橢圓曲線密碼系統軟體實現技術之探討

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

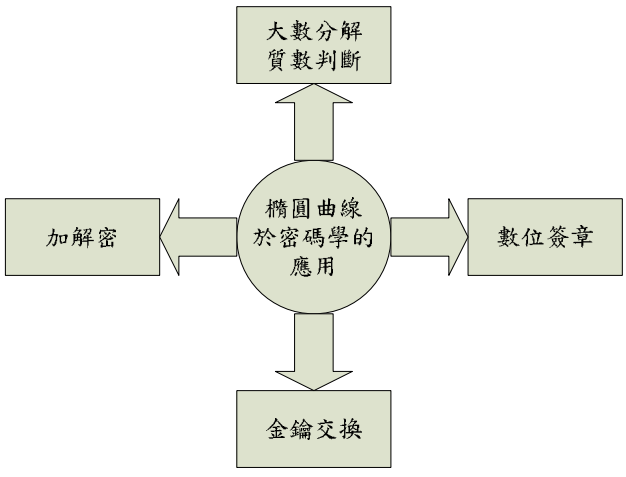
摘要

橢圓曲線密碼系統(ECC)近年來已被廣泛地制訂於國際標準。在相同的安全強度下，ECC 的密碼學參數可遠較諸如 RSA 的其他公開金鑰密碼系統為小，這使得 ECC 非常適合在例如智慧卡或行動裝置的有限資源環境下使用。但是 ECC 的實現較目前國內常用的 RSA 或 Diffie-Hellman 密碼系統複雜。本文以橢圓曲線數位簽章演算法 ECDSA 為例，探討如何以 Mathematica 工具及 C 語言快速開發與測試橢圓曲線密碼系統相關軟體。

關鍵詞：密碼學，橢圓曲線，數位簽章。

壹、前言

密碼學技術是目前所知唯一能有效地在不安全的網路上安全傳遞訊息的工具，而橢圓曲線密碼系統（elliptic curve cryptosystem， ECC）[8][12]則是在西元 1985 年由 Koblitz 與 Miler 各別提出的一種新的密碼學技術，且 ECC 近年來已被廣泛地制訂於國際標準如 ISO 11770-3、ANSI X9.62、IEEE P1363 、FIPS 186-2 等。橢圓曲線的技術不只可用在加解密、數位簽章、金鑰交換等，也可用於大數分解(factorization)與質數判斷(primality testing)，如圖一。在相同的安全強度下，ECC 的密碼學金鑰長度可遠較諸如 RSA 的其他密碼系統為小且速度較快，這使得 ECC 非常適合在例如智慧卡或無線行動裝置的有限資源環境下使用。



圖一：橢圓曲線於密碼學的應用

ECC 在具體實現方面有許多瓶頸，遠較目前國內常用的 RSA 或 Diffie-Hellman 複雜。但是也由於 ECC 與傳統的公開金鑰密碼系統(RSA 或 Diffie-Hellman 等)比較時具有執行效率上的優勢，所以目前 ECC 已廣被採納，甚至美國國防部的可當成智慧卡或 PKI/KMI 符記(token)[5]也採納 ECC。

在相同的安全強度下，ECC 的金鑰長度與 RSA 的金鑰長度比較 [10]，如下表所示。從表一中可見 ECC 的金鑰長度或數位簽章的長度遠比 RSA 小。隨著加解密演算法由 DES/Triple-DES 改進為 AES，且 128 位元金鑰到 256 位元金鑰 AES 的安全性為 2^{128} 至 2^{256}，相同安全性的 RSA 金鑰長度需 3072 位元到 15360 位元，但 ECC 僅需 256 位元到 512 位元。若 RSA 或 ECC 是用做金鑰交換來保護 256 位元金鑰的 AES 時，RSA 應該用 15360 位元的公開金鑰，而對應的 ECC 僅需使用 512 位元金鑰。所以無論從增快執行速度或節省空間的角度，我們可見 ECC 是優於 RSA。

表一：相同安全性時RSA與ECC金鑰長度比較[10]

安全性 / 演算法	2^{80}	2^{112}	2^{128}	2^{192}	2^{256}
RSA長度(位元)	1024	2048	3072	7680	15360
ECC長度(位元)	161	224	256	384	512
金鑰長度比	6:1	9:1	12:1	20:1	30:1

橢圓曲線的通用格式為 $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$。用於密碼學技術的橢圓曲線是由滿足上述方程式的所有點(x, y)及一個無限遠點(point at infinity) O所形成的集合，其中座標x與y屬於某個有限體(finite field)。目前軟硬體具體實現的橢圓曲線有限體為質數體(prime field， GF(p))、二元體(binary field ，GF(2^n))、最佳擴展體(optimal extension field ，GF (p^n))[2]等三種。

橢圓曲線上的點可進行兩點間之加法[8][12]。幾何上，如果要計算相異兩點P與Q的和，則我們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，再將此點對x軸做鏡射得到和。如果橢圓曲線上的某三點共線的話，三點相加之和就是O。

若$P = (x_1, y_1)$與$Q = (x_2, y_2)$為橢圓曲線上的任意兩點，但$P \neq O \neq Q$，且選取質數體（此時橢圓曲線方程式為 $y^2 = x^3 + ax + b$），則我們有下列兩點加法的運算規則：

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

$$x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda\ (x_1 - x_3) - y_1, \quad \lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases} \tag{1}$$

如果我們選擇二元體，那麼橢圓曲線方程式為 $y^2 + xy = x^3 + ax^2 + b$，上述的加法規則 3 須改為 $P + Q = R = (x_3, y_3)$,

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

$$\lambda = \begin{cases} \dfrac{y_2 + y_1}{x_2 + x_1} & \text{if } P \neq Q \\ x_1 + \dfrac{x_1}{y_1} & \text{if } P = Q \end{cases} \tag{2}$$

公式(1)與(2)的計算(加法、減法、乘法、除法/反元素)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計算。RSA 或 Diffie-Hellman 密碼系統的基礎為進行模指數運算 $a^b \bmod c$，而點乘法運算 $k \bullet P$ 為橢圓曲線密碼系統的基礎。點乘法計算 $k \bullet P$，其中 k 為正整數而 P 為橢圓曲線上的一個點：

$$k \bullet P = \overbrace{P + P + \ldots + P}^{k\text{個點相加}}$$

如果$n \bullet P = 0$ 則n稱之為點P的級數(order)。在合適的橢圓曲線上，我們可找到一個級數$n>2^{160}$的基點(base point) G，而此橢圓曲線系統參數基點G可公開。隨機選取小於n的正整數d當作私密金鑰，計算$Q = d \bullet G$為對應的公開金鑰。如同我們有多種演算法來加速模指數運算，我們也有多種演算法來加速點乘法運算。

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 [1]與FIPS 186-2 [9] ECDSA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。在網際網路上，我們可找到提供橢圓曲線運算且提供原始碼的公開軟體，例如Crypto++ [4]、LiDIA [7]、PARI-GP[11]等。這些軟體可作為橢圓曲線密碼系統開發的參考，而本文主要在介紹如何迅速的了解橢圓曲線密碼系統的技術，並且說明如何以Mathematica及C語言迅速開發橢圓曲線密碼系統。

貳、橢圓曲線數位簽章演算法ECDSA

我們以橢圓曲線數位簽章演算法 (Elliptic Curve Digital Signature Algorithm，ECDSA) [1][6][9]為例，說明簽章的產生與檢驗方法。假設G是橢圓曲線系統基點且其級數為n，正整數d為簽署者的私密金鑰，而$Q = d \bullet G$則為簽署者的對應公開金鑰，$h(m)$為訊息m的雜湊函數值。在ECDSA標準中，對應於訊息m的簽章(r, s)產生步驟如下:

1. 挑選一亂數 k， $n - 1 \geq k \geq 1$
2. 計算 $k \bullet G = (x_1, y_1)$ 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。

ECDSA 簽章檢驗的步驟如下:

1. 計算$w = s^{-1} \bmod n$
2. 計算$u_1 = h(m)\, w \bmod n$ 與 $u_2 = r\, w \bmod n$.
3. 計算$u_1 \bullet G + u_2 \bullet Q = (x_0, y_0)$ 與 $v = x_0 \bmod n$.
   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，ECDSA 有限體的選擇我們是採用 FIPS 186-2 建議的質數體曲線參數 P-256、P-384、P-521，若採用二元體則簽章長度將增加 8 到 12 位元組。從表二可見相同安全性時兩種演算法簽章的長度差異甚大，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	● 安全性 2^{128}：384 位元組 ● 安全性 2^{192}：960 位元組 ● 安全性 2^{256}：1920 位元組	● 安全性 2^{128}：64 位元組(質數體) ● 安全性 2^{192}：96 位元組(質數體) ● 安全性 2^{256}：132 位元組(質數體)
安全基礎	大數分解。	橢圓曲線離散對數。
優點	演算法歷史悠久容易說明，且同時可用做加解密。	速度快，簽章長度小。
缺點	速度慢，簽章長度較大。	理論較難理解，實現技術較複雜。

參、Mathematica 模擬 ECDSA

Mathematica [3] [12]是個商業軟體，它內建數學函數庫，不但可以在 Windows、Linux/Unix、Mac 等各種平台上直接進行大整數的加減乘除及模算術(modular arithmetic)，也可處理多項式運算。橢圓曲線技術可用做質數判斷，而 Mathematica 的內建 *PrimeQ* 質數判斷模組便包含橢圓曲線的運算。

相對應於橢圓曲線兩點加法的公式(1)，Mathematica 的程式如圖二所示，其中 Mod 為 Mathematica 內建的餘數函數(Mod[*a*,*b*] = *a* mod *b*，亦即 *a*÷*b* 的餘數)，PowerMod[*a*,*b*,*c*] = a^b mod *c*，g2 代表橢圓曲線方程式的係數 *a*，slope 即公式(1)的λ。由於 Mathematica 已內建整數與多項式的資料結構與模算術，所以無論是質數體或二元體的橢圓曲線運算程式很簡潔且容易設計。

```
If [Mod[x1 - x2, p] == 0,
    If [Mod[y1 + y2, p] == 0,
        slope = Mod[(3 x1^2 + g2) PowerMod[2 y1, -1, p], p]
    ],
    slope = Mod[(y2 - y1) PowerMod[x2 - x1, -1, p], p]
];
x3 =   Mod[slope^2 - x1 - x2, p];
y3 =   Mod[slope (x1-x3) - y1, p];
```

圖二：對應於橢圓曲線加法(公式 1)的 Mathematica 程式

圖三為我們以 Mathematica 模擬 ECDSA 的簽章產生。測試資料是取自 ANXI X9.62 Annex J.3.1 的 192 位元質數體橢圓曲線($y^2 = x^3 - 3x + b$)。我們有下列資料：

- 質數體 *p*= 6277101735386680763835789423207666416083908700390324961279
- 橢圓曲線係數 *b* = (16 進制) 64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
- 基點 *x* 座標 = (16 進制) 188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
- 基點 *y* 座標 = (16 進制) 07192b95ffc8da78631011ed6b24cdd573f977a11e794811
- 私密金鑰 *d* = 651056770906015076056810763456358567190100156695615665659
- 基點的級數 *n* = 6277101735386680763835789423176059013767194773182842284081
- 訊息 *m* 的雜湊函數值 *e* = 968236873715988614170569073515315707566766479517
- 亂數值 *k* = 614050706706500106306506556566740556000616155656565656654

利用上述資料，如圖三所示，Mathematica 計算出來的簽章(*r*, *s*)為

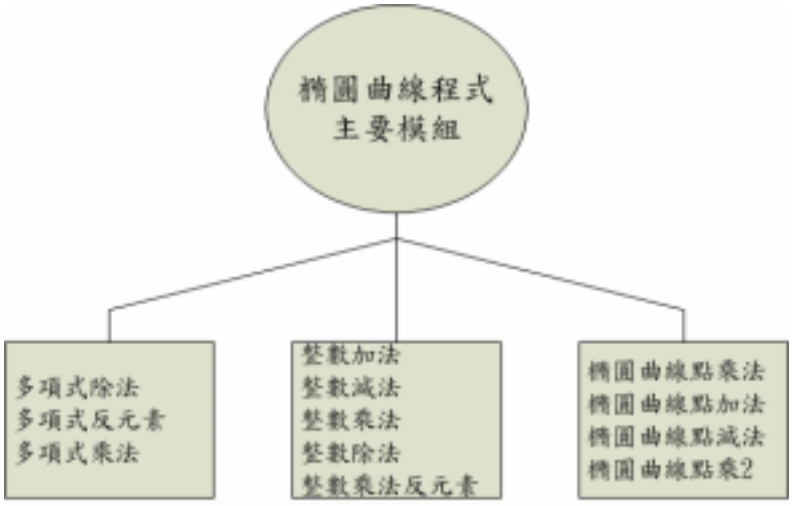
r = 3342403536405981729393488334694600415596881826869351677613

s = 573582232888815525468389499789757195156855364289202998234 2

圖三：以 Mathematica 計算 192 位元有限體 ECDSA 的數位簽章

肆、C 程式執行 ECDSA

用 Mathematica 可讓我們很快進行橢圓曲線密碼系統的測試，也可提供更詳盡的測試資料，但難以整合到網路安全或資訊安全系統。我們可開發橢圓曲線密碼系統 C 語言函式庫模組及獨立可執行程式，主要模組如圖四所示，包括三大部分：(i) 整數運算、(ii)多項式運算、(iii)橢圓曲線運算。由於我們用到的橢圓曲線為 160 位元以上，所以有限體的元素必須以某種資料結構(如陣列)表示並進行運算。



圖四：橢圓曲線密碼系統 C 語言函式庫主要模組

橢圓曲線密碼系統 C 語言的設計可參考 Crypto++［4］、LiDIA［7］、PARI-GP［11］等開放原始碼軟體。圖五是我們的 ECDSA 程式中，ANSI X9.62 Annex J.2.1 二元體 $GF(2^{191})$

ECDSA 的測試模組，圖六為程式中的數位簽章產生模組，圖七為數位簽章檢驗模組。

```
    print_field ("ECDSA over GF(2^191)\npoly_prime = ", &poly_prime);
/* setup co-factor */
    null( &Base.cofactor);
    Base.cofactor.e[NUMWORD] = 2;

/* setup curve coefficients a2, a6*/
    copy (&ecdsa_a, &Base.crv.a2);
    copy (&ecdsa_b, &Base.crv.a6);

/* setup base point and its order */
    copy (&ecdsa_x, &Base.pnt.x);
    copy (&ecdsa_y, &Base.pnt.y);
    copy (&ecdsa_n, &Base.pnt_order);

/* display curve data */
    print_curve("Public curve: ", &Base.crv);
/* display base point */
    print_point("Base point G: \n", &Base.pnt);

    copy (&ecdsa_d, &Signer.prvt_key);
/* display private key */
    print_field("\nSigner's secret key = ", &Signer.prvt_key);
    printf ("Signers public key:\n");
/* point multiplication */
    EC_mul( &Signer.prvt_key, &Base.pnt, &Signer.pblc_key, &Base.crv);
/* display public   key */
    print_point("", &Signer.pblc_key);
    field_to_int( &ecdsa_h, &hash_value);

/*   ECDSA signature generation */
    print_int ("hash value of message = ", &hash_value);
    ECDSA_Sign(hash_value, &Base, &Signer.prvt_key, &signature);
    print_field("signiture (r) = ", &signature.r);
    print_field("signiture (s) = ", &signature.s);
```

圖五、$GF(2^{191})$ ECDSA 程式的測試模組

```
        copy (&ecdsa_k, &random_key.prvt_key);
        printf ("\n");
        print_field("random number k = ", &random_key.prvt_key);

/* compute k * G */
        EC_mul( &random_key.prvt_key, &public_curve->pnt, &random_key.pblc_key,
&public_curve->crv);

/*    compute r */
        field_to_int( &public_curve->pnt_order, &point_order);
        field_to_int( &random_key.pblc_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 */
        field_to_int(secret_key, &key_value);
        int_mul( &key_value, &r_value, &temp);
        int_add( &temp, &hash_value, &temp);
        int_div( &temp, &point_order, &quotient, &k_value);

        field_to_int( &random_key.prvt_key, &temp);
        mod_inv( &temp, &point_order, &u_value);
        int_mul( &u_value, &k_value, &temp);
        int_div( &temp, &point_order, &quotient, &sig_value);
        int_to_field( &sig_value, &signature->s);
}
```

圖六、ECDSA 程式的數位簽章(*r*,*s*)產生模組

```
/*   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 * s */
     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);

/*   convert x value of verify point to an integer modulo point order */
     field_to_int( &Verify.x, &temp);
     int_div( &temp, &point_order, &quotient, &check_value);

/*   compare resultant message digest from original signature    */
     int_sub( &r_value, &check_value, &temp);
     while( temp.hw[0] & 0x8000)          /*    ensure positive zero */
          int_add( &point_order, &temp, &temp);
```

圖七、ECDSA 程式的數位簽章檢驗模組

我們以 ANSI X9.62 Annex J.2.1 的範例做為 ECDSA 程式測試的資料，範例中二元體 $GF(2^{191})$ 是採用 $f = x^{191} + x^{9} + 1$ 不可約的多項式(irreducible polynomial)。

橢圓曲線的定義方程式為 $y^2+xy=x^3+ax^2+b$，其中:

a = （16 進制） 2866537B 67675263 6A68F565 54E12640 276B649E F7526267

b = （16 進制） 2E45EF57 1F00786F 67B0081B 9495A3D9 5462F5DE 0AA185EC

公開的基點 $G = (x_G, y_G)$:

x_G =（16 進制）36B3DAF8 A23206F9 C4F299D7 B21A9C36 9137F2C8 4AE1AA0D

y_G =（16 進制）765BE734 33B3F95E 332932E7 0EA245CA 2418EA0E F98018FB
基點的級數 n =（16 進制）400000000000000000000004a20e90c39067c893bbb9a5
私密金鑰 d =（16 進制）340562e1dda332f9d2aec168249b5696ee39d0ed4d03760f
對應的公開金鑰 $Q = d \bullet G = (x_1, y_1)$ ：

x_1 =（16 進制）5DE37E75 6BD55D72 E3768CB3 96FFEB96 2614DEA4 CE28A2E7

y_1 =（16 進制）55C0E0E0 2F5FB132 CAF416EF 85B229BB B8E13520 03125BA1
訊息雜湊函數值 $h(m)$ =（16 進制）a9993e364706816aba3e25717850c26c9cd0d89d
測試用的亂數值 k =（16 進制）3eeace72b4919d991738d521879f787cb590aff8189d2b69
利用這些數值，我們的 ECDSA 程式執行，如圖八所式，產生的簽章(r, s)為

r =（16 進制）38e5a11fb55e4c65471dcd4998452b1e02d8af7099bb930

s =（16 進制）c9a08c34468c244b4e5d6b21b3c68362807416020328b6e

```
ECDSA over GF(2^191)
poly_prime =   80000000        0        0        0        0      201
Public curve:
a2= 2866537b 67675263 6a68f565 54e12640 276b649e f7526267
a6= 2e45ef57 1f00786f 67b0081b 9495a3d9 5462f5de aa185ec

Base point G:
x: 36b3daf8 a23206f9 c4f299d7 b21a9c36 9137f2c8 4ae1aa0d
y: 765be734 33b3f95e 332932e7  ea245ca 2418ea0e f98018fb

Signer's secret key =   340562e1 dda332f9 d2aec168 249b5696 ee39d0ed 4d03760f
Signers public key:
x: 5de37e75 6bd55d72 e3768cb3 96ffeb96 2614dea4 ce28a2e7
y: 55c0e0e0 2f5fb132 caf416ef 85b229bb b8e13520  3125ba1
hash value of message =
                968236873715988614170569073515315707566766479517

random number k =   3eeace72 b4919d99 1738d521 879f787c b590aff8 189d2b69
signiture (r) =    38e5a11 fb55e4c6 5471dcd4 998452b1 e02d8af7  99bb930
signiture (s) =    c9a08c3 4468c244 b4e5d6b2 1b3c6836 28074160 20328b6e
```

圖八：ECDSA 程式執行(DOS 模式)的輸出

伍、結論

橢圓曲線的數學理論背景雖很深奧，但用於密碼學的橢圓曲線系統卻不難實現，橢圓曲線採二元體時較適合硬體的實現，而質數體較適合軟體的實現。相同安全性時，橢圓曲線密碼系統的金鑰長度遠較目前國內常用的 RSA/Diffie-Hellman 等公開金鑰密碼學演算法的長度為短，且執行速度較快，所以我們可預見橢圓曲線密碼系統將逐漸成為公開金鑰密碼系統的主流。在同一平台下，尋求迅速又簡捷的實現方式也一直是密碼學及網路安全研究人員追求的目標，雖然橢圓曲線密碼系統效率的複雜度較難分析，但這也提供我們更多的研究與創新機會。

[誌謝]

作者感謝日本 NTT 公司於 2001 年暑假的正式訪問邀請及中華電信研究所研究計畫（91-8F02）、國科會研究計畫(NSC 91-2213-E-017-002、NSC 92-2213-E-017-003)的經費補助。

參考文獻

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

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

[3] D.M. Bressoud and S. Wagon, *A Course in Computational Number Theory*, Key College Publishing, 2000.

[4] Wei Dai, Crypto++ library, http://www.eskimo.com/~weidai/cryptlib.html

[5] Department of Defense, *Public Key Infrastructure and Key Management Infrastructure Token Protection Profile*, V3.0, http://www.niap.nist.gov/cc-scheme/PP_PKIKMITKNPP-MR_V3.0.pdf, March 2002.

[6] D. Johnson and A. Menezes, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," Technical Report CORR 99-34, Centre for Applied Cryptographic Research (CACR), University of Waterloo, August 1999. http://www.cacr.math.uwaterloo.ca/techreports/ 1999/corr99-34.pdf

[7] LiDIA, http://www.informatik.tu-darmstadt.de/TI/LiDIA/

[8] A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer, 1993.

[9] NIST, FIPS 186-2, *Digital Signature Standard (DSS)*, October 2001, http://csrc.nist.gov/publications/fips/ fips186-2/fips186-2-change1.pdf

[10] NIST, DRAFT Special Publication 800-57, *Recommendation on Key Management*, January 2003, http://csrc.nist.gov/CryptoToolkit/kms/guideline-1-Jan03.pdf

[11] PARI/GP, http://pari.math.u-bordeaux.fr/

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

[13] S. Wolfram, *Mathematica Book*, 5th ed., Wolfram Research, Inc., 2003.