服器收到用戶端加密傳輸資料後，用自己的私密金鑰解密，獲得對稱式演算法的金鑰(Session Key)，盡而建立與用戶端的加密訊息通道。

2. Record

由於對稱加密金鑰由用戶端產生，使用伺服器的公開金要加密後在傳送到網路，所以只有擁有私密金鑰的伺服器才能解密獲得正確的對稱示金鑰，伺服器與用戶端就可以使用此對稱是金鑰加密需要傳輸的資料，以及解密對方傳送的資料。

圖 2-11 SSL 連線 Handshake 流程

第六節 DiskOnKey USB Flash Disk

DiskOnKey 為傳輸速度全世界最快的 USB 外接快閃式硬碟，由以色列的 M-Systems 公司所開發，DiskOnKey 平台以新型 DOK T4 晶片為動力，讀取速度 1000KBytes/sec，可攜式 USB 界面快閃記憶體產品是目前的大熱門，這項技術的實用性的確不容小覷。目前在同級產品中，最棒的似乎是 M-Systems 所推出的 DiskOnKey 。M-Systems DiskOnKey 採用了許多新技術（如 KeySafe 與 MyKey）來保護 DiskOnKey 中的資料。裡頭 32 位元的 ARM7 處理器能讓您直接 DiskOn Key 執行一些工具程式，透過加密提供所向披靡的資料保障。此外 DiskOnKey 還附上了一個資料安全軟體 KeySafe，這個軟體就存放在 DiskOnKey 裡。

利用 DiskOnKey 的 KeySafe 軟體，使用者可以設定 Safe Partition 密碼，Safe Partition 的容量大小可以由使用者利用 KeySafe 調整，即使整個 DiskOnKey 的容量都設為 Safe Partition 都可以，若進入 Safe Partition 的密碼輸入錯誤 3 次，Key Safe 會自動格式化 DiskOnKey。

DiskOnKey 還具有一項特點–Hidden Area，以 DiskOnKey version 3.x 來說，具有 15K 的 Hidden Area 可供 M-System 的協力開發廠商開發應用程式存取隱藏區，應用程式開發廠商可以產生片段的整數或字串並儲存在 Hidden Area 以供未來應用程式存取使用，舉例說，隱藏區可以被用來儲存「密碼提示」，使用者忘記密碼時，應用程式就可讀取隱藏區取得密碼提示。有一點要注意的是使用者不能直接存取得隱藏區的內容，必須要透過應用程式才可以存取隱藏區裡的值。本論文即是利用隱藏區此一特點，完成以下的實做。

第三章

系統架構與流程

第一節 視窗平台上安全(Security)相關的發展工具與環境

1.1 程式開發工具設定

在 Microsoft Windows 平台上開發應用程式，必須依照 Microsoft MSDN Libr-ary 所發表的 Security 相關知識，並使用微軟發布的函式庫 Platform SDK，Platfo-rm SDK 為一個免費下載的函式庫，程式原始碼並不公開，但提供各式各樣的 SDK(System Develop Kit) 讓有意在 Microsoft 作業系統上開發軟體的程式設計師使用，以下介紹程式開發前準備工作:

Ⅰ.準備軟體

Microsoft Visual C++ 6.0 以上版本

Microsoft Platform SDK 最新版本(下載網址:

http://www.microsoft.com/msdownload/platformsdk/sdkupdate/)

Ⅱ.設定 Library Path

Platform SDK 下載並安裝完成後，還需在 Visual C++裡設定 Library Path 和 Include Path，Platform SDK 會預設安裝於 C:\Program Files\Microsoft SDK。

1. 開啟 Visual C++ 6.0 主視窗上方 Tools->Options->Directories，在 Show Dir-ectories for:選項中選擇 include files 並將 C:\Program Files\Microsoft SDK \include 加入 Directories 裡，要注意的是使用者必須把 C:\Program Files \Microsoft SDK\include 欄位提升到最高的欄位，否則在編譯程式時，會因為 compiler 先讀取到舊的 include files 造成編譯錯誤。

2. 開啟 Visual C++ 6.0 主視窗上方 Tools->Options->Directories，在 Show Dir-ectories for:選項中選擇 Library files 並將 C:\Program Files\Microsoft SDK \Lib 加入 Directories 裡，要注意的是使用者必須把 C:\Program Files\ Micro-soft SDK\Lib 欄位提升到最高的欄位，否則在編譯程式時，會因為 comp-iler 先讀取到舊的 Library files 造成未知的編譯錯誤。

1.2 Windows 環境與 Security 發展工具

本論文程式主要使用 Platform SDK 裡的 CryptoAPI 來開發，CryptoAPI 提供程式發展公具，讓程式設計師可以透過 CryptoAPI 進行加密、解密、驗證數位憑證、將資料編碼成 ASN.1(Abstract Syntax Notation One)型式或解碼 ASN.1 的訊息，開發應用程式的程式設計師可以使用 CryptoAPI 裡的函式而不需要知道底層硬體的實際運作流程，CryptoAPI 的運作需要 Cryptographic Service Provider(CSP)的配合才能正確執行密碼函式。要在 windows 平台上開發應用程式，必須先清楚的了解微軟提供的軟體開發環境並具備 windows 平台相關知識，軟體開發者須遵循微軟發布的系統發展規則，應用程式才能在 windows 平台上正確的執行，以下介紹實作本篇論文程式所必備的知識:

(1). 密碼服務管理員(Cryptographic Service Providers)

CSP 是一個實際運作密碼學演算法相關的驗證、編碼、加密等運作的獨立軟體模組，並提供金鑰貯藏和安全相關功能，舉例來說，特定的應用程式可以讀取、寫入、或將特定的資料結構交給 CSPs ， CSPs 獨立的運作並將結果傳回給應用程式，windows 系統上所有的金鑰被設計成 CSP 的特定變數，這樣的設計可以讓應用程式執行在各式各樣的安全機制，當應用程式推出更新版本實，只需修改加密演算法的金鑰，而不需要修改整個應用程式的架構。微軟一共提供 10 組不同的 CSPs 供程式設計師使用。

定義名稱	所代表的意義
MS_DEF_PROV	"Microsoft Base Cryptographic Provider v1.0"
MS_ENHANCED_PROV	"Microsoft Enhanced Cryptographic Provider "
MS_STRONG_PROV	"Microsoft Strong Cryptographic Provider"
MS_DEF_RSA_SIG_PROV	"Microsoft RSA Signature Cryptographic Provider"
MS_DEF_RSA_SCHANNEL_PROV	"Microsoft RSA SChannel Cryptographic Provider"
MS_DEF_DSS_PROV	"Microsoft Base DSS Cryptographic Provider"
MS_DEF_DSS_DH_PROV	"Microsoft Base DSS and Diffie-Hellman Cryptographic Provider"
MS_ENH_DSS_DH_PROV	"Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider"
MS_DEF_DH_SCHANNEL_PROV	"Microsoft DH SChannel Cryptographic Provider"
MS_SCARD_PROV	"Microsoft Base Smart Card Cryptographic Provider"

表 3-1 Platform SDK 提供的 CSP 列表

由於 CSP 的操作關係到視窗作業系統的安全機制和架構，故必須小心的使用，操作 CSP 必須符合以下 3 個規定:

1. 應用程式不能直接存取、修改金鑰，存取的動作必須透過 CSP 提供的介面才能執行，因為所有的金鑰皆由 CSP 產製，應用程式使用金鑰必須透過特定的 handler，這一限制讓應用程式存取金鑰較不會有安全上的風險，
2. 應用程式不能對密碼學運算的細節做任何操作或改變，CSP 介面允許應用程

式選擇加密、簽章演算法，但實際的密碼運作都交由 CSP 完成。

3. 應用程式不能先前使用者的信用資料或驗證相關訊息，信用資料就是先前其他使用者透過安全原則(Security Principal) 所建立的身分鑑別實體，例如:密碼或是 Kerberos 協定票証，將來 CSPs 加入生物指紋的驗證機制時，應用程式不需修改驗證模組，只需修改傳給 CSPs 的變數資料結構。

(2) 資料編碼與解碼

在安全的通道傳送加密資料必須先序列化(Serialize)，例如:有兩台電腦透過網路或電話線傳送資料，傳送者的資料需先編碼成 0 與 1 組合而成的串列，資料訊息才能透過網路或電話線依序傳送給接收者並解碼為原始的資料格式。這段過程需透過軟體運算與硬體傳送才能完成。

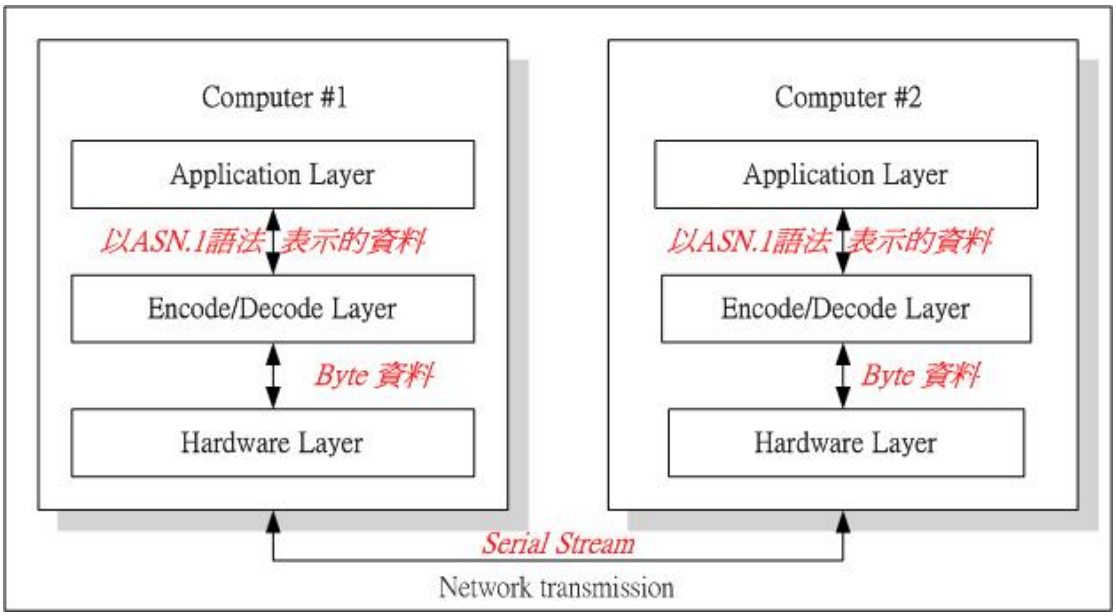

圖 3-1 傳送資料的編碼解碼流程

上圖說明兩台電腦透過網路傳送資料的流程，電腦 1 的應用層將欲傳送的原始資料交給編碼/解碼層，編碼/解碼層將資料編碼為 byte 資料格式交給硬體層，硬體層在將 byte 資料轉換為序列化串列並傳送給電腦 2，電腦 2 將硬體層收到序列化串列以先前敘述的程序逆向解碼為原始資料格式，在這過程中有一個重大的議題，編碼/解碼層如何將資料編碼或解碼成雙方皆可認可的資料格式呢?大部分的通訊協定會使用雙方皆認可的抽象資料或抽象物件表示方法，國際上認可的抽象

資料/物件表示法為 ASN.1(Abstract Syntax Notation One)，由 ITU 制定並經由 ISO (International Standard Organization)認可通過，ASN.1 定義在 CCITT Recommendation X.208 文件中，故電腦 1 的應用層必須將資料表示為 ASN.1 語法，在傳給編碼/解碼層，電腦 2 的編碼/解碼層在收到 byte code 後，需將 byte code 編碼為 ASN.1 語法的資料，如此接收者才能正確得到傳送者的原始資料格式。

(3) 金鑰容器(Key Container)

即金鑰儲存庫，用來儲存特並使用者所有的金鑰(公開金鑰和私密金鑰)，每一個容器都有其獨一無二的容器名稱，當使用 CryptAcquireContext()輸入容器名稱，就可以得到金鑰容器的 handle。

(4) 憑證貯藏(Certificate Store)

依照字面上解釋，即是憑證永久的貯藏倉庫，當程式使用到某一憑證，程式設計師可以只在記憶體創造或開啟一憑證貯藏，而不一定將憑證儲藏在永久的憑證貯藏，此憑證會隨著程式執行結束而消失。憑證貯藏的存在是為了要集中管理憑證和分類憑證，Windows 預先定義了四種不同的憑證貯藏: 個人(MY Stor-e)、信任的根憑證授權(Root Store)、企業信任(Trust Store)、中繼憑證授權(CA Store)，此四種不同階層的憑證貯藏，即如同先前所述 X.509 階層式架構一樣，簽發個人憑證(儲存在 MY Store)的發行者公鑰憑證，若亦儲存在信任的根憑證授權(Root Store)或企業信任(Trust Store)裡，則此個人憑證即受到信任且可以使用，若簽發個人憑證(儲存在 MY Store)的發行者公鑰憑證，沒有儲存在信任的根憑證授權(Root Store)或企業信任(Trust Store)裡，則此個人憑證即未受到信任且不能使用，往上一層來看，若簽發企業信任憑證的發行者公鑰憑證沒有儲存在中繼的根憑證授權(CA Store)裡，則此企業信任憑證和企業信任憑証所簽發的個人憑證皆不受到信任且無法使用，這裡的憑證貯藏和 X.509 憑證金字塔制度的觀念是相通的。

(5) Windows 的憑證資料結構

1. Certificate Context

Certificate Context 為電子憑證的 C 資料結構，包含:憑證貯藏的 handle、一指標指向原始編碼後的憑證體(Certificate Blob)、和一個指向憑證資訊(Certificate Info)資料結構的指標。CERT_INFO 為憑證的中心所在，包含憑證所有的基本資訊，其結構如同先前所述 X.509 v3 版本的電子憑證一般

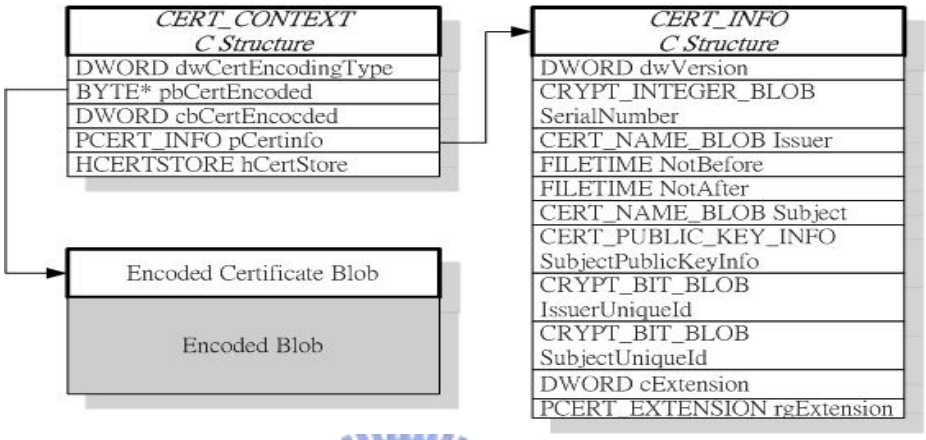

圖 3-2 CERT_CONTEXT 和 CERT_INFO 關係圖

若程式設計師想要將特定的資料寫入 CERT_INFO，資料需先寫入特定的資料結構(SDK 先定義好的)，再經由特定函式編碼(CertEncode())，編碼過後的資料方能寫入 CERT_INFO。

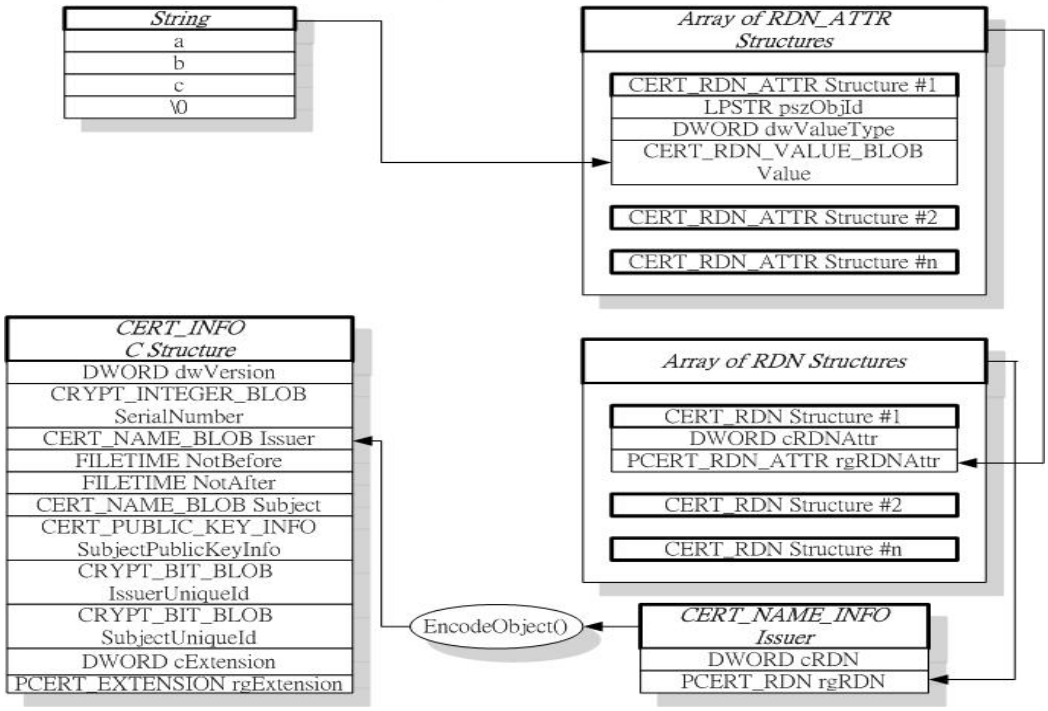

圖 3-3 編碼範例

上圖表示將 abc 字串加入 CERT_INFO 資料結構使得憑證的發行者名稱為 abc 過程，下列步驟詳細說明整個流程:

Ⅰ. 宣告發行者名稱的字串，即為包含 abc 的字串。

Ⅱ. 產生 CERT_RDN_ATTR 陣列並初始化陣列成員 CERT_RDN_VALUE_BLOB 為發行者名稱 ABC。

Ⅲ. 產生 CERT_RDN 結構並將結構成員指標 PCERT_RDN_ATTR 指向 CERT_RDN_ATTR 起始位址。

Ⅳ. 產生 CERT_NAME_INFO 結構並將結構成員指標 PCERT_RDN 指向 CERT_RDN 起始位址。

Ⅴ. 呼叫 CryptEncodeObject()將先前所產生的 abc、CERT_RDN_ATTR、CERT_RDN、CERT_NAME_BLOB 編碼為 byte 資料型態。

Ⅵ. 將編碼後的 byte 資料指派給 CERT_INFO 成員變數 CERT_NAME_BLOB，如此在 windows 平台上所看到的憑證發行者名稱即為 abc。

3. 憑證擴充屬性(Certifcate Extended Properties)

X.509 v3 憑證的憑證擴充屬性欄位值，在憑證產生之後就不能做任何的更改，只能讀不能寫；憑證擴充屬性屬於憑證本體的一部分，但發行者簽發時只簽署憑證本體，而不理會憑證擴充屬性，然而，在微軟 Windows 平台上以 CryptoAPI 產生的憑證，CryptoAPI 允許程式設計師動態增加憑證擴充屬性，要注意的是在 windows 平台上產製的憑證，其憑證擴充屬性無法在其他平台上正常運作。憑證擴充屬性可以包含以下資料:

- 和憑證公鑰成對的私鑰的儲存資訊
- 憑證可以使用的雜湊類型(MD5 或 SHA-1)
- 提供與憑證相關的使用者定義資訊

在微軟 windows 平台上，擴充屬性的值可以被附加到憑證上，微軟預先定義了屬性 ID 讓程式設計師可以藉由屬性 ID 鑑別擴充屬性的值所代表

的意義:

- CERT_KEY_PROV_HANDLE_PROP_ID:代表憑證所使用的特定 CSP
- CERT_KEY_PROV_INFO_PROP_ID:代表憑證相對應的私鑰
- CERT_SHA1_HASH_PROP_ID and CERT_MD5_HASH_PROP_ID:這兩個屬性指出憑證執行雜湊運算時，所支援的雜湊演算法。

第二節系統架構

2.1 系統架構簡介

本篇論文主要目的在於提供企業網路或區域網路安全的郵件通訊及安全的私鑰貯藏，以下圖說明:

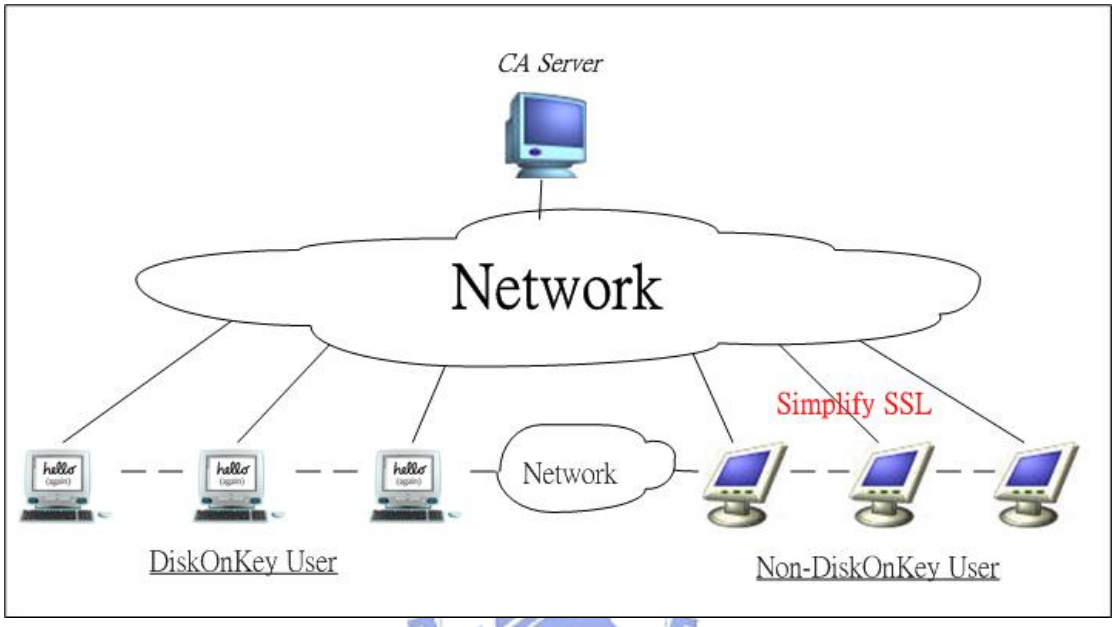

圖 3-4 系統架構

首先企業網路內必須要有一 CA 伺服器，負責簽發網路內所有員工和使用者憑證，企業內的員工如果有 DiskOnKey，在 CA 簽發憑證時會把憑證相對的私鑰藏在 DiskOnKey 內，企業主將 DiskOnKey 發給員工後，員工把 DiskOnKey 接在個人電腦上，透過網路連線取得先前 CA 簽發的公鑰憑證後，即可匯入公鑰憑證和私鑰到本機電腦上;企業內的員工，如果沒有 DiskOnKey，可以透過學生以 SSL 連線為基礎所修改的安全連線，向 CA 要求簽發公鑰憑證，待欲互相通訊的雙方匯入憑證於各自的電腦後，雙方即可用先前產生的憑證在 Outlook Express 上傳送加密和簽章的郵件。要注意的是:有 DiskOnKey 的員工在將公鑰憑證和私鑰匯入電腦後，無法經由探險家瀏覽器 Internet Explore 或主控台(Microsoft Manage-ment Console)匯出 PKCS#12 的私鑰檔，私鑰只會放在 DiskOnKey 和本機電腦內，且放在 DiskOnKey 的私鑰亦經由密鑰(Session key)加密;沒有 DiskOnKey 的

員工將公鑰憑證和私鑰匯入電腦後，無法經由探險家瀏覽器 Internet Explore 或主控台(Microsoft Management Console)匯出 PKCS#12 的私鑰檔，私鑰只會放在本機電腦內。

2.2 運作流程

本系統的架構主要分為三個角色和三大部分，三個角色分別為:產生自簽憑證(SelfSigned Certificate)的 CA Server、有 DiskOnKey 的使用者和沒有 DiskOnKey 的使用者。三大部分為:

1. 在 Server 產生自簽的 CA 憑證 SelfSignedSIG.cer 和 SelfSignedEX.cer 並匯入 CA 伺服器，SelfSignedSIG.cer 用來簽發憑證，SelfSignedEX.cer 用來建立安全的網路連線。

圖 3-5 CA 產生憑證流程

2. 有 DiskOnKey 的使用者，需先將 DiskOnKey 拿到 CA 伺服器產生由 CA 自簽憑證簽發的公鑰憑證，程式並將公鑰憑證相對應的私鑰放到 DiskOnKey;之後使用者將公鑰憑證加上 DiskOnKey 內的私鑰匯到自己的個人電腦裡。

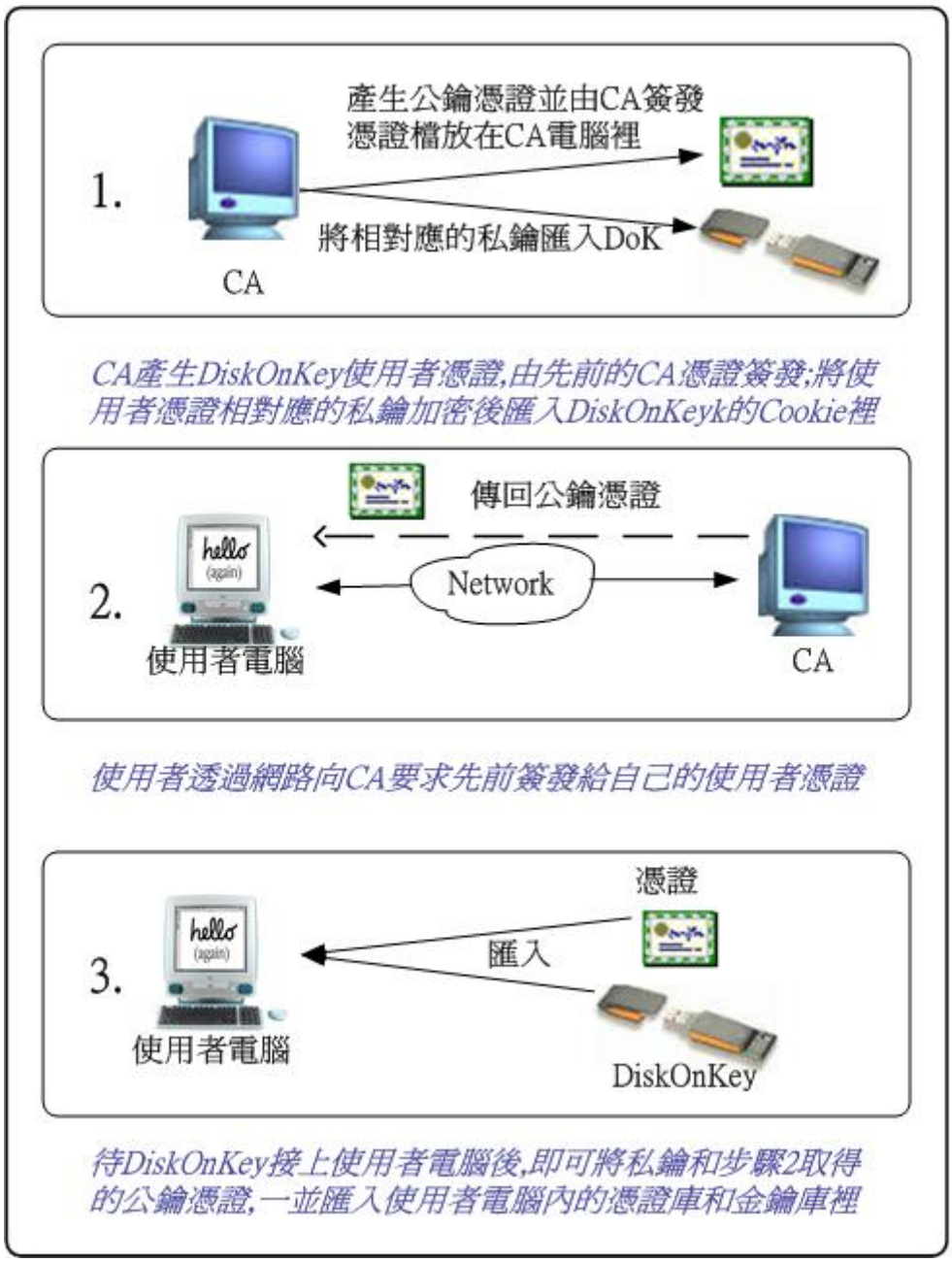

圖 3-6 有 DiskOnKey 的使用者流程

3. 沒有 DiskOnKey 的使用者電腦透過安全的網路連線向 CA 要求簽發使用者憑證，CA 在簽發完畢後傳回簽發的使用者憑證，使用者再將 CA 簽發的憑證和私鑰一起匯入本機電腦中。

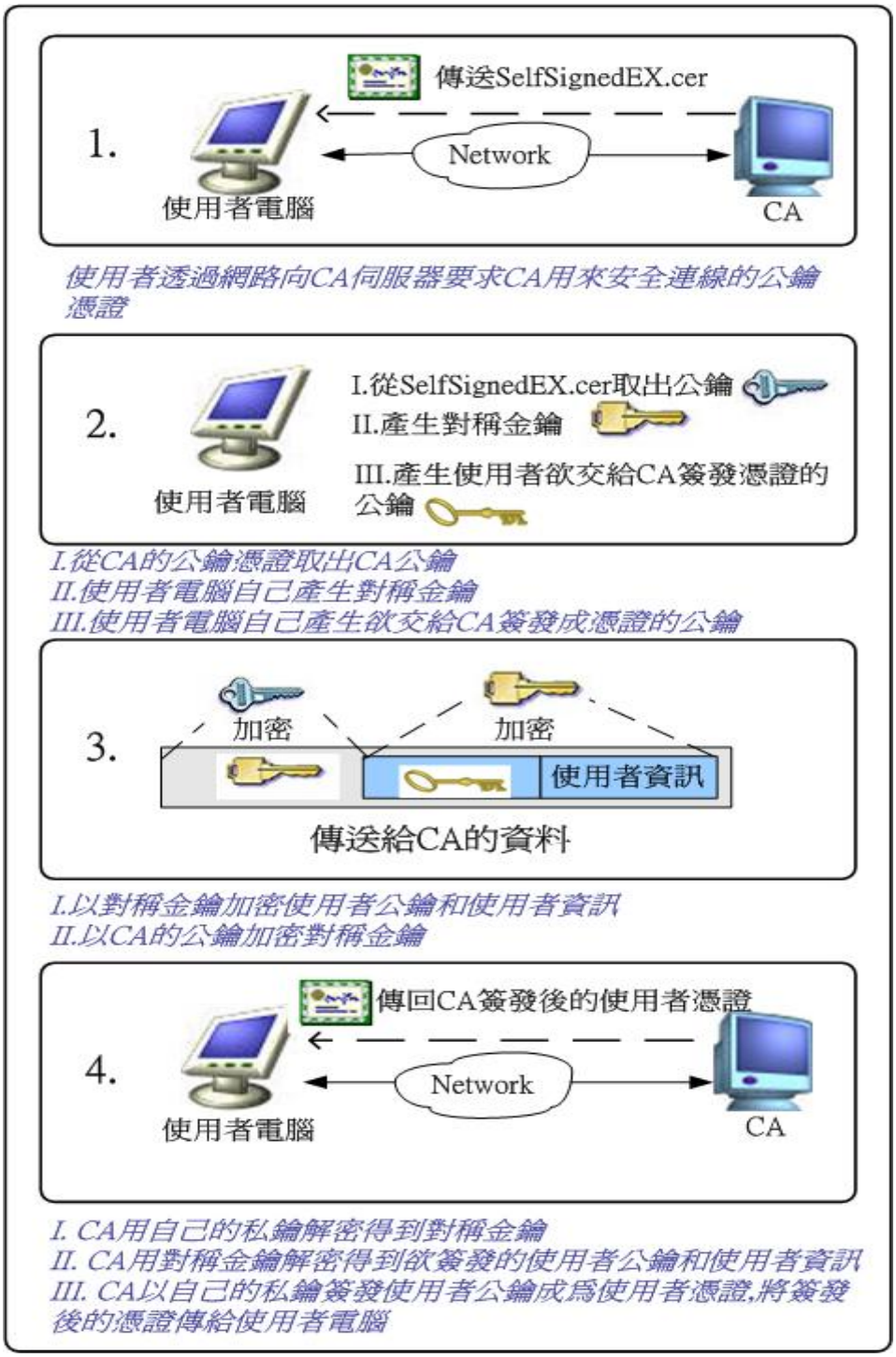

圖 3-7 沒有 DiskOnKey 的使用者流程

第三節 程式碼撰寫流程與研究

3.1 程式使用函式說明

本論文所有程式僅適用在微軟 Windows 平台上執行，使用 Platform SDK 提供的函式開發，函式說明如下:

Platform SDK 函式	說明
CertStrToName()	將使用者輸入的字串編碼成填入 X.509 憑證欄位的名稱。
CryptAcquireContext()	取得特定金鑰庫(Key Container)和管理此金鑰庫的 CSP 的處理(handle)。
CryptGenRandom()	將隨機位元填滿緩衝區(Buffer)。
CryptGenKey()	隨機產生對稱金鑰或公鑰私鑰對。
CryptExportPublicKeyInfo()	以 Public_Key_info 資料型態匯出目前 CSP 管理的金鑰庫裡的公鑰。
CryptCreateHash()	初始化雜湊物件。
CryptHashData()	將輸入的資料執行雜湊運算。
CryptGetHashParam()	取得雜湊運算後的值。
CryptEncodeObject()	將輸入的資料編碼為函式內引數 lpszStruct 所指定的資料型態。
CryptSignAndEncodeCertificate()	將資料編碼為 X.509 憑證並簽署之。
CryptDeriveKey()	從資料的雜湊值產生密鑰(session key)，相同的資料產生相同的雜湊值，相同得雜湊值產生相同的密鑰。

Platform SDK 函式	說明
CryptExportKey()	匯出 CSP 管理的密鑰或公鑰私鑰對。
CertOpenStore()	開啟憑證貯藏。
CertAddEncodedCertificateToStore()	將編碼過後的憑證儲存在憑證貯藏裡。
CertSetCertificateContextProperty()	設定憑證的憑證擴充屬性
CertFindCertificateInStore()	由引述輸入的資訊找尋儲存憑證貯藏裡的憑證。
CertGetCertificateContextProperty()	取得憑證擴充屬性資訊。
CertFindExtension()	取得憑證擴充陣列的第一個擴充屬性
CryptDecodeObject()	將輸入的資料解碼為函式內引數 lpszStruct 所指定的資料型態。
CryptImportKey()	將密鑰或公鑰私鑰對由 key blob 型態匯入 CSP 並取得輸入金鑰的處理(handle)。

表 3-2 Platform SDK 函式

使用的 DiskOnKey SDK 3.1.2.6 的函式說明如下:

DiskOnKey SDK 函式	說明
DOKSDK_Init()	初始化 DiskOnKey SDK。
startListen()；	喚起 startListen 方法並等待(listen) DiskOnKey 銜接上電腦。
detachDevice()	從 DiskOnKey 裝置卸除 CDOK 物件。
stopListen()	停止接收從等待物件(listener object)傳回的結果(event)。
DOKSDK_fini()	終結 DiskOnKey SDK。
WaitForSingleObject()	等待 DiskOnKey 裝置連接上電腦所傳回的結果(event)。

getDeviceCapacity()	取得 DiskOnKey 所有區域容量大小。
minPartitionSize()	取得 DiskOnKey 裝置 Public、Safe、Dummy 分割的最小分割容量。
supportDeposit()	驗證 DiskOnKey 裝置是否支援隱藏分割功能。
getDeviceType()	傳回 DiskOnKey 裝置是否具有可調整分割功能或固定分割。
configureDiskPartitions()	格式化 DiskOnKey 並配置隱藏分割容量。
writeHiddenString()	將字串值寫入 DiskOnKey Hidden Area 特定區間的特定進入點
readHiddenString()	讀取 DiskOnKey Hidden Area 特定區間的特定進入典禮的字串資料。

表 3-3 DOK SDK 函式

3.2 CA 產生憑證程式碼研究

本段程式主要在 CA 產生兩個 CA 自簽憑證 SelfSignedSIG.cer 和 SelfSigned EX.cer，並將這兩個憑證匯入信任的根憑證授權(Root)憑證貯藏裡，私鑰則只儲存在本機電腦上相對應的金鑰庫(key ocntainer)。本段程式的流程如下:

1. 首先選擇接下來使用的密碼服務管理員(Cryptgraphic Service Provider，CSP)，在這裡我們選擇 Microsoft Enhanced Cryptographic Provider ， Microsoft Enhanced Cryptographic Provider 提供 1024bits 長度的公鑰。
2. 由使用者輸入即將產生的憑證的 X.509 名稱，例如:”CN=Certificate , OU=Su -pport , O=Company , L=Seattle , E=abc@def.ghi.tw”，其中 CN:憑證名稱，OU:組織單位名稱，O:組織名稱，L:所在城市，E:電子郵件位址。並將使用者輸入的 X.509 名稱編碼為 ASN.1 型態資料。
3. 產生全球通用識別碼(Universal Unique Identifier，UUID)，以此識別碼當成金

鑰庫名稱(key container name)，長度 128bits。

4. 由 CSP 產生公鑰/私鑰對。

5. 產生一個空的 CertInfo 資料結構，填入以下資料:

Ⅰ.憑證版本:X509_V3。

Ⅱ.憑證序號:由 SDK 函式隨機產生 8 bytes 數字，填入此欄。

Ⅲ. 簽章演算法:由使用者輸入決定,有 MD5 和 SHA-1 兩種,預設為 SHA-1。

Ⅳ. 發行者: 步驟 2 編碼後的 X.509 名稱。

Ⅴ. 有效日期:由使用者輸入，以月為單位。

Ⅵ. 使用者(主體):步驟 2 編碼後的 X.509 名稱。

Ⅶ. 公開金鑰資訊:將步驟 4 產生的公鑰轉成 PUBLIC_KEY_BLOB 型態填入。

Ⅷ. 憑證擴充:

a. 主體金鑰識別元:將公鑰值執行 SHA-1 運算，雜湊值編碼為使用者金鑰識別碼型態填入此欄。

b. 金鑰使用方式:若憑證使用目的為簽發憑證，則將支援數位簽章、私鑰無法匯出或拋棄、支援以金鑰簽發憑證、支援以金鑰簽發憑證撤銷清單四個旗標(Flag)寫入此憑證擴稱欄位。若憑證使用目的為交換金鑰連線，則將支援數位簽章、支援明文加密、支援加密對稱金鑰、支援不同機器共用對稱金鑰四個旗標(Flag)寫入此憑證擴充欄位。

c. 基本限制:此欄位用來識別憑證是否為 CA 憑證?在此輸入 TRUE，若為 CA 憑證，可簽發的多手個憑證階層?在此輸入 1，表示可簽發的對象為一般使用者(即終端節點，使用者憑證不能再簽發憑證)。

d. 授權金鑰識別元:因為在此憑證為自簽憑證，故授權金鑰識別元的金鑰 ID 與主體金鑰識別一樣，填入相同值; 授權金鑰識別元的憑證發行者與發行者一樣,填入相同值; 授權金鑰識別元的序號和憑證序號一樣，填入相同值。

6. 以步驟 4 的私鑰簽發此憑證，所使用的簽章演算法即為 CertInfo 內的簽章演算法欄位指定的方法。

7. 開啟信任的根憑證授權(Root)憑證庫(Certificate Store)，將簽發後的憑證加入憑證庫裡。

8. 設定此憑證相對應的私鑰所在的金鑰庫(key container)，CSP 名稱，在此為 Microsoft Enhanced Cryptography Service Provider，CSP 支援的演算法，在此為 PROV_RSA_FULL。

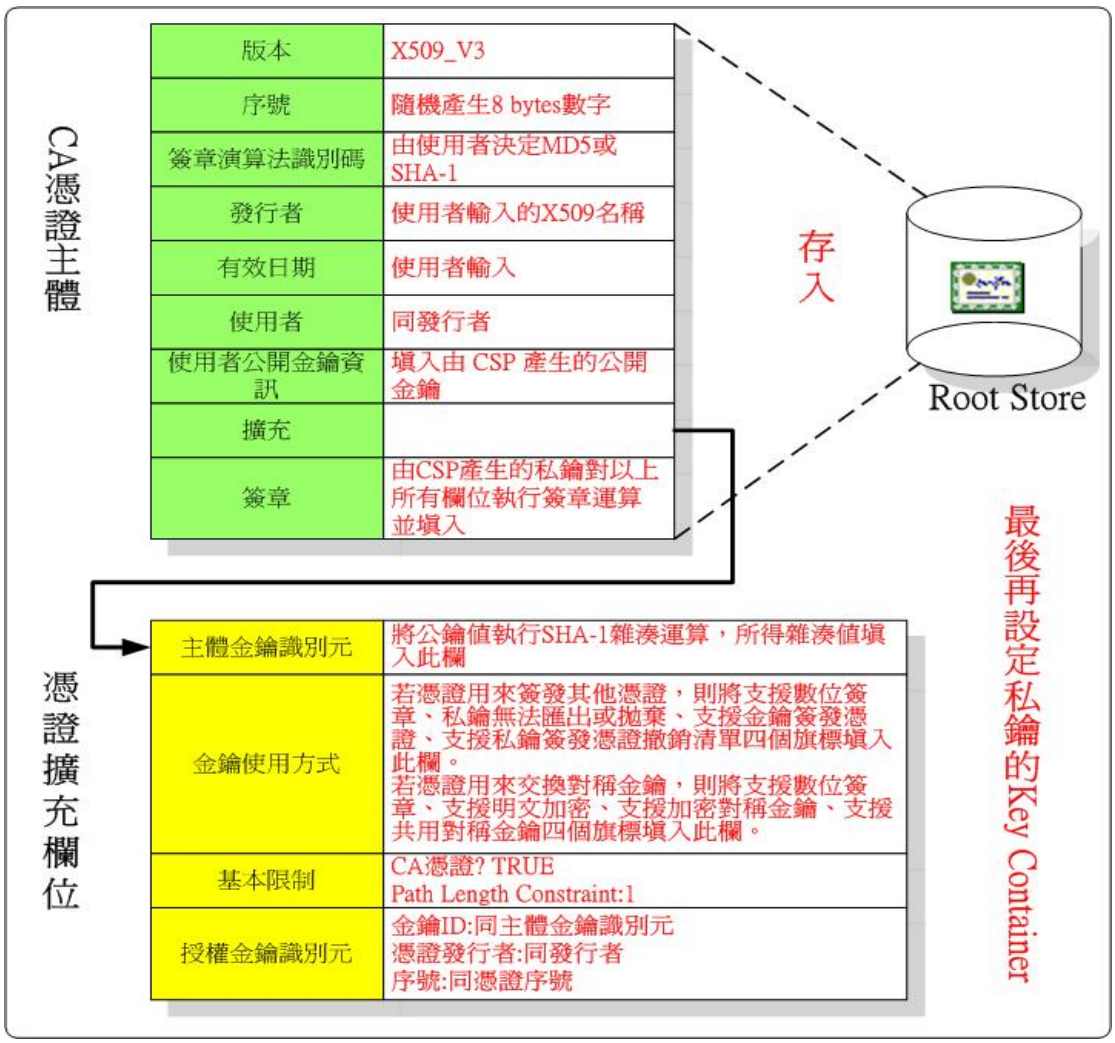

圖 3-8 以 CryptoAPI 產生 CA 憑證

3.3 DiskOnKey 使用者程式碼研究

本程式目的要產生 DiskOnKey 使用者憑證，由上一段程式產生的 CA 憑證

簽發，公鑰憑證以檔案方式(.cer)儲存在 CA 電腦裡，私鑰以使用者輸入的密碼得到的對稱金鑰加密後儲存在 DiskOnKey Hidden Area 裡。待使用者回到自己的電腦上，即可以 DiskOnKey 裡的私鑰配上先前產生的公鑰憑證匯入使用者電腦裡，匯入後私鑰無法以 PKCS#12 格式匯出。私鑰只存在 DiskOnKey 裡。分成兩個部分，1. 產生憑證並將私鑰匯入 DiskOnKey。2. 從 DiskOnKey 匯入私鑰到使用者電腦。以下分說明:

(1)產生憑證並將私鑰匯入 DiskOnKey

產生使用者憑證的部分和上一節產生 CA 憑證部分大致相同，以下僅列和上節不同的程式流程:

1. 因為此憑證由上一節產生的 CA 憑證簽發，故需先找出 CA 憑證的主體金鑰識別元、CA 憑證私鑰的 CSP，接著才能簽發其他憑證，實做 FindCertificate()函式，用來取得 CA 憑證的相關資訊，FindCertificate()流程如下:

Ⅰ. 開啟中繼跟憑證授權憑證庫(Root Store)。

Ⅱ. 以 CA 憑證的 X.509 名稱(CN=…)在 Root Store 搜尋 CA 憑證本體。

Ⅲ. 憑證找到後再找尋提供 CA 憑證相對應私鑰的 CSP。

Ⅳ. 取得 CA 憑證的主體金鑰識別元(KeyID)。

Ⅴ. 解碼主體金鑰識別元為 Byte 資料型態。

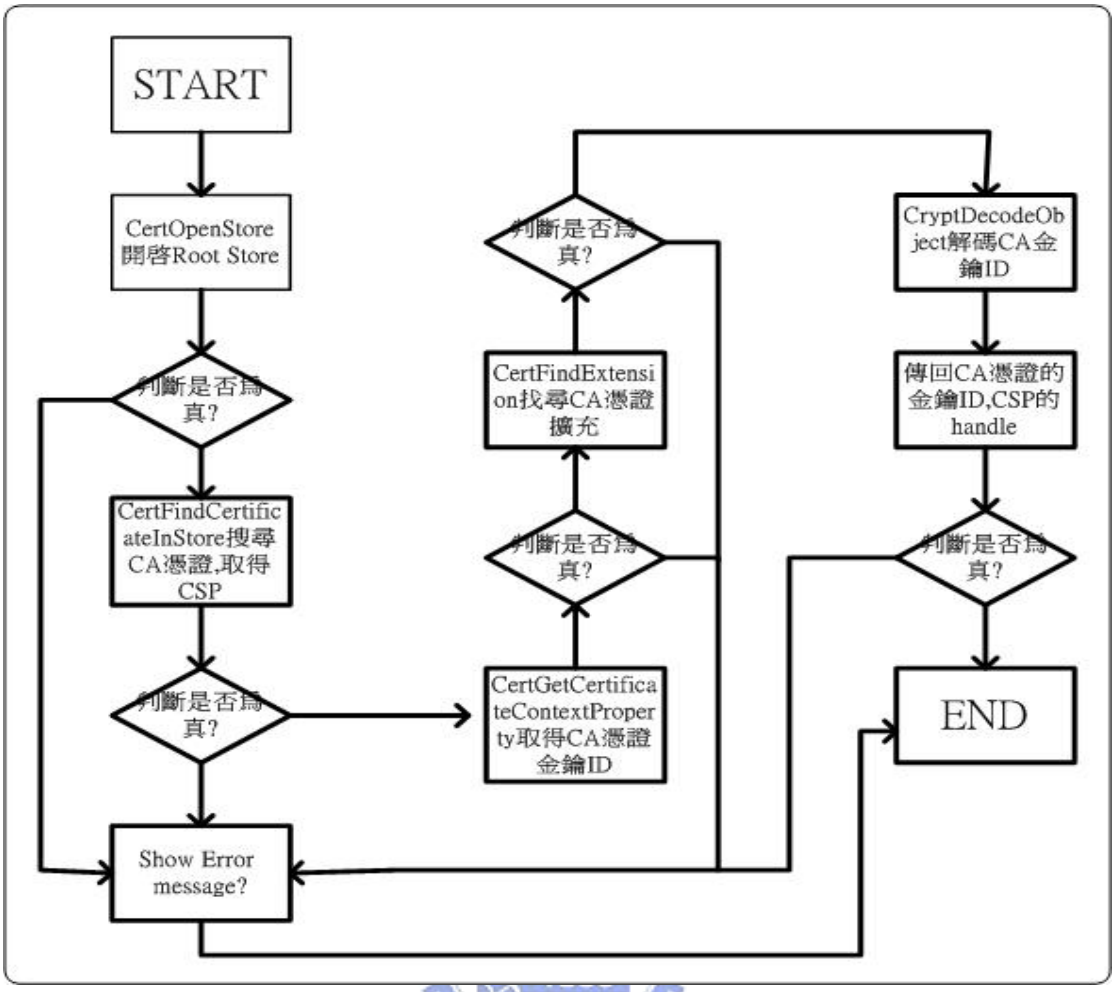

圖 3-9 FindCertificate 函式流程

2. 使用者憑證的發行者欄填入 CA 憑證內使用者欄位的內容。

3. 憑證擴充部分:

Ⅰ. 因為憑證用來加解密電子郵件及簽章,所以金鑰使用方式設為支援數位簽章、支援明文加密、支援加密對稱金鑰、支援不同機器上共用對稱金鑰四個旗標寫入此欄位。

Ⅱ. 因為憑證用來加解密電子郵件及簽章，所以增加一憑證擴充欄位-增強金鑰使用方法，填入安全的電子郵件旗標。

Ⅲ. 基本限制欄內是否為 CA 憑證填入 FALSE。

Ⅳ. 授權金鑰識別元改成由 FindCertificate()函式取得的 CA 憑證主體金鑰識別元。

4. 以 FindCertificate()函式取得的 CA 私鑰 CSP 搭配 CA 憑證簽章演算法欄

位內指定的方法簽發此憑證，寫入簽章欄裡。

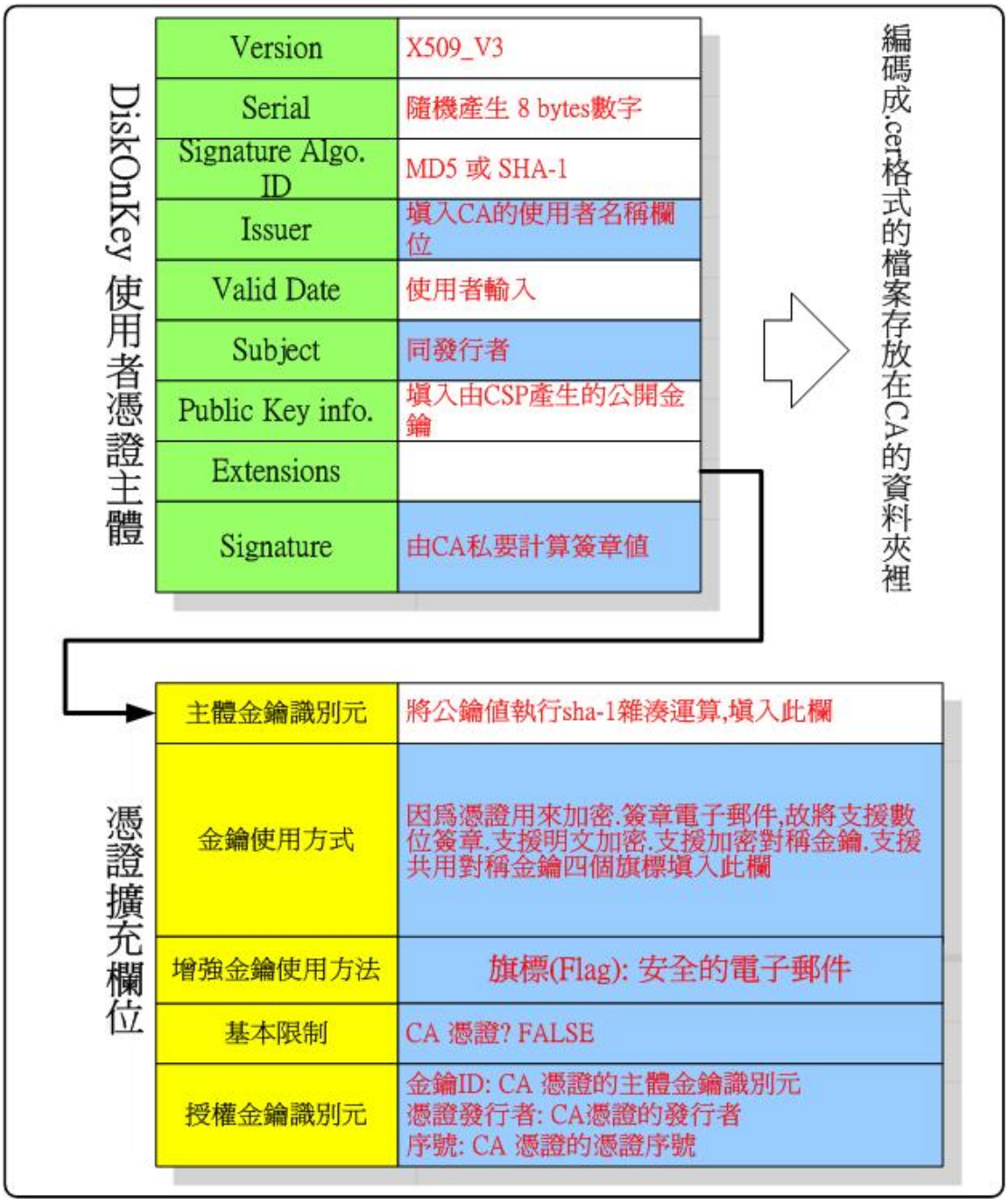

圖 3-10 DiskOnKey 使用者憑證產生流程

5. 將使用者憑證編碼成.cer 檔，存在 CA 電腦裡。

6. 格式化 DiskOnKey。

7. 利用使用者輸入的密碼得到對稱金鑰，以此對稱金鑰加密使用者憑證的私鑰並寫入 DiskOnKey 的 Hidden Area 裡，要注意的是 DiskOnKey 只接受字串寫入 Hidden Area，但加密後的私鑰為長度 508 位元組的連續 Byte Data

且有多個 0(16 進位)存在，0 轉成字串為 NULL，故寫入 DiskOnKey Hidden Area 時，無法完整寫入加密後私鑰，學生使用的方法為:

Ⅰ. 分割 508 個位元組為 254+254 個位元組，前 254 個位元組為 A，後 254 個位元組為 B。

Ⅱ. 在 A 的前面加入一個 byte 資料,在此稱為 Difference,加了 Difference 的 A 長度為 255 個 byte，計算 Difference 的值使得 A 內所有的值加上 Differe -nce 後不為 0(16 進位)，根據鴿籠原理，此 Difference 的值必存在。

Ⅲ. 將 Difference+A 轉換為字串值寫入 DiskOnKey Hidden Area 裡，Difference+B 部分，同理。

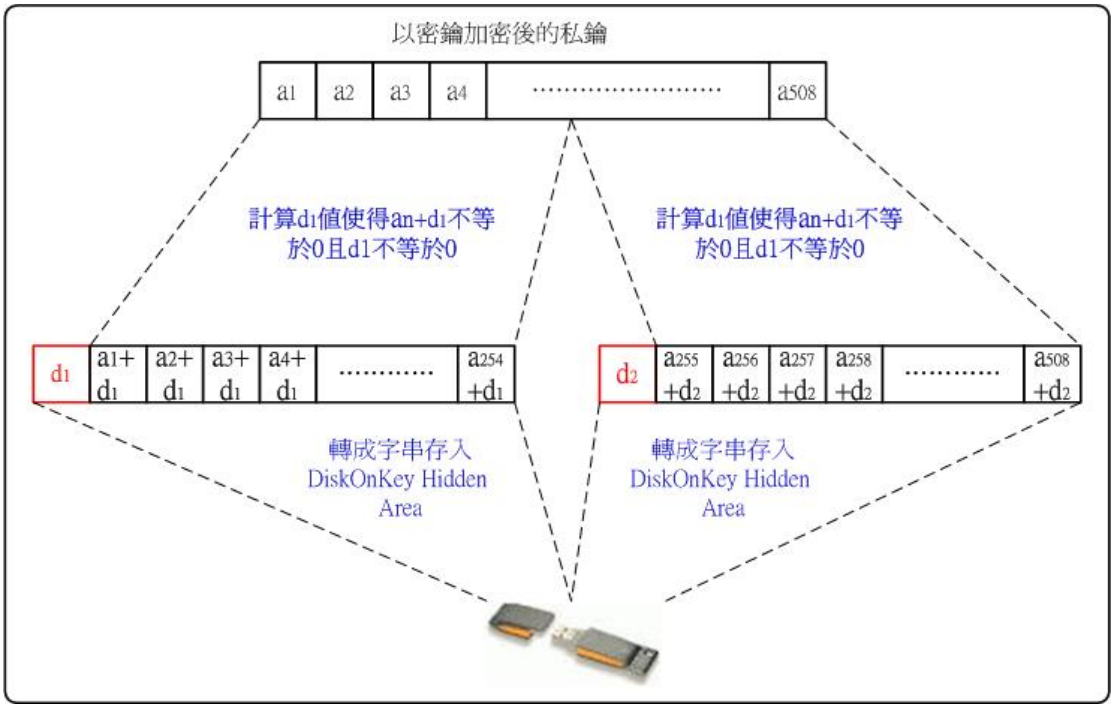

圖 3-11 寫入 DiskOnKey Hidden Area 的方法

8. 將使用者輸入的密碼以 MD5 執行雜湊運算，寫入 DiskOnKey Hidden Area 裡。

(2) 從 DiskOnKey 匯入私鑰到使用者電腦

程式流程如下:

1. 計算使用輸入的密碼 MD5 雜湊值，驗證此雜湊值和存在 Hidden Area 內

值是否相同?若是，繼續步驟 2，若否，顯示輸入錯誤密碼訊息，密碼輸入錯誤 3 次，執行 DiskOnKey 格式化。

2. 取的 Hidden Area 內的值，還原成原先對稱金鑰加密後的值。
3. 利用正確的使用者輸入密碼得到對稱金鑰，以對稱金鑰解密步驟 2 的值，得到私密金鑰值。
4. 開啟個人憑證貯藏(MY Store)，加入使用者憑證並設定私鑰。

3.4 沒有 DiskOnKey 的使用者程式碼研究

本程式主要目的為以 SSL 的觀念完成的 TCP 網路連線取得 CA 簽發後的公鑰憑證，搭配使用者電腦上的私鑰匯入使用者電腦裡。產生憑證的不分和上一節大致相同，以下僅列出不同的程式流程:

1. 透過 TCP 連線(未加密)取得 CA 的公鑰憑證 SelfSignedEX.cer，此憑證主要用來建立安全的連線，不能簽發憑證。
2. 從 CA 公鑰憑證 SelfSignedEX.cer 裡解碼出公鑰值 P。
3. 使用者輸入密碼，以密碼值產生新的對稱金鑰 S，另位使用者電腦自行產生公鑰 Pu/私鑰值 Su。
4. 以公鑰值 P 加密對稱金鑰 S，以對稱金鑰 S 加密使用者資訊。使用者資訊包括:欲交由 CA 簽發為公鑰憑證的公鑰值 Pu，簽發後憑證檔案名稱，使用者憑證的 X.509 名稱…等。
5. 透過 TCP(加密)連線傳送給 CA。
6. CA 收到後，使用 SelfSignedEX.cer 憑證的私鑰解密得對稱金鑰值 S，以對金鑰值解密得使用者資訊，依照使用者資訊的內容，產生憑證由 CA 憑證 SelfSignedSIG.cer 的私鑰簽發此憑證，簽發後的憑證，可以選擇以對稱金鑰 S 加密簽發後的憑證或不加密(因為傳送內容為公鑰憑證)，回傳給使用者。

7. 使用者收到公鑰憑證後，搭配先前產生的私鑰 Su，一同匯入使用者憑證貯藏裡，並設定私鑰所在的金鑰庫名稱。

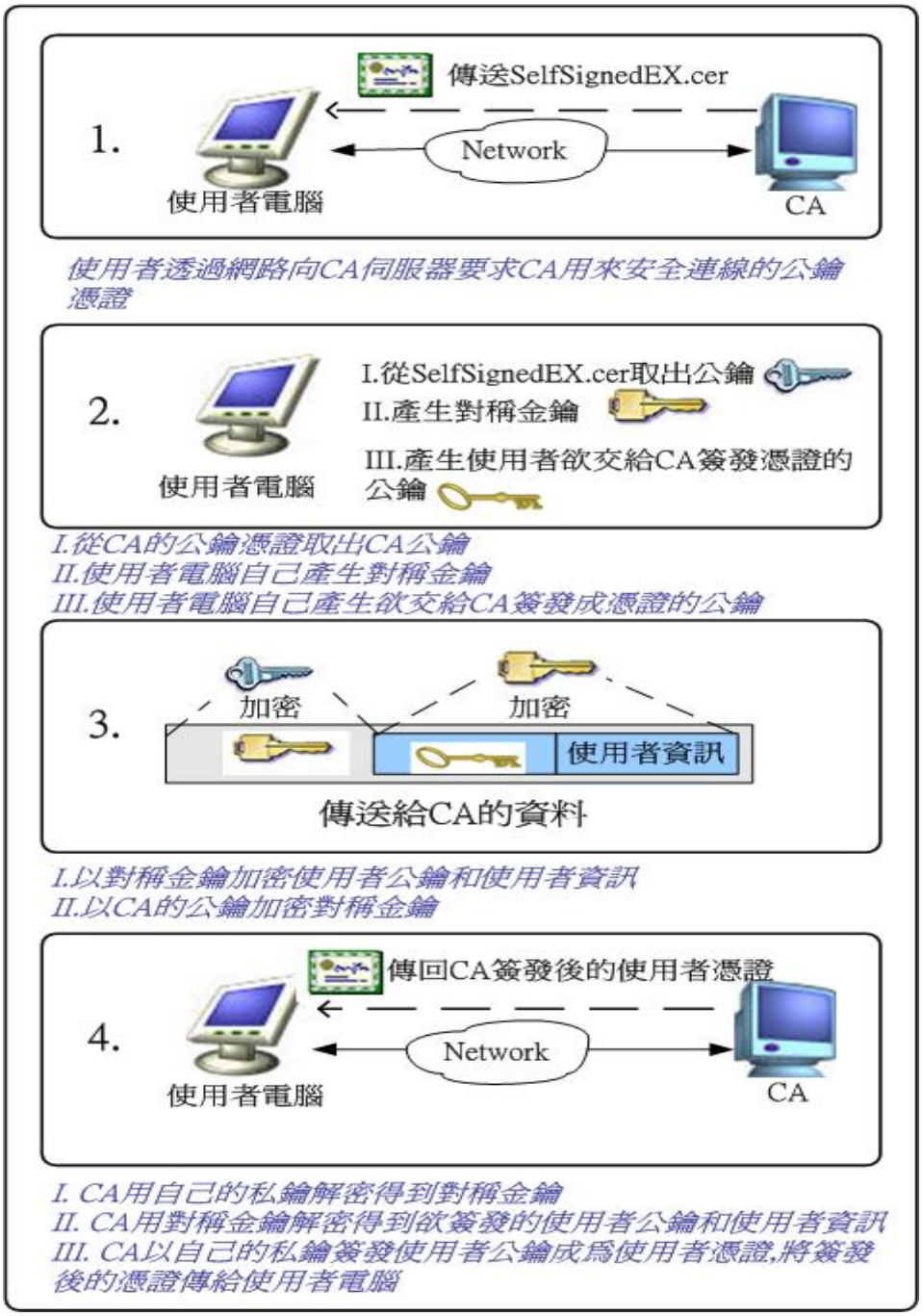

圖 3-12 安全連線流程

第四章

實驗結果與分析

實驗需要 3 台機器，一台當 CA，一台當有 DiskOnKey 的 User，一台當沒有 DiskOnKey 的 User。CA 的 IP 為 140.113.216.125，接受安全連線的 port:5160，有 DiskOnKey User 的 IP 為 140.113.216.123，沒有 DiskOnKey User 的 IP 為 140.113.167.202。以下分別就四個不同的部分分別說明 CA 產生憑證、有 DiskOnKey 的使用者、沒有 DiskOnKey 的使用者、使用憑證於 Outlook Express。

(1)CA 產生憑證部分：依序產生兩個憑證 SelfSignedSIG.cer 和 SelfSignedEX.cer，SelfSignedSIG.cer 的 X509 name 為”CN=RootCA, OU=NCTU, O=EC621, L=Taiwan, S=Taiwan”，KeyType 為 AT_SIGNATURE，雜演算法 SHA-1，有效日期 100 個月。SelfSignedEX.cer 的 X509 name 為”CN=CAX, OU=NCTU, O=EC621, L=Taiwan, S=Taiwan”，KeyType 為 AT_KEYEXCHANGE，雜演算法 SHA-1，有效日期 100 個月。產生的憑證如下:

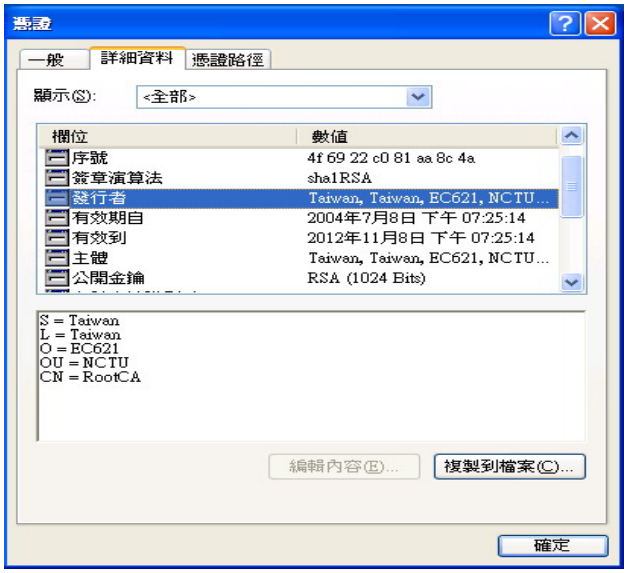

圖 4-1 SelfSignedSIG.cer 憑證內容

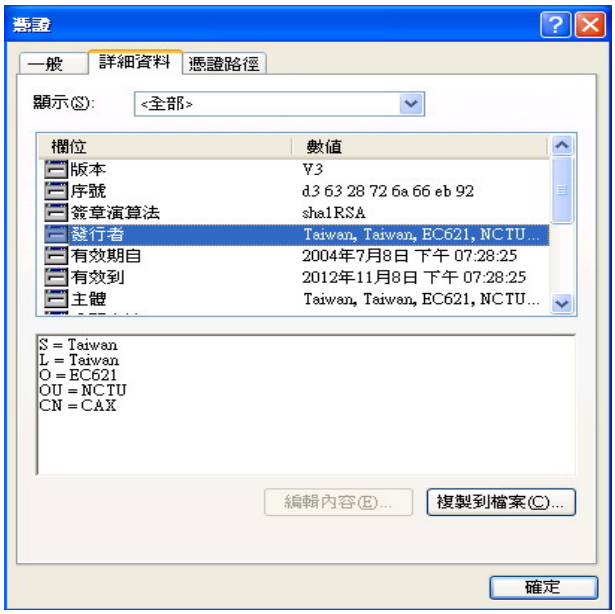

圖 4-2 SelfSignedEX.cer 憑證內容

有 DiskOnKey 的 User 部分:先到 CA 電腦產生使用者憑證，使用者憑證的 X.509 name: ”CN=chengwei, E=chengwei@csie.nctu.edu.tw”，KeyType 為 AT_KEYEXCHANGE，雜演算法 SHA-1，有效日期 80 個月，憑證檔檔名 ch.cer。憑證產生後，接著帶 DiskOnKey 回到使用者電腦匯入憑證和 DiskOnKey 上的私鑰。產生的憑證如下:

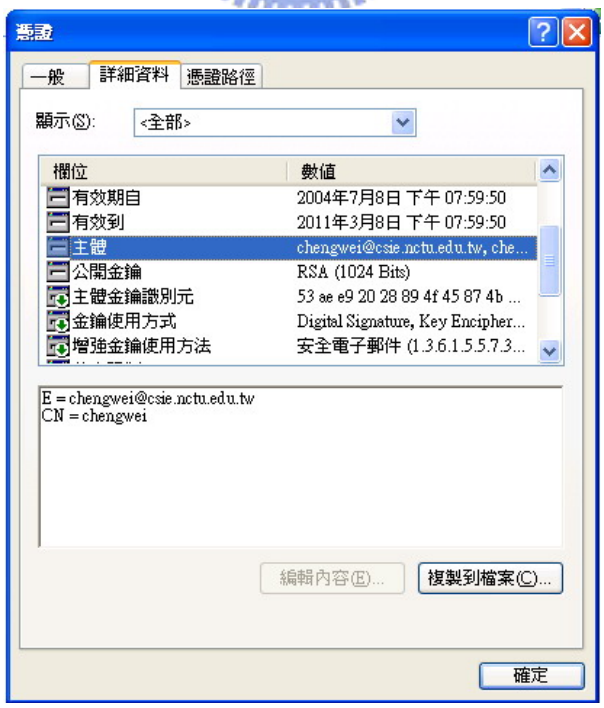

圖 4-3 ch.cer 憑證內容

沒有 DiskOnKey 的 User 部分:使用者憑證的 X.509 name 為"CN=jackal, E=u6191579@tknet.tku.edu.tw, S=Taiwan"，KeyType 為 AT_KEYEXCHANGE，雜演算法 SHA-1，有效日期 80 個月，憑證檔檔名 jk.cer。TCP 連線後公鑰憑證 jk.cer 和私鑰自動匯入電腦裡，CA 和 User 之間的連線為加密連線，產生的憑證和加密後的封包如下:

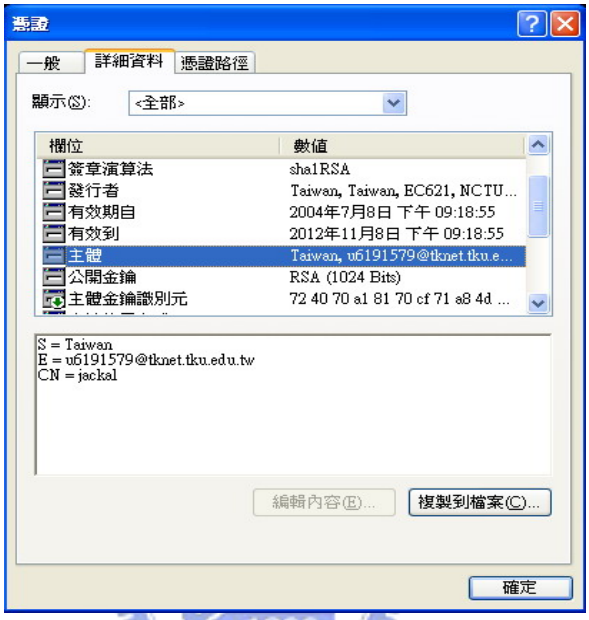

圖 4-4 jk.cer 憑證內容

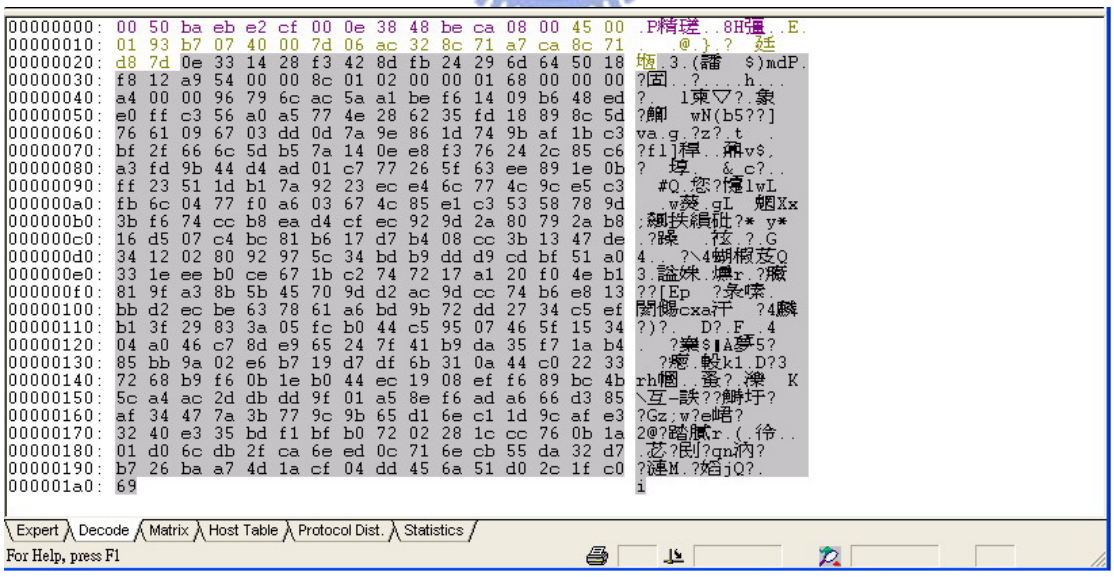

圖 4-5 傳送給 CA 加密的使用者資訊

使用者分別將自己的憑證和收件者的憑證匯入 Outlook Express 後，即可傳送加密及簽章的郵件。以 DiskOnKey User:chengwei，E-mail:chengwei@csieNctu.edu.tw

為寄件者，收件者為沒有 DiskOnKey User:jackal，E-mail:u6191579@tknet .tku.edu.tw，寄件者必須要有自己的公鑰憑證和私鑰才可以簽章，要有收件者的公鑰憑證才可加密郵件。收件者必須要有自己的公鑰憑證和私鑰才可解密，要有寄件者的公鑰憑證才可驗證簽章。Jackal 收到的郵件如下：

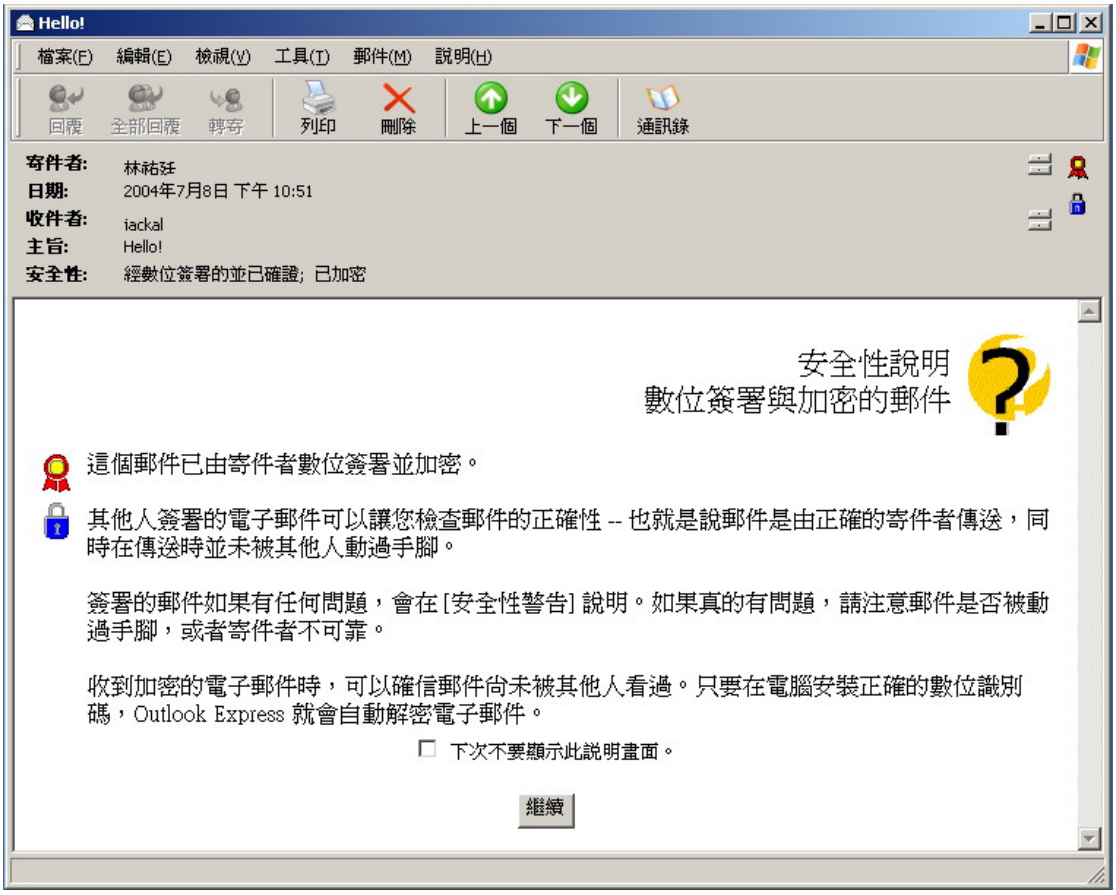

圖 4-6 加密和簽章後的郵件

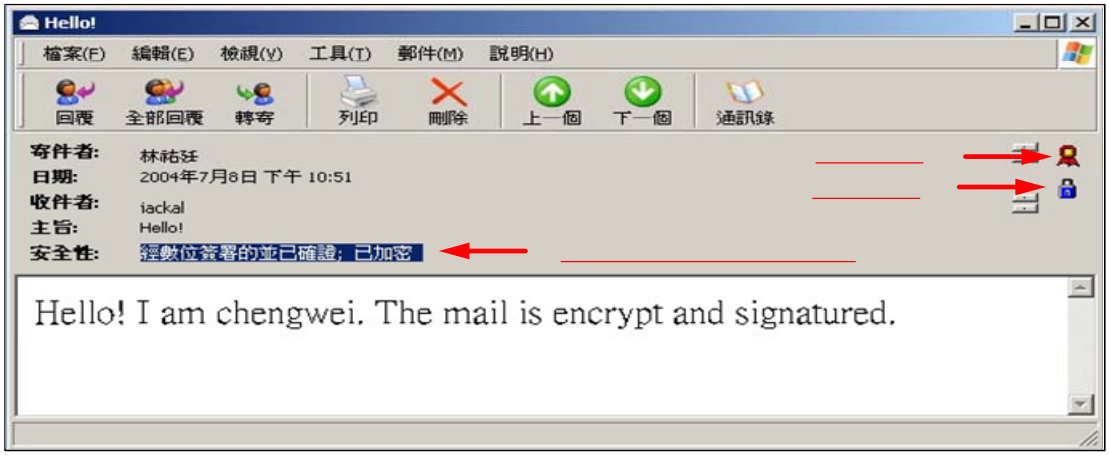

圖 4-7 郵件按下繼續後

以上兩個圖顯示可將本系統產生的憑證應用於 Outlook Express 中。

第五章

結論與未來展望

第一節 結論

本研究的在於實現私密金鑰的可攜性，讓使用者可以隨身攜帶金鑰，隨處上網執行電子郵件加密和簽章，若儲存的憑證由網路上的電子商務網站、網路銀行、或政府憑證中心所發行，更可以隨處執行電子商務、網路下單、網路報稅、網路銀行、簽章等和數位憑證有關的應用，若遺失 DiskOnKey，私密金鑰的值經過加密，竊取者不易解得正確的金鑰值，加上密碼輸入錯誤 3 次立即格式化 DiskOnKey 機制，讓私密金鑰多加了一層保護，使用者密碼不侷限於數字，一般的字元和符號皆可當成 DiskOnKey 密碼輸入，讓密碼的排列組合變化更多，提高了暴力解法的難度。數位憑證技術發明已有一段時間，由於社會大眾對數位憑證相關技術和應用並不了解，加上憑證可攜性不足，造成數位憑證普及率低，民眾使用意願低，僅有少數了解數位憑證的人在使用。以往的數位憑證提供使用者匯出金鑰憑證到磁片(以網路下單為例)，並提供 6 位數數字密碼保護，但磁片的體積較大可攜性不足，一般使用者都有許多磁片，使用上不易分辨儲存金鑰憑證的磁片且容易遺失，且僅提供 6 位數密碼保護，保密性不夠，一張磁片的容量約 1.44MB，僅能儲存一筆金鑰憑證，若使用者申請的多個網路憑證，更加容易混淆或遺失。DiskOnKey 的 Hidden Area 容量有 15Kbyte，可儲存約 30 筆私密金鑰和密碼雜湊值，使用者可自定密碼提示，利用 DOK SDK 亦可在隱藏區儲存 30 筆以上密碼提示，使用上更加方便，提供更多數位憑證共用 DiskOnKey，加上金鑰隱藏在藏在 Hidden Area 裡，所存的金鑰值是看不到的，只有應用程式能存取。

更提高了金鑰的隱密性。只要有網路連線就可以取得 DiskOnKey 上相對應的公鑰憑證，就可以匯入私鑰，郵件即能加密和簽章。若能降低 DiskOnKey 的生產成本，相信會對日後的推廣工作上有所幫助。

第二節 未來展望

未來還是有一些功能可以往下繼續探討:1. 增加一機制，當 DiskOnKey 一接上電腦，自動向 CA 取得公鑰憑證，搭配 DiskOnKey 上的私鑰只要密碼正確立刻自動匯入本機電腦。2. 增加一機制，當 DiskOnKey 一卸除本機電腦，立刻刪除電腦上的私鑰。3. 提供多筆私密金鑰、密碼雜湊值存在 Hidden Area 裡。

其實 DiskOnKey 上本身即具有長度 512bits 的公鑰和私鑰，原本 M-System 的設計理念為:將私鑰燒錄在 DiskOnKey 硬體裡，公鑰可由 SDK 取得，若要解密資料，資料傳送給 DiskOnKey，利用 DiskOnKey 本身的 ARM 7 處理器解密資料並傳回給作業系統，簽章時亦同。無奈到現在為止 DiskOnKey 的 SDK 僅提供 63bytes 的資料區段解密，DiskOnKey 的公鑰長度為 64bytes，經由 DiskOnKey 公鑰加密後的資料也必定為 64bytes 的整數倍，故解密時，亦需以 64bytes 為區段分段解密，才能解的正確值，若一次僅解密 63bytes，因為雪崩效應，解密資料必與原資料相去十萬八千里，這一點實在想不通 M-System 在想什麼，也許是 DOK SDK 還沒開發完成吧!再者若 DiskOnKey 提供 64bytes 解密資料的 SDK，但在 Windows 架構下，憑證驗證簽章和解密資料都必須要把私密金鑰存在 key container 裡，以 DiskOnKey 的設計，除非 Windows 提供新的 API，或者 Windows 平台上所有有關憑證的應用，M-System 都必須重新開發新的應用軟體來支援 DiskOnKey 的私鑰必不可匯出的特性，這樣的工程太過龐大，也太過耗費經費了，也許這就是 M-System 所想的吧。

參考文獻

［1］Rivest, R.; Shamir, A.; and Adleman, L. " A Method for Obtaining Digital Signatures and Public Key Cryptosystems."*Comm. ACM* **21**, 120-126, 1978.

[2]鍾慶豐，2002，近代網路安全與編碼機制，頁 6-3~頁 6-76.

[3]陳彥學，2000，資訊安全理論與實務，頁 2-44~頁 2-47.

[4]張博峻，2004，資訊安全管理實務，頁 8-28~頁 8-36.

[5]王百輝，2003，Implementation of Secure FTP Servers Using OpenSSL，國立高雄第一科技大學電腦與科技工程學系碩士論文.

[6] 朱天元，A Research on A New Electronic Payment Mechanism with Its Security，長庚大學企業管理研究所碩士論文.

[7] 楊重駿，楊照崑，2002，數論在密碼上的應用，數學傳播第七卷第二期、第七卷第三期.

中英文網站:

[8] 政府憑證管理中心(Government Certification Authority)
http://www.pki.gov.tw/

[9] 內政部憑證管理中心(Ministry of the Interior Certification Authority)
http://moica.nat.gov.tw/

[10] RSA Security, Inc.
http://www.rsasecurity.com/

[11] Microsoft MSDN Home
http://msdn.microsoft.com/

[12] Microsoft Platform SDK
http://www.microsoft.com/msdownload/platformsdk/sdkupdate/

[13] Netscape Communications Corporation, “Introduction to SSL,”
http://developer.netscape.com/docs/manuals/security/sslin/index.htm.

[14] M-System DiskOnKey Software Release Notes, DOK SDK Version 3.1.2.6

http://www.diskonkey.com/