Live-CA：結合Java Card與PKI的憑證管理系統之設計與實現

蔡育儒
高雄師範大學資訊教育研究所
tasis@wtps.ptc.edu.tw

楊中皇
高雄師範大學資訊教育研究所
chyang @computer.org

摘要

網際網路的應用例如電子文件和電子商務不斷的普及,已經逐漸改變人們生活方式和交易型態，因此開放式網路環境中的安全機制也更顯其重要性。除了對資料傳輸的保密外，網路使用者的身分認證更是重要的議題。公開金鑰基礎建設(Public Key Infrastructure, PKI)是目前可提供一個可信賴及安全基礎的網路環境中最成熟及最有效的解決方案，本研究採用開放原始碼 OpenCA 建置一個符合 PKI 架構的二層 CA 憑證管理中心並提供 LDAP 目錄服務。研究中亦開發出 PKI 使用者介面，避免使用者需不斷的切換頁面來進行憑證的申請、註銷及查詢的作業，同時結合 OpenCA 自動安裝程式、OpenCA Live CD 和 Java Card 的應用，讓使用者能快速的安裝及使用更安全的 PKI 網路環境。

關鍵詞：PKI, OpenCA,憑證,憑證機構,數位簽章.

Abstract

With the wide application of Internet, it has gradually changed the life styles and business models. Therefore, increasing importance has been attached to the security problem of the public network. When users access the Internet application services, they expect to have services with integrity, confidentiality and authentication. PKI (Public Key Infrastructure) is a mature and consummate formula for establishing a trusted and secure network at present. In this research, we use the open source software: OpenCA which realizes the technology of PKI to establish a two layer management of the Certification Authority (CA) that provides LDAP directories service. We also develop the PKI user interface with JavaCard, OpenCA installation program, and OpenCA Live CD to establish the development of PKI much more quickly within a more secure network.

Keyword: PKI, OpenCA, Certificate, Certification Authority, Digital Signature.

1. 前言

在資訊技術與網際網路迅速發展的今日，許多使用者已經能透過網際網路取得分享的資源與服務。網路上的應用，像是電子商務、電子銀行及電子化政府等等，正倚賴著網路的連接性而蓬勃發展的同時，隨之而來的網路安全和身分認證問題也正逐漸受到重視。一般網路常見的攻擊有中斷、竊取、竄改、偽造等方式，因此提供使用者一個保密性(Confidentiality)、確認性(Authentication)、完整性(Data Integrity)、不可否認性(Non-repudiation)、存取控制(Access Control)和可用性(Availability)等安全的網路環境[1]，來確保網路應用的安全及可靠性，是目前最重要的議題。

PKI(Public Key Infrastructure)[7]公開金鑰基礎建設結合公開金鑰(public key)及憑證(certificate)的功能，在電子訊息傳遞與交換過程中，可確保訊息的保密性、確認性、完整性、及不可否認性等四種重要的安全保證。透過PKI架構中的憑證機構(Certification Authority)的認證制度，可確認通訊雙方的真實身分，提供使用者一個安全可信賴的網路環境，所以世界各國及企業正在大力推行公開金鑰基礎建設(PKI)的建置[4]。

我國也在民國八十六年委託中華電信股份有限公司規劃、建置及營運「政府憑證管理中心(Government Certification Authority, GCA)」提供電子認證功能以利人民使用自然人憑證透過網路報稅及電子化政府之各項應用[5]。

Java Card[9]是利用 Java 程式語言撰寫應用程式 (applet) 的智慧卡，在網路上的付款安全，網路連接的安全與電子簽章等應用扮演重要角色，近年來政府所倡導的自然人憑證，即為 PKI 結合 Java Card 的一種應用。

由於目前商用的PKI軟體價格昂貴而且建置較為繁雜，均限制PKI網路環境的發展。因此本研究採用開放原始碼的OpenCA套件[22]為基礎，建置一個符合PKI架構的憑證管理中心系統，並使用shell script撰寫OpenCA自動安裝程式讓使用者能快速的安裝及設定OpenCA平台環境，再將OpenCA平台環境客製成Live CD和開發一個結合Java Card的PKI使用者介面，讓使用者可以簡易且方便的使用更安全的PKI網路環境。

2. 文獻探討

本章節就建置 PKI 憑證管理系統所需的理論及技術加以探討：其中包括相關的密碼學演算法、公開金鑰基礎建設、OpenCA 憑證系統、LDAP 目錄服務、Live CD 及 Java Card 等。

2.1 相關的密碼學演算法

密碼系統(crypotosystem)就是將原先的資料明文(plaintext)經由某些數學運算將其轉換成另一種偽裝形式，稱之為密文(ciphertext)。這種將明文轉成密文的過程稱為加密(encryption)，傳送者必須選擇一個參數用來轉換明文成為不同的密文，而這個參數稱為加密金鑰(encryption key)。加密過後的資料不同於原始資料的地方在於它無法被輕易地理解，故可以用來確保電子訊息的保密性，但是已經不能滿足目前網路應用的需求，因此發展出其他能提供完整性、確認性、及不可否認性的密碼技術，也可整合到 PKI 系統中。茲介紹如下：

2.1.1 私密金鑰密碼系統

私密金鑰密碼系統(Secret-key cryptosystem)又稱對稱式密碼系統(Symmetric cryptosystem)、在此系統中通訊的雙方擁有相同的一把加密金鑰，用來加密與解密資料。其優點是執行速度較快，缺點則是加密金鑰傳遞時的安全問題，因為一旦加密金鑰被截取，加密的功能便喪失了。目前常見的私密金鑰密碼系統有 DES(Data Encryption Standard)、Triple DES[17]、AES(Advanced Encryption Standard)[20]。

2.1.2 公開金鑰密碼系統

公開金鑰密碼系統(Public-key cryptosystem)又稱非對稱式密碼系統(Asymmetric cryptosystem)，與對稱式加解密技術的差別在於加密與解密時是使用不同的金鑰。最早提出此觀念的是 Diffie 和 Hellman，在 1976 年所提出[8]，在此系統中通訊的雙方各自擁有一對金鑰，一把是公鑰(public key)可對外公開的，另一把是私鑰(private key)自己保存並不對外公開;其中一把若做為加密用，另一把則做為解密用。其優點是沒有金鑰傳遞的安全問題，此外可提供數位簽章功能來確認發送者的身分，缺點則是執行速度太慢，以及公鑰認證的問題，目前常見的公開金鑰密碼系統有 RSA[13]、橢圓曲線(Elliptic Curve)[10]。

2.1.3 單向雜湊函數

單向雜湊函數(One-Way Hash Function)可以將一個不定長度的訊息 M，經運算後產生一個固定長度的訊息雜湊值，有時也被稱為訊息摘要(message digest)。訊息經單向雜湊函數轉換成一個雜湊值後，就無法從這一雜湊值還原得到原來的訊息，避免訊息在傳遞的過程中遭到竄改，這就是單向的意義。單向雜湊函數最常運用在公開金鑰數位簽章，提供訊息的防偽和不可否認性的功能。目前常見的雜湊函數有 MD5[12]和 SHA[18]。

2.1.4 數位簽章

藉由公開金鑰密碼系統所發展出來最重要的成果就是數位簽章(Digital Signature)[19]，數位簽章乃是對訊息的內容先進行湊雜演算法產生一個訊息摘要，要傳送訊息的人用其私鑰將此訊息摘要進行加密轉換成為一個數位簽章。收訊者收到數位簽章和訊息後，利用傳訊者的公開金鑰將數位簽章解密得到原來的訊息摘要並進行比對，若兩者的雜湊值相符則代表是傳訊者本人的簽章而且訊息內容並未被竄改。目前常見的數位簽章有數位簽章演算法(Digital Signature Algorithm)、RSA 數位簽章演算法、和橢圓曲線數位簽章演算法(ECDSA)。

2.2 公開金鑰基礎建設

公開金鑰基礎建設(Public Key Infrastructure)是使用公開金鑰密碼技術來保護網路傳送訊息的系統，由憑證機構(Certification Authority)、註冊中心(Registration Authority)、憑證/憑證廢止清冊資料庫(Certificate/CRL Repository)及使用者(End Entity)等架構所組成如圖 1，並制訂一個安全政策(A Security policy)藉此表明在安全性及公證性上的信賴度[15]。在此系統中結合憑證、數位簽章及加密的概念，將可以提供訊息的保密性、完整性、確認性及不可否認性等功能，並可確認通訊雙方的真實身分，進而大幅提升通訊安全。

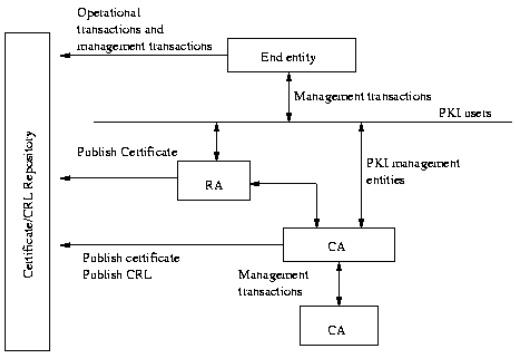

圖 1 公開金鑰基礎架構

2.2.1 憑證(certificate)

憑證的產生是由一個具有公信力的第三者(Trust Third Party)機構所簽發，也就是由憑證機構先將一個公開金鑰與使用者身分做確認後，對此公開金鑰和使用者相關資訊做數位簽章轉換成為一個憑證，並在網路上公開該憑證。為了讓憑證能在不同資訊系統中流通，必須訂定標準的憑證格式，一般憑證採用X.509 V3[11]作為標準，其憑證規範格式包含了版本、序號、簽章演算法、簽發單位名稱、憑證有效期限、使用者名稱、公開金鑰資料、簽發者識別號、使用者識別號、擴充欄位、數位簽章演算法、憑證簽章等內容。

2.2.2 憑證機構(Certification Authority)

憑證機構(CA)是具有公信力的第三者，也是PKI 架構中的核心組織，主要的工作有提供使用者認證的服務與憑證的簽發、憑證與公開金鑰的管理、發布憑證與憑證發止清冊以及處理與其他認證中心之間的信任關係。

2.2.3 註冊中心(Registration Authority)

註冊中心(RA)位於憑證機構的前端，負責個人及機關團體的身分確認。使用者在申請憑證前，必須先向註冊中心提出申請並且註冊。註冊中心確認申請者的身分及相關資料後，再交由憑證機構發行憑證。

2.2.4 目錄服務(Certificate/CRL Repository)

目錄服務(Certificate/CRL Repository)，是公開金鑰基礎架構中的資料儲存體，負責提供外界憑證目錄檢索查詢服務，例如憑證及憑證廢止清冊之公布，有效憑證及廢止憑證之查詢。相較於一般的資料庫系統，目錄服務主要工作是提供應用程式對資料庫內部資料快速的搜尋、讀取、瀏覽，而且符合X.500 目錄存取協定(Directory Access Protocol)。

2.2.5 憑證路徑(Certificate Path)

當有很多使用者欲申請憑證服務時，通常讓所有使用者都採用同一個憑證機構所簽發的憑證，並不是很實際的做法。在使用者的憑證為不同憑證機構所簽發的情況下，我們必須透過憑證路徑(certificate path)或憑證鏈(certificate chain)來驗證對方憑證的正確性與合法性[16]。圖 2 是最簡單的憑證路徑。憑證的信任關係是由使用者所信任的 Root CA 一直延伸到使用者的憑證，也就是 Root CA 對 Sub CA 簽發憑證，Sub CA 再對使用者簽發憑證。為了憑證管理上的需要，CA 與 CA 之間必須建立階層式架構的憑證路徑，一直到我們可以信賴的憑證中心。

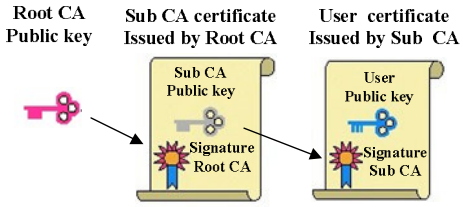

圖 2 憑證路徑

2.3 OpenCA 憑證系統

公開金鑰基礎建設(PKI)被視為目前可提供一個可信賴及安全基礎網路環境最成熟及最有效的解決方案，但是 PKI 系統的建置卻難以設定及安裝，而且必須花費高昂的成本，因此開放原始碼 Open CA PKI 發展計畫便開始發展。該計畫起源於 1999 年，希望能透過協同合作來發展一個堅固、全功能及開放原始碼的憑證系統，並可支援 OpenLDAP、OpenSSL、Apache 計畫及 Apache mod_ssl。

OpenCA PKI 發展計畫目前進行兩項主要工作：研究及強化安全架構確保最安全的憑證模型，及發展易於安裝及管理的憑證系統。

2.4 LDAP 目錄服務

X.500 雖定義出完整的目錄服務標準，但在實作上卻因整體過於龐大繁複而難以在現行的網路環境下實行。因此美國密西根大學根據 X.500 標準重新改良，提出一套可以在現行的 TCP/IP 架構上實作的版本，稱為輕量級目錄存取協定 LDAP (Lightweight Directory Access Protocol)新的版本為 LDAP V3 發表為 RFC 2551[14]，而 Open CA 憑證系統亦提供 LDAP 目錄服務。

2.5 Live-CD

一種不需安裝即可啟動的作業系統稱為 Live 系統，一般都是以 CD 片製作因此也稱為 LiveCD。雖然 Linux 作業系統為開放原始碼且免費使用，但其安裝設定上並不像 Windows 系統那麼方便，因此在自由軟體的前提下，有人將 Linux 作業系統改寫成 Live 系統，讓使用者省去安裝 Linux 的麻煩，而且無須硬碟就可以直接在光碟上執行完整的 Linux 作業系統。目前已有許多的 Live CD 釋出，較著名的如 KNOPPIX [3,21]、Mandrake Move、Fedora Live CD…等等。

2.6 Java Card

Java Card [6,9]是智慧卡的一種，包括了內嵌的中央處理器以及多種記憶體，有執行Java技術的Java Card虛擬機器(JCVM)及執行環境(JCRE)，並提供網路應用的確認性、保密性及便利性。同時Java Card 也提供許多具有物件導向程式設計特色的API，其中最重要的部份就是與密碼學有關的程式開發界面，3DES、RSA、SHA-1、MD5等，讓程式開發比一般智慧卡更容易，也使得Java Card 本身可以在卡片內部進資料的加解密及數位簽章等安全性功能。

3. 系統實作

3.1 系統架構

本研究的系統架構分為三部份，憑證管理系統、發卡系統和PKI用戶端，如圖3所示。憑證管理系統使用開放原始碼OpenCA套件，安裝在兩台Linux Fedora Core 4平台上，運用階層管理模式建置一個符合PKI架構的二層憑證管理中心，並且將OpenCA

憑證管理系統客製成Live CD。在發卡系統方面，我們則開發Terminal(主機端)與Applet(卡片程式端)程式，透過Applet程式，使用者可將憑證等相關資料經讀卡機存放於Java Card中。在PKI用戶端方面，我們使用Borland C++ Builder 6.0 開發出結合Java Card的憑證管理介面，提供使用者一個視窗管理介面來進行各項憑證系統作業，並可將私鑰及憑證存放於Java Card中，加強系統的安全性。

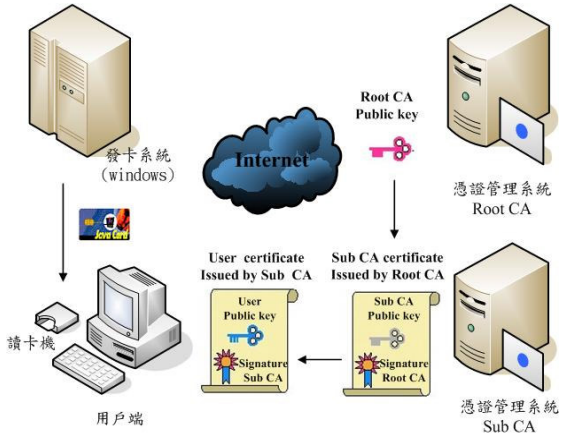

圖 3 系統架構

3.2 OpenCA 憑證系統的安裝

OpenCA 套件並不是一個獨立的完整系統，在操作過程中會使用到其他許多由開放原始碼社群所發展的軟體，所以在安裝 OpenCA 之前需先安裝以下軟體 OpenSSL、Apache、Perl5、mod-ssl、MySQL 及 OpenLDAP 等套件並依其建議版本做安裝[2]。因此 OpenCA 的安裝過程相當複雜、為了簡化 Open CA 的安裝，我們使用 Shell script 程式語言寫出一組安裝程式，提供使用者可執行安裝程式，來自動化安裝、設定及啟動 OpenCA 和所需相關的套件。執行該程式可進入 OpeCA 全自動編譯、安裝畫面，也可以選擇個別安裝 OpenSSL、Apache、Perl5、mod-ssl、MySQL 及 OpenLDAP 等套件，如圖 4。

圖 4 Open CA 自動安裝程式

3.3 建置二層憑證管理中心

在 PKI 的架構中，為了方便管理憑證，我們往往需要配置多層的 CA，本研究的 PKI 架構是一個二層 CA 的架構，使用者可根據組織的需求自行建立 CA 組織架構的深度與廣度。但是 OpenCA 對二層 CA 的建置並不夠完善，需要靠手動和修改檔案的方式，才能完成二層 CA 的配置方式。以下是二層 CA 的配置方式：

3.3.1 Root CA 實作

先準備一台伺服器進行 OpenCA 的安裝，CA 的安裝目錄為(/usr/local/openca/)，RA 的安裝目錄為(/usr/local/openra/)，接著分別修改 CA 和 RA 目錄中的 config.xml 檔案，在 Dataexchange configuration 選項中，將預設 CA 和 RA 的資料交換位置由軟碟(/dev/fd0)更改到(/usr/local/openca/OpenCA/var/tmp/)目錄中，便於匯出和匯入憑證的交換。

1. CA 的初始化：

使用瀏覽器進入 CA 的管理頁面(https://<Root CA URL>/ca/)進行憑證中心的初始化分三個步驟。

Phase Ⅰ CA 初始化：主要是完成 CA 自我簽發的憑證 (Root CA)，如圖 5 所示。CA 憑證初始化依序由 Initialize Database 執行到 Self Signed CA Certificate (from altready generated request) 為止，此時完成了 Root CA 自我簽發的憑證動作，最後點選 Rebuild CA Chain ，完成 CA 憑證初始化。

圖 5 CA 憑證初始化

Phase Ⅱ 建立 CA 管理者：作用是產生 CA 管理者的憑證，注意申請憑證時，將 Level 設為 High，Role 設為 CA Operator，最後將 CA 管理者的憑證格式設為 PEM 下載保存。

Phase Ⅲ 建立 RA 管理者：作用是產生 RA 管理者的憑證，將 Level 設為 High，Role 設為 RA Operator，最後將 RA 管理者的憑證格式設為 PEM 下載保存。

2. RA 的初始化：

使用瀏覽器進入 RA 的管理頁面(https://<Root CA URL>/ra/)，點選 Server Management 進入 RA 節點的管理頁面，點選 Administration 選項中的 Dataexchange，和 CA 進行所有的資料交換，完成 RA 的初始化。現在的 Open CA 憑證系統已經可以為使用者提供基本的 PKI 服務，使用者使用瀏覽器進入使用者頁面(https://<Root CA URL>/ pub/)，即可申辦憑證申請服務。

3.3.2 Sub CA 實作

再準備另一台伺服器安裝 OpenCA，此伺服器的建置同 Root CA。進入 CA 的頁面 https://<Sub CA

URL>/ca/)準備進行 Sub CA 的系統初始化，如圖 5，依序由 Initialize Database 執行到 Generate new CA Certificate Request (use generated secret key)，此時將產生 Sub CA 系統的憑證請求，位於 Sub CA 伺服器中的(/usr/local/openca/OpenCA/var/ crypto/reqs/ careq.pem)文件中。用手動的方式將 careq.pem 下載。

使用瀏覽器進入 Root CA 的使用者頁面(https://<Root CA URL>/pub/)，點選 User 選項中的 Request a Certificate。選擇 server request 的憑證請求模式。在 server request 的模式中會要求使用者匯入 PEM 格式的憑證，並產生該憑證的請求。進入 Root CA 的 CA 管理頁面(https://<Root CA URL>/ca/)簽發 Sub CA 的憑證請求。

進入使用者頁面(https://<Root CA URL>/pub/)將已簽發的 Sub CA 憑證下載儲存為 PEM 格式。從 -----BEGIN CERTIFICATE----- 開始，反白複製到 -----END CERTIFICATE----- 為止，並且用文書編輯器另存為 cacert.pem 的檔案名稱。

將從 Root CA 下載的 cacert.pem 傳送到 Sub CA 伺服器 /tmp 的目錄中執行以下命令：tar –cvf /usr/local/openca/OpenCA/var/tmp/ca-local cacert.pem，將 cacert.pem 上傳到 Sub CA 伺服器中的 CA server。進入 Sub CA 的 CA 初始頁面，如圖 5。點選 Import CA certificate (approved by Root CA) 將 Root CA 簽發的 Sub CA 憑證匯入到 CA server 中。最後點選 Rebuild CA Chain 完成 Sub CA 的憑證初始化。同樣地進行 CA 管理者憑證、RA 管理者的憑證和 RA 的初始化，同 Root CA 的程序，至此完成 Sub CA 系統建置。

3.4 OpenCA Live CD 客製化

Linux 算是最早發展出 Live CD 的作業系統，其中功能最完整的就屬 KNOPPIX Linux。KNOPPIX 是由德國程式設計師克勞斯(Klaus KNOPPER)所釋出的 Live 系統，是以 Linux 的 Debian 套件為主幹，因此本研究以 KNOPPIX Live CD 作為基礎，在 KNOPPIX 系統中安裝 OpenCA，並加入開機執行檔，將其重製為 OpenCA Live CD，使用者使用光碟開機後，免安裝、免設定即可架設一台 OpenCA 憑證管理中心伺服器。開機畫面如圖 6。

圖 6 OpenCA Live CD 啟動畫面

3.5 PKI 用戶端

本研究使用 Borland C++ Builder 6.0 來開發 PKI 用戶端的使用者介面，使得憑證的申請及發布更為簡便，軟體的主畫面如圖 7 所示，採用單一的視窗介面來進行各項憑證系統作業，同時結合 Java Card 的應用，提供使用者簡便及安全的操作介面來整合憑證管理系統功能。

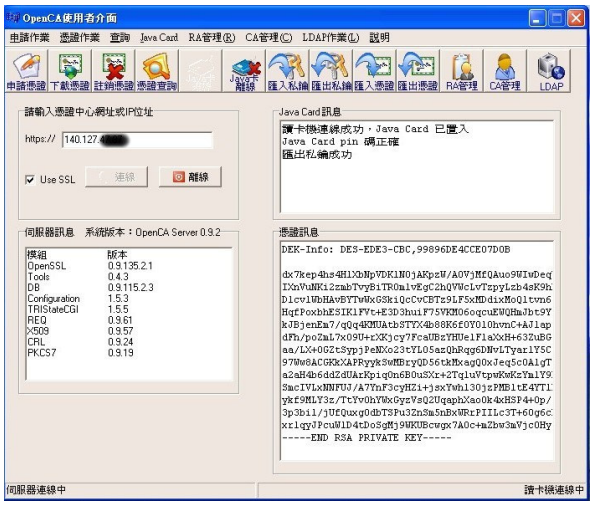

圖 7 PKI 使用者介面

3.5.1 操作說明

使用者先在欄位中輸入憑證中心的IP位址或DNS與憑證中心連線確認後，便會在訊息畫面中顯示出目前伺服器所使用的相關軟體套件及版本訊息，操作介面也支援SSL通訊安全架構，提供使用者與伺服器建立SSL安全連線，進行傳輸資料的加密。

確認連線後，使用者便可進行憑證的申請如圖 8 或憑證的註銷，使用者需在申請表格中填入正確的資料，按下送出便可以將申請資料上傳至憑證管理中心的前端註冊中心，待 RA 管理者確認申請者的身分及相關資料後，再交由憑證機構的 CA 管理者進行憑證的簽發及註銷。

使用者的申請資料經憑證管理中心核可後，便可以進行憑證的下載，使用者先將憑證及個人私鑰下載至個人的資料中，再利用匯出憑證的功能將資料儲存於 Java Card。

最後使用者也可利用憑證查詢的功能如圖 9 所示，來進行憑證目錄檢索查詢服務，例如有效憑證、過期的憑證、已廢止憑證及已註銷的憑證之查詢。

圖 8 申請憑證

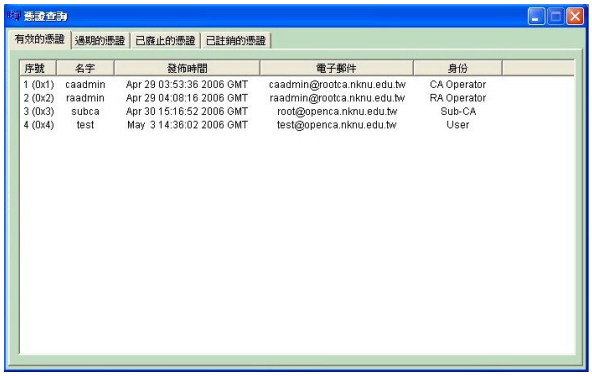

圖 9 憑證查詢

4. 結語

本文研究主旨是在開放式網路環境下，提出利用開放原始碼 OpenCA 實作和設計出簡易、安全及符合 PKI 架構的 OpenCA 憑證管理系統。由於 OpenCA 套件在 Linux 系統安裝上較為繁雜，本研究提出 OpenCA 自動安裝程式和 OpenCA Live CD，讓使用者省去安裝和設定上的麻煩。在憑證管理系統下，使用者需不斷切換頁面來進行憑證的申請、註銷及查詢，對於憑證存取控制仍有不便之處，因此本研究開發出 PKI 使用者介面，同時結合 Java Card 的應用來整合憑證管理系統功能，讓使用者的憑證作業能簡化，也讓 OpenCA 憑證管理系統易於使用，以提供使用者更完整、更安全、更有效率的公開金鑰基礎架構之網路環境。

參考文獻

[1]楊中皇，網路安全理論與實務，2006，台北：金禾資訊。

[2]鄭佩技，Live-CA：結合 IC 卡的 PKI 憑證管理系統之設計與實現，國立高雄師範大學資訊教育研究所碩士論文，2005。

[3]KNOPPIX 中文交流網, http://kxoppix.tnc.edu.tw/

[4]亞洲公開金鑰基礎建設論壇, http://www.pki.org.tw

[5]政府憑證管理中心，http://www.pki.gov.tw/

[6] M. Baentsch , "JavaCard—From Hype to Reality" , IEEE Concurrency, pp.36-42, October 1999.

[7] W. Burr , "Public Key Infrastructure (PKI) Technical Specifications: Part A –Technical Concept of Operations" , September 1998. http://csrc.nist.gov/pki/twg/baseline/pkicon20b.PDF

[8] W. Diffie and M. Hellman, "New Directions in Cryptography" ,IEEE Transactions on Information Theory 22,pp.644-654,1976.

[9] C. Enrique Ortiz, "An Introduction to Java Card Technology ", May 2003. http://java.sun.com/products/javacard/

[10] A. Menezes and S.Vanstone , "Elliptic curve cryptosystems and their implementation" , Journal of Cryptology, Vol. 6 , pp.209-224, 1993 .

[11] R.Housley，W.Ford，W. Polk and D. Solo "Internet Public Key Infrastructure Part I: X.509 Certificate and CRL Profile "，March 1996. http://www.faqs.org/ftp/rfc/pdf/rfc2510.txt.pdf

[12] R. Rivest ,The MD5 Message-Digest Algorithm, April 1992. http://www.faqs.org/ftp/rfc/pdf/rfc1321.txt.pdf

[13] R. Rivest, A. Shamir, and L. Adleman,"A Method for Obtaining Digital Signatures and Public- Key Cryptosystems ", Communications of ACM, Vol. 21, pp.120-126 , February 1978.

[14] M. Wahl, T. Howes, and S. Kille, Lightweight Directory Access Protocol (v3), December 1997. http://www.ietf.org/rfc/rfc2251.txt

[15] S.Xenitellis, "The Opensouce PKI Book,OpenCA Team ",2000. http://ospkibook.sourceforge.net/docs/OSPKI-2.4.7/OSPKI.pdf

[16] William T.Polk and Nelson E. Hastings ,"Public Key Infrastructures that Satisfy Security Goals" , Internet Computing IEEE , Vol.7, pp.60-76, August 2003.

[17] NIST FIBS 46-3 ,Data Encryption Standard (DES) October 1999. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[18] NIST, FIBS 180-2,Secure Hash Standard, August 2002. http://csrc.nist.gov/publications/ fips/fips180-2/ fips180-2withchangenotice.pdf

[19] NIST, FIBS 186-2, Digital Signature Standard (DSS), January 2000. http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf.

[20] NIST, FIBS 197, Advanced Encryption Standard (AES), November 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[21]KNOPPIX, http://www.knoppix.org/

[22] OpenCA Labs, http://www.openca.org/