

# 結合智慧卡的 ECDSA 數位簽章軟體設計與實現

張惟淙

國立高雄師範大學資訊教育研究所

[bug@icemail.nknu.edu.tw](mailto:bug@icemail.nknu.edu.tw)

楊中皇

國立高雄師範大學資訊教育研究所

[chyang@nknucc.nknu.edu.tw](mailto:chyang@nknucc.nknu.edu.tw)

---

## 摘要

數位簽章是用途非常廣泛的資訊安全技術，尤其在電子商務應用上，是不可或缺的交易安全保護要件。目前具備數位簽章功能的軟體，如網際網路瀏覽器或電子郵件收發軟體，大多都採用 RSA 演算法。近年來，橢圓曲線密碼系統(Elliptic Curve Cryptosystem, ECC)，已經開始挑戰 RSA。就相同的安全性而言，ECC 所需要的密碼學金鑰長度較 RSA 短，而且有更佳的執行效率，這個特性使得 ECC 相當適合用於智慧卡、手機或其他無線行動裝置。

本研究即在發展一個結合智慧卡應用的橢圓曲線數位簽章軟體，我們使用 Java Card 作為 ECC 金鑰存取的媒介，並以此軟體產生及檢驗 ECDSA 數位簽章(Elliptic Curve Digital Signature Algorithm, ECDSA)，簽章產生過程中用以計算訊息摘要的雜湊函數演算法則採用 SHA-2 演算法。使用者可將持有的智慧卡視為鑰匙環(Keyring)，用以存放本身的公開金鑰(Public key)與秘密金鑰(Private key)，以及通訊對方的公開金鑰，金鑰經由卡片存取能提升使用上的便利與安全。使用者藉此與他人進行安全通訊時，得以確認對方身份及所傳送訊息的完整性。

**關鍵字：**密碼學、橢圓曲線、數位簽章、智慧卡、ECDSA

---

## 壹、前言

數位簽章是公開金鑰演算法最重要的應用，普遍用於維護電子商務交易安全。現今大部分具備數位簽章功能的軟體，如網路瀏覽器或電子郵件軟體，幾乎

都是採用 RSA 作為數位簽章演算法。近年來，為了保持 RSA 的安全性，金鑰的位元長度逐漸有增加的需求，亦因此加重使用 RSA 演算法的大量安全交易平台之處理負擔。

西元 1985 年時，美國華盛頓大學的 Neal Koblitz 教授(Koblitz, 1986)及 IBM 的 Victor Miller(Miller, 1987)分別提出以橢圓曲線演算法設計公開金鑰演算法的密碼技術，此後更發展出許多橢圓曲線密碼系統（Elliptic Curve Cryptosystem，ECC）的國際標準，如 ISO 11770-3、ANSI X9.62、IEEE P1363、FIPS 186-2 等。就相同的安全性而言，ECC 所需要的密碼學金鑰長度較 RSA 短，如表一所示(NIST, 2003)。所以無論從增快執行速度或節省空間的角度來看，可見 ECC 是優於 RSA。

表一 相同安全性時，RSA 與 ECC 金鑰長度比較(NIST, 2003)

演算法 \ 安全性	$2^{80}$	$2^{112}$	$2^{128}$	$2^{192}$	$2^{256}$
RSA 金鑰長度(bits)	1024	2048	3072	7680	15360
ECC 金鑰長度(bits)	160	224	256	384	512
金鑰長度比	6:1	9:1	12:1	20:1	30:1

本研究目的在於發展一個結合智慧卡的 ECDSA 數位簽章軟體，我們使用 Java Card 作為 ECC 金鑰存取媒介，並透過軟體產生 ECDSA 數位簽章，其中產生與檢驗簽章過程會用到的雜湊函數演算法，則採用 SHA-2 演算法。使用者可以將持有的 Java Card 視為鑰匙環，其中除存放本身的公開金鑰與秘密金鑰外，還可存放通訊對方的公開金鑰，藉此與他人進行安全通訊時，用以確認其身份及所傳送訊息的完整性。

## 貳、文獻探討

### 一、橢圓曲線密碼學(Elliptic Curve Cryptography)

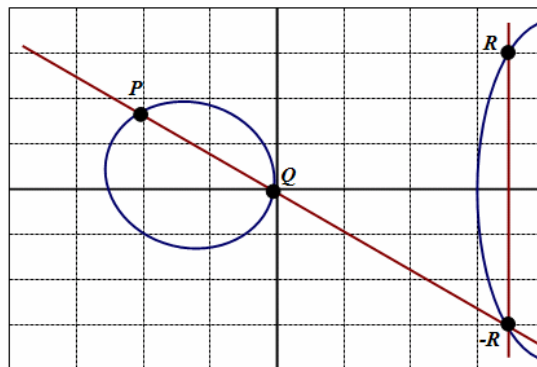
#### (一)、橢圓曲線密碼學原理簡介

橢圓曲線的密碼學技術不僅可以用於數位簽章、金鑰交換及加解密，還可應用在大數分解(Factorization)與質數判斷(Primality testing)。橢圓曲線的通用方程式如下：

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

密碼學的橢圓曲線是由滿足該方程式的所有點 $(x, y)$ 及一個無限遠點(Point at infinity) $O$ 所形成的集合，座標 $x$ 與 $y$ 屬於某個有限體(finite field)。目前軟體具體實現的有限體為質數體(Prime field,  $GF(p)$ )、二元體(Binary field,  $GF(2^n)$ )、最佳擴展體(Optimal extension field,  $GF(p^n)$ )等三種(Bailey and Paar, 2001)。

橢圓曲線上的點可進行兩點間之加法(Menezes, 1993; Silverman and Tate, 1992)。幾何上，如果要計算相異兩點 $P$ 與 $Q$ 的和，則先找出通過這兩點的直線，然後找出這條直線與橢圓曲線相交的第三點 $(-R)$ ，再將此點對 $x$ 軸做鏡射得到和 $(R)$ ，如圖一所示。如果橢圓曲線上的某兩點共線的話，兩點相加之和就是 $O$ 。



圖一 橢圓曲線幾何圖表示兩點加法

若 $P = (x_1, y_1)$ 與 $Q = (x_2, y_2)$ 為橢圓曲線上的任意兩點，而 $P \neq O \neq Q$ ，且選取質數體，此時橢圓曲線方程式為：

$$y^2 = x^3 + ax + b \tag{2}$$

兩點加法的運算規則如下所示：

1.  $P + O = O + P = P$
2.  $P + (-P) = (x_1, y_1) + (x_1, -y_1) = O$
3.  $P + Q = R = (x_3, y_3)$ ,

$$x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1, \quad \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases} \tag{3}$$

如果選擇二元體，則橢圓曲線方程式為：

$$y^2 + xy = x^3 + ax^2 + b \quad (4)$$

而上述公式(3)的加法規則 3 必須改為：

$$P + Q = (x_3, y_3),$$

$$x_3 = \begin{cases} \lambda^2 + \lambda + x_1 + x_2 + a & \text{if } P \neq Q \\ \lambda^2 + \lambda + a & \text{if } P = Q \end{cases}, \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1,$$

$$\lambda = \begin{cases} \frac{y_2 + y_1}{x_2 + x_1} & \text{if } P \neq Q \\ x_1 + \frac{x_1}{y_1} & \text{if } P = Q \end{cases} \quad (5)$$

公式(3)與(5)的計算(加法、減法、乘法、除法/反元素)必須在相關的有限體進行，若選取質數體時僅需進行模算術(Modular arithmetic)，若選取二元體則需進行多項式算術(Polynomial arithmetic)。點乘法計算  $k \cdot P$  為橢圓曲線密碼系統的基礎，其中  $k$  為正整數，而  $P$  為橢圓曲線上的一個點：

$$k \cdot P = \overbrace{P + P + \dots + P}^{k \text{ 個點相加}} \quad (6)$$

如果  $n \cdot P = O$  則  $n$  為點  $P$  的級數(order)。在合適的橢圓曲線上，可以找到一個級數  $n > 2^{160}$  的基點(Base point)  $G$ ，而此橢圓曲線系統參數基點  $G$  可公開；另隨機選取小於  $n$  的正整數  $d$  當作私密金鑰，計算  $Q = d \cdot G$  為對應的公開金鑰。點乘法的計算如果直接做  $k$  個點相加，則需要執行  $k-1$  次加法運算，效率不佳，目前已有許多可以加速點乘法計算的演算法。本研究所開發的橢圓曲線密碼模組則是採用 NAF(Non-Adjacent Form)演算法。

橢圓曲線密碼系統的實現必須考慮下列因素：

1. 有限體的選擇
2. 橢圓曲線的挑選
3. 有限體元素的運算(加法、減法、乘法、除法/反元素)
4. 橢圓曲線點的運算(加法、減法、乘法)

美國 ANSI X9.62(ANSI, 1998)與 FIPS 186-2(NIST, 2001)標準中針對質數體與二元體建議不同長度的橢圓曲線與合適的基點，所以有關橢圓曲線的參數挑選可參考該標準。網際網路上，有許多關於橢圓曲線運算的開放原始碼軟體，如：LiDIA、PARI-GP、OpenSSL 或 Crypto++...等，可作為橢圓曲線密碼系統開發的參考。

## (二)、橢圓曲線數位簽章演算法(ECDSA)

在 ECDSA 相關標準 ANSI X9.62(ANSI, 1998)與 FIPS 186-2(NIST, 2001)中提及對於訊息  $m$  的數位簽章  $(r, s)$  產生步驟如下：

1. 挑選一亂數  $k$ ， $n-1 \geq k \geq 1$
2. 計算  $k \cdot G = (x_1, y_1)$  且  $r = x_1 \bmod n$ ，如果  $r = 0$ ，則回到步驟 1
3. 計算  $s = k^{-1} \{h(m) + dr\} \bmod n$
4. 如果  $s = 0$ ，則回到步驟 1

而檢驗 ECDSA 簽章是否正確的步驟如下：

1. 計算  $w = s^{-1} \bmod n$
2. 計算  $u_1 = h(M)w \bmod n$  與  $u_2 = rw \bmod n$
3. 計算  $u_1 \cdot G + u_2 \cdot Q = (x_0, y_0)$  與  $v = x_0 \bmod n$
4. 若且唯若  $v = r$ ，則簽章正確

ECDSA 簽章產生時至少須進行一次點乘法以及一些模算術，簽章檢驗時則進行兩次點乘法以及一些模算術。表二為 RSA 與 ECDSA 用於數位簽章的比較，從表二可見相同安全性時，兩種演算法所產生出來的簽章長度差異甚大，這將影響簽章傳遞的時間與儲存的空間。

表二 數位簽章演算法 RSA 與 ECDSA 的比較

演算法	RSA	ECDSA
簽章長度	<ul style="list-style-type: none"> <li>• 安全性 <math>2^{128}</math>：384 位元組</li> <li>• 安全性 <math>2^{192}</math>：960 位元組</li> <li>• 安全性 <math>2^{256}</math>：1920 位元組</li> </ul>	<ul style="list-style-type: none"> <li>• 安全性 <math>2^{128}</math>：64 位元組(質數體)</li> <li>• 安全性 <math>2^{192}</math>：96 位元組(質數體)</li> <li>• 安全性 <math>2^{256}</math>：132 位元組(質數體)</li> </ul>
安全基礎	大數分解	橢圓曲線離線對數
優點	歷史悠久，容易說明，亦可同時用以加解密。	速度快，簽章長度小。
缺點	速度慢，簽章長度較大。	理論不易理解，實現技術較複雜。

## 二、SHA-2 雜湊函數演算法

數位簽章演算法的一般做法，會對欲傳輸的訊息以雜湊函數計算出一組訊息摘要，再以秘密金鑰將其加密後產生數位簽章，並一併與原始訊息傳送給對方。接收方會將數位簽章以傳送方的公開金鑰解開取出訊息摘要，再將接收到的訊息以同樣的雜湊函數演算法計算出另一組訊息摘要，以進行比對，相同則表示資料未被竄改，如此則可驗證資料的完整性。因此雜湊函數演算法對於數位簽章本身的安全性至為重要。

SHA(Secure Hash Algorithm)演算法是由美國國家標準技術研究院(NIST)所發展的，並且在 1993 年成爲第 180 項聯邦處理標準(FIPS PUB 180)。目前最新的修訂版是 FIPS PUB 180-2，其中新增包含 SHA-256、SHA-384 及 SHA-512 等三種雜湊演算法，統稱爲 SHA-2 演算法(NIST, 2002)。目前 ECDSA 實作較常用的是 SHA-1 演算法，然而 NIST 已宣佈將於 2010 年後不再支持處理流程類似於 MD5 且有安全疑慮的 SHA-1(NIST, 2005)。因此我們選擇較新的 SHA-2 作爲產生 ECDSA 數位簽章時使用的雜湊函數演算法。

## 三、Java Card

Java Card 是一種標準的智慧卡，智慧卡在當今網路安全應用最重要的三個特點爲：確認性、保密性及便利性。其內部有源於 Java 技術的 Java Card 虛擬機器(Java Card Virtual Machine, JCVM)及 Java Card 執行環境(Java Card Runtime Environment, JCRE)。此外，Java Card 尚提供一套具有物件導向程式設計特色的 API (Ortiz, 2003)。本研究所開發的軟體中用以存取公開金鑰及秘密金鑰的 Java Card 是 IBM JCOP20 卡，它完全符合 Visa OpenPlatform Card 架構及 Java Card API 2.1.1 版的規範。

## 參、軟體實作

### 一、ECDSA 程式模組

我們使用自行開發的 ECDSA 數位簽章的簽章產生與檢驗程式模組，設計軟體的核心功能。圖二及圖三分別是橢圓曲線參數爲質數體的數位簽章產生模組及檢驗模組。此外還有設計橢圓曲線參數爲二元體的數位簽章相關模組，這些模組均已經使用 ANSI X9.62(ECDSA)標準文件中所提供的測試樣本檢驗無誤(楊中皇，2005)。

```

{
    EC_KEYPAIR      random_key;
    BIGINT          x_value, k_value, sig_value, r_value;
    BIGINT          temp, quotient;
    BIGINT          random_k;
    BIGINT          key_value, point_order, u_value;
    INDEX          i, count;

    copy (&ecdsa_k, &random_key.prvt_key);
    print_field ("Given random number k = ", &random_key.prvt_k
/* compute k * G */
    EC_mul( &random_key.prvt_key, &public_curve->pnt, &random_k

/* display public key */
    printf ("k G = \n");
    print_point("", &random_key.pblic_key);

/* compute r = k G(x) mod n */
    field_to_int( &public_curve->pnt_order, &point_order);
    field_to_int( &random_key.pblic_key.x, &x_value);
    int_div( &x_value, &point_order, &quotient, &r_value);
    int_to_field(&r_value, &signature->r);

/* compute s = k^-1 (e + d r) mod n */

```

圖二 ECDSA 數位簽章產生模組(部分)

```

{
    POINT          Temp1, Temp2, Verify;
    BIGINT          r_value, s_value;
    BIGINT          temp, quotient, h1, h2;
    BIGINT          check_value, point_order;
    INDEX          i, count;
    FIELD_P        h1_field, h2_field;

/* compute inverse of second signature value */
    field_to_int( &public_curve->pnt_order, &point_order);
    field_to_int( &signature->s, &temp);
    mod_inv( &temp, &point_order, &s_value);

/* compute elliptic curve multipliers: h1 = hash * s, h2 = c *
    int_mul( &hash_value, &s_value, &temp);
    int_div( &temp, &point_order, &quotient, &h1);
    int_to_field( &h1, &h1_field);
    field_to_int( &signature->r, &r_value);
    int_mul( &s_value, &r_value, &temp);
    int_div( &temp, &point_order, &quotient, &h2);
    int_to_field( &h2, &h2_field);

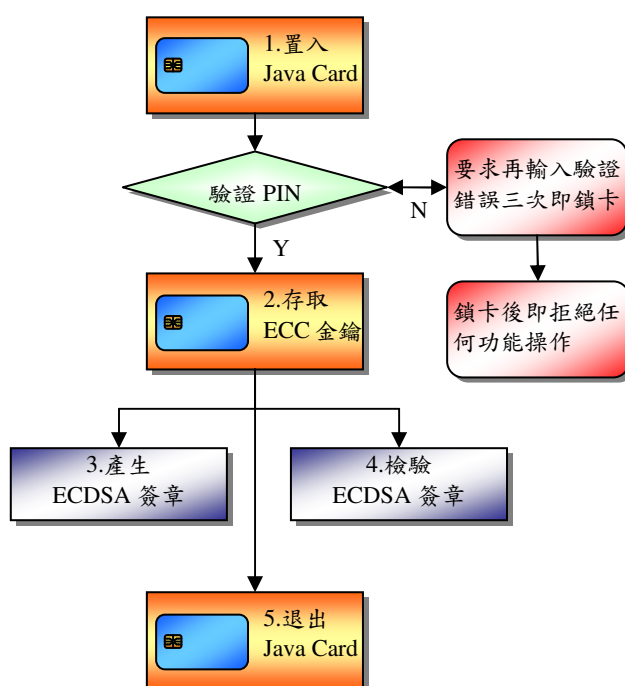
/* find hidden point from public data */
    EC_mul( &h1_field, &public_curve->pnt, &Temp1, &public_curve->crv);
    EC_mul( &h2_field, &signer_point, &Temp2, &public_curve->crv);
    EC_add( &Temp1, &Temp2, &Verify, &public_curve->crv);

```

圖三 ECDSA 數位簽章檢驗模組(部分)

## 二、應用流程設計

使用者必須先持有內建 ECC 金鑰存取程式的 Java Card，才能使用這套軟體。第一次使用的時候必須先建立自己的 ECC 金鑰對，爾後可以匯出自己的及匯入他人的公開金鑰，以便與他人進行通訊時使用。在執行任何有關數位簽章的動作之前，軟體本身就會檢查是否已經從卡片讀取到金鑰，金鑰是不允許使用者手動輸入的，如果沒有金鑰，軟體將不會執行數位簽章的產生與檢驗等相關功能。應用本軟體的流程如圖四：



圖四 ECDSA 數位簽章軟體的應用流程圖

操作的步驟說明如表三：

表三 ECDSA 數位簽章軟體的應用流程說明

步驟	說明
1	將內建有 ECC 金鑰存取程式的 Java Card 置入讀卡機。
2	讀取卡片之後，會要求使用者輸入 PIN 碼驗證，驗證成功後，才可繼續使用，驗證錯誤達三次即鎖卡。第一次使用應建立自己的 ECC 金鑰對，這個部分還提供匯出入公開金鑰的功能。本身的 ECC 金鑰亦可以更新置換，他人的公開金鑰則可更新或刪除，但置換金鑰會影響無法檢驗已建立的數位簽章。

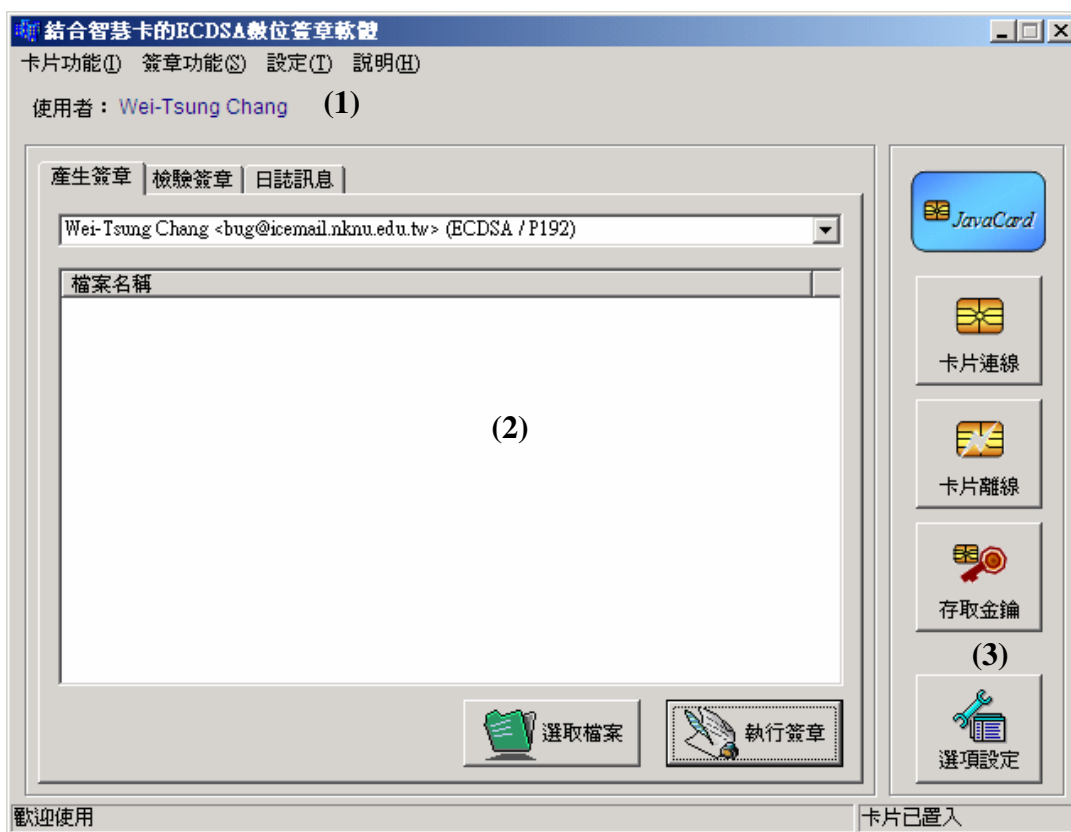


3	以 ECC 秘密金鑰對檔案產生數位簽章。可以同時對多個檔案進行批次產生簽章。
4	以自己或他人的 ECC 公開金鑰對數位簽章進行檢驗。可以同時對多個檔案進行批次檢驗簽章。
5	退出 Java Card，結束所有作業，並清除軟體執行時暫存金鑰內容的變數值。

### 三、軟體使用簡介

#### (一)、操作介面說明

我們使用 Borland C++ Builder 6 開發，它不僅能快速建立視窗介面，且能整合 C/C++ 原始碼。軟體主畫面如圖五所示：



圖五 軟體主畫面

主畫面大致劃分以下三個主要功能區塊說明：

- (1) 主選單及使用者名稱：主選單有卡片功能、簽章功能等兩個子選單，以及設定與說明功能。所有的功能亦可直接從畫面上的圖示按鈕點選執行，使用者名稱的欄位只有在卡片已連線並經驗證後，才會顯示。

- (2) 數位簽章功能區塊：有三個分頁功能，分別是數位簽章的產生、檢驗以及記錄相關處理過程的日誌訊息。
- (3) 卡片及設定工具鈕：與卡片存取有關的功能可以直接從這裡去點選執行，另外還有選項設定的功能，可以修改一些常用選項。此區塊最上面的卡片狀態圖如果顯示彩色表示卡片目前連線使用中，若是黑白圖片則表示尚未置入卡片。

## (二)、金鑰存取

金鑰的存取畫面(如圖六)，主要有以下功能，使用前必須先勾選相關的使用者名稱：

- (1) 讀取：讀出金鑰的內容，並儲存到軟體中的變數，以供後續使用。
- (2) 新增：新增 ECC 金鑰對。必須輸入使用者名稱及 E-Mail，另外還需選擇金鑰長度，我們參考 NIST 的 FIPS 186-2 標準，共提供五種質數體 (P-XXX)及十種二元體(B-XXX、K-XXX)的金鑰參數，如圖七所示。
- (3) 匯出：匯出自己的公開金鑰。
- (4) 匯入：匯入他人的公開金鑰。
- (5) 移除：移除勾選的使用者之金鑰。



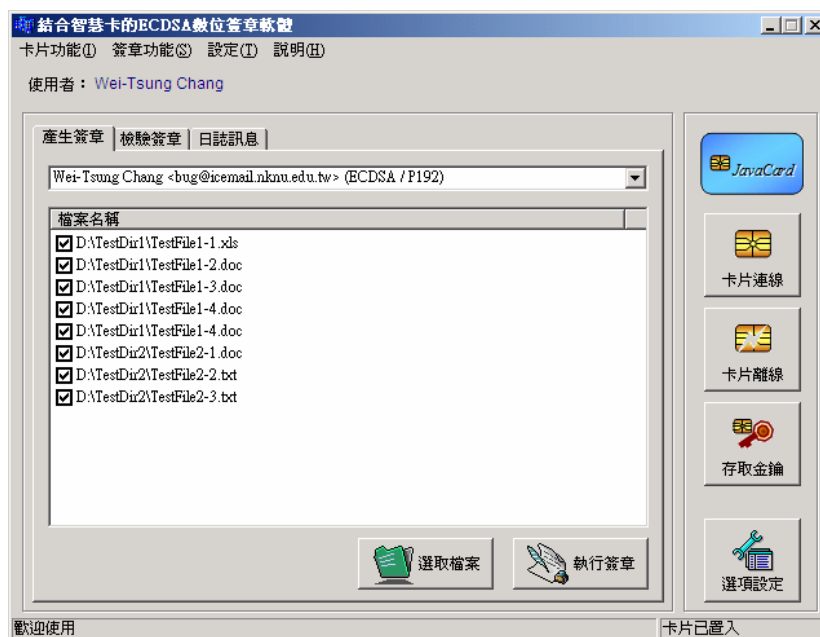
圖六 ECC 金鑰存取



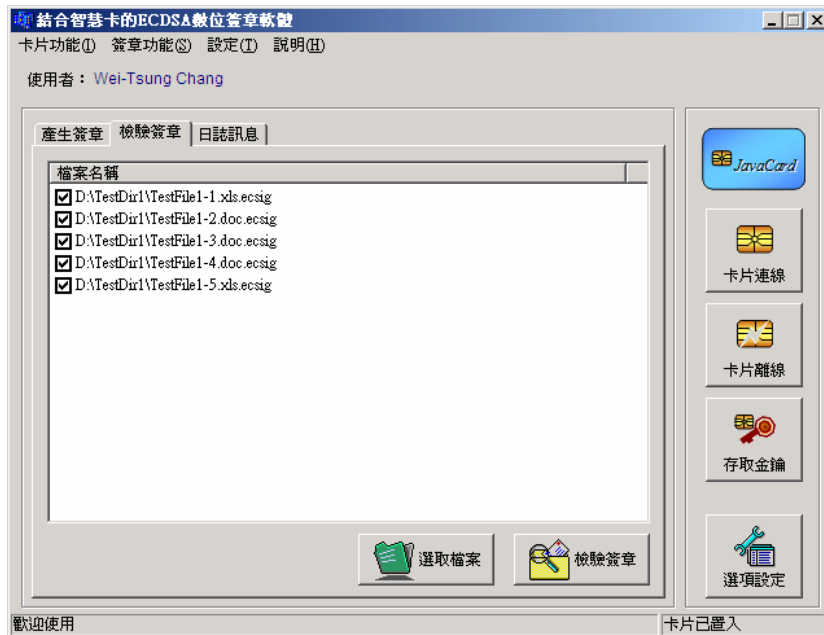
圖七 新增 ECC 金鑰對

### (三)、ECDSA 簽章的產生與檢驗

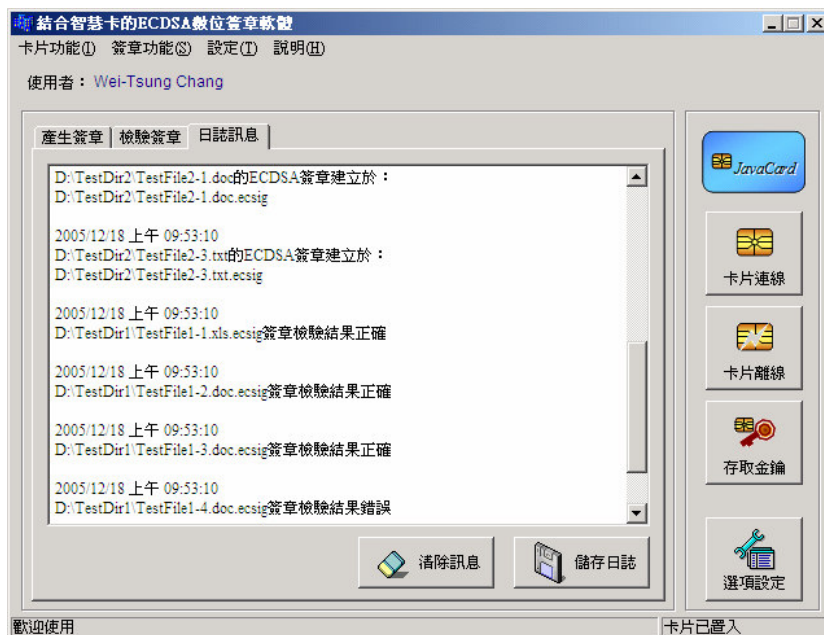
無論是產生或檢驗數位簽章，都可以選取多個檔案執行，準備要用以產生數位簽章的檔案，或是要檢驗的數位簽章檔，均會列表供使用者瀏覽，如果某個檔案不欲執行相關動作，只要勾選取消即可，橢圓曲線數位簽章的副檔名我們使用預設為".ecsig"，以便與坊間其他軟體所慣用的數位簽章檔名".sig"作區別。相關的範例畫面如圖八與圖九。產生與檢驗簽章的處理過程會顯示在主畫面第三個分頁「日誌訊息」欄內，會記錄處理的日期時間與結果，如圖十所示。



圖八 選取欲產生數位簽章的檔案



圖九 選取欲檢驗的數位簽章檔



圖十 檢視處理過程的訊息

#### 四、結論

雖然橢圓曲線密碼學早在 1985 年就有學者提出，然而迄今實務應用仍不普遍。本研究以 Java Card 為例，並使用自行設計的橢圓曲線密碼模組開發一個結合智慧卡且具有友善視窗使用介面的 ECDSA 數位簽章軟體，這不僅應證 ECC 的實務應用，同時還提供我們研究的思維與創新的契機。根據對 ECC 相關文獻

的探討，可知在相同的安全性之下，ECC 所需的金鑰長度遠比 RSA 金鑰來得小，不僅如此，ECC 還有更好的執行效率，故在可預見的未來，ECC 的技術勢必將逐漸成為公開金鑰密碼系統的主流，同時亦將為電子商務交易提供品質更好、效率更佳的安全保障。

## 伍、參考文獻

ANSI X9.62, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm(ECDSA) , 1998.

A. Menezes, Elliptic Curve Public Key Cryptosystems, Kluwer, 1993.

C.E. Ortiz, An Introduction to Java Card Technology, <http://developers.sun.com/techtopics/mobility/javacard/articles/javacard1/>, 2003.

D.V. Bailey and C. Paar, Efficient arithmetic in finite field extensions with applications in elliptic curve cryptograph, J. Cryptology, 2001, Vol. 14, No. 3, pp. 153-176.

IBM, JCOP20 Technical Brief, <http://www.zurich.ibm.com/jcop/download/specs/JCOP20Brief.pdf>.

J. H. Silverman and J. Tate, Rational Points on Elliptic Curves, 1992, Undergraduate Texts in Mathematics, Springer-Verlag.

LiDIA, <http://www.informatik.tu-darmstadt.de/TI/LiDIA/>.

N. Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation, 1987, Vol. 48, No. 177, pp. 203-209.

NIST, DRAFT Special Publication 800-57, Recommendation on Key Management, <http://csrc.nist.gov/CryptoToolkit/kms/guideline-1-Jan03.pdf>, 2003.

NIST, FIPS 186-2, Digital Signature Standard (DSS), <http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf>, 2001.

NIST, FIPS PUB 180-2, Secure Hash Standard, <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>, 2002.

NIST, NIST Brief Comments on Recent Cryptanalytic Attacks on SHA-1, [http://csrc.nist.gov/hash\\_standards\\_comments.pdf](http://csrc.nist.gov/hash_standards_comments.pdf), 2005.

PARI/GP, <http://pari.math.u-bordeaux.fr/>.

The OpenSSL Project, <http://www.openssl.org/>.

V. Miller, Use of Elliptic Curves in Cryptography, In Advances in Cryptology: Proceedings of Crypto '85, 1986, volume 218 of Lecture Notes in Computer Science, pp. 417-426, Berlin, Springer-Verlag.

Wei Dai, Crypto++ library, <http://www.eskimo.com/~weidai/cryptlib.html>.

楊中皇，橢圓曲線密碼系統軟體實現技術之探討，資訊安全通訊，2005 年 1 月，第 11 卷，第 2 期，頁 15-25。