Speeding Scalar Multiplication of Elliptic Curve Over GF(2^{mn})

Ding Yong[1,2], Yin-Fang Hong[1], Wei-Tao Wang[1], Yuan-Yuan Zhou[1], and Xiao-Yang Zhao[1]
(Corresponding author: Ding Yong)

School of Mathematics and Computational Science, Guilin University of Electronic Technology[1]
Jinji Road 1, Qixing District, Guilin City, Guangxi Province, China
Future Network Centre, City University of Hong Kong Shenzhen Research Institute[2]
Nanyuan Road 68, Futian District, Shenzhen City, Guangdong Province, China
(Received Apr. 3, 2009; revised and accepted May 12, 2009)

Abstract

Lee et al. proposed two methods to speed up the computation of scalar multiplication of elliptic curve defined over $GF(2^{mn})$ with a medium size of m in the range $10 \leq m \leq 20$. In these methods, Frobenius map is utilized to expand the integer k and each coefficient of the expansion is represented as a binary string. In this paper, with the application of joint sparse form (JSF) to the coefficients, some variations of Lee et al.'s methods are proposed to achieve a better performance at a lower storage requirement.

Keywords: Elliptic curve cryptography, frobenius map, joint sparse form, scalar multiplication

1 Introduction

Elliptic curve cryptosystem (ECC) was first proposed by Koblitz [17] and Miller [25] independently, and has been widely studied in recent years. It has the advantage of shorter key length and higher efficiency with the same security level over RSA, which make it more and more popular in applications especially in wireless communication system. Table 1 is listed below to illustrate the key length comparison of ECC and RSA and Table 3 to the speed comparison [25]. As for those desired properties, it had been adopted by many standards and commercial systems.

Table 1: Key length comparison of RSA and ECC

MIPS Year	RSA Key Length	ECC Key Length	Ratio
10^4	512	106	5:1
10^8	768	132	6:1
10^{11}	1024	160	7:1
10^{20}	2048	210	10:1
10^{78}	21000	600	35:1

In most applications of ECC, such as signature, public key encryption, Diffie-Hellman key exchange and so on, Scalar Multiplication (SM) is the basic and most time-consuming operation. It is the k-time addition of point P, where k is a large positive integer and P is a base point on the curve. The computational speed of SM is affected by three aspects: Finite field operations, of which inversion is the most key one [1, 6, 12, 14, 16, 31]; Curve point operation, including double, addition, double and addition etc. [3]; Representation of the scalar k. An overview of the research results on SM of ECC can be taken from [10, 23].

Of the above three aspects, decomposition of scalar k gains most of the recent research interests. For the purpose of speeding up the computation, there are three factors in decomposition need to be considered. Firstly, the base should be selected such that the base time of points is efficient. Secondly, the representation length should be short. Finally, the representation Hamming density, which is the ratio of the non-zero digits to all digits, should be small.

According to the expansion base, scalar expansion methods can be classified as two kinds: integer base methods and endomorphism base methods. Integer 2 was firstly used as a base to decompose k. After that, NAF (Non Adjacent Form), which can assure that there are no two contiguous non-zero digits, was proposed to lower the Hamming density [2]. Furthermore, w window techniques were applied on NAF to get NAF_w, which make the density more and more sparse [5]. Besides 2, any a positive integer bigger than 2 can also be used as base to decompose k. w-based expansions have shorter length [8]. In 2006, double base chain, which uses the integer pair (2,3) as base, was given. It achieves shorter length and lower density [7, 30]. Endomorphism is a map from an ECC group to itself. Selecting an efficient endomorphism as

base is another improvement of method in SM computation. For example, endomorphism τ of Koblitz curve only needs a simple shift-row operation which is almost free when underlying finite field adopts normal basis. τ based decomposition methods has a much faster computational speed [18, 27]. Following τ, many efficient endomorphisms were given [4, 9, 19, 24].

In particular, the Frobenius map which is an endomorphism was used to obtain the Frobenius expansion of k for the curve over GF(2^n) with a small m [26]. Recently, Lee et al. extended the idea to the curve over GF(2^{mn}) with a medium size of $m(10 \leq 20)$ and proposed two algorithms to accelerate the computation of kP [20]. The joint sparse form (JSF) [28] was a variation of signed binary representation of a pair of integers, which could lead to more double zero positions and thus a reduction in computational complexity. In lee *et al's* methods, each coefficient of Frobenius expansion is represented as a binary string and all these bits build up the bits coefficient matrix. Finally, the matrix is deal with longitudinal w-bit combination and transverse w-bit combination respectively to build two methods.

Based on their idea, we have a little modification on the generation of the bits coefficients matrix. After Frobenius expansion of k is obtained, which is $(C_0, C_1, \ldots, C_{l-1})$. If $l \bmod 2 = 1$, set $C_1 = 0$. Let $s = \lfloor \frac{(l-1)}{2} \rfloor$, and JSF is applied on every coefficient pair of $(c_{2x}, c_{2x+1})(x = 0, 1, \ldots, s)$ to obtain the coefficient matrix

$$c = \begin{bmatrix} c_{0,m-1} & \cdots & c_{0,0} \\ \vdots & \vdots & \vdots \\ c_{2s+1,m-1} & \cdots & c_{2s+1,0} \end{bmatrix}$$

.

After the matrix is generated, we use almost the same idea to compute SM. That is to say, our main idea is combination of JSF and Frobenius map inspired by Lee et al.

The rest of this paper is organized as follows. In Section 2, the joint sparse form is briefly introduced. Lee *et al.*'s methods, as well as the Frobenius map are described in Section 3. Then we will propose our modified algorithms in Section 4, together with a detailed analysis of the computational complexity and the storage requirement. Finally, a conclusion is drawn in Section 5.

2 Joint Sparse From

Joint sparse form is a variation of the signed binary representation of a pair of integers that leads to more double zero positions. Algorithm 1 is adopted to obtain the JSF of a given pair of integers a and b. The notation $c = a \bmod b$ means that $c \equiv a \bmod b$ and $-b/2$ ¡ $c \leq b/2$.

- (JSF-1) Of any three consecutive positions, at least one is a double zero. In other words, for any positions i and j, we have $u_{i,j+k} = u_{1-i,j+k}$ for $k = 0$ or $u_{i,j+k} = u_{1-i,j+k} = 0$ for $k = 0$ or ± 1.

Table 2: Speed comparison of RSA and ECC

Functions	Time(ms) ECC-163bit Security Builder 1.2	Time(ms) RSA-1024bit BSAFE 3.0
Key Generation	3.8	4,708.3
Sign	2.1(ECNRA) 3.0(ECDSA)	228.4
Verification	9.9(ECNRA) 10.7(ECDSA)	12.7
DH Exchange	7.3	1,654.0

Algorithm 1 Algorithm 1 (JSF)

Input integer pair a and b Output
JSF of a and b
Process
Set $k_0 = a$ and $k_1 = b$, Set $j = 0$
while k_0 ¿ 0 or k_1 ¿ 0 **do**
  For $i = 0$ to 1 do
  if k_i is even **then**
    $u = 0$
    else
    $u = k_i \bmod 4$
    if $k_i \equiv \pm 3(\text{mod}8)$ and $k_i \equiv 2(\text{mod}4)$ **then**
      $u = -u$
      Set $u_{i,j} = u$
    end if
    next i
    For $i = 0$ to 1 do
    $k_i = (k_i = u_{i,j})/2$
    next i
    $j = j + 1$
  end if
  Periodically refresh the *observations storage*
end while
End
The JSF possesses the following properties that had been proved in [21], namely, JSF-1, JSF-2, JSF-3 and JSF-4.

- (JSF-2) Adjacent terms do not have opposite signs. In other words, there is never the case that $u_{i,j}u_{1-i,j+k} = $ -1.
- (JSF-3) If $u_{i,j+k} \neq 0$ then $u_{1-i,j+k} = \pm 1$, and $u_{1-i,j} = 0$.
- (JSF-4) The probability of occurrence of double zero, which satisfies $u_{i,j+k} = u_{1-i,j+k} = 0$ for position j, is 1/2.

3 Lee et al.'s Methods

For elliptic curve over finite field GF(2^{mn}), where m is a middle size integer generally bounded in the area of [22, 28]. With the utilization of Frobenius map, Lee at

el. Proposed two methods to speed up the scalar multiplication.

1) **Frobenius Expansion**
For a non-supersingular curve over $GF(_{q^n})$, where $q = 2^m$, given by the Weierstrass equation of the form

$$E(GF(q^n)) : y^2 + xy = x^3 + ax^2 + b,$$

where a, $b \in GF(q)$ and $b \neq 0$, Frobenius map is an endomorphism of the elliptic curve group $E(GF(q^n))$. It is defined from $E(GF(q^n))$ to itself as:

$$\phi : E \mapsto E(x, y) \mapsto (x^{\wp}, y^{\wp}).$$

Let $\sharp E(GF(q^n))$ denote the number of $GF(q^n)$-rational point of $E(GF(q^n))$ and $t = q + 1$ - $\sharp E(GF(q^n))$, then the characteristic polynomial of ϕ is ϕ^2 - $t\phi$ + q = 0Athat is $\phi 2(P)$ - $t\phi(P)$ + qP = 0 for all $P \in E(GF(q^n))$. Since there is a natural homomorphism from the ring $Z[\phi]$ to the ring $Z[\phi]$, which $maps\alpha = (t + \sqrt{t^2 - 4q}/2)$ to ϕ, the integer k can be expressed as the sum of $k = \sum c_i\phi^i$. This expression is called the Frobenius expansion of k. For the map, there are following theorems that had been proved in [29].

Theorem 1. *[29] For any given positive integer k, the Frobenius expansion $k = \sum^{l} -1_{i=0} C_i\phi^i(-q/2 < c_i \leq\leq q/2, q \geq 64)$ exists. It is unique and has length $l \leq n + 3$.*

Theorem 2. *[29] For any given positive integer k, the Frobenius expansion $k = \sum^{l} -1_{i=0} c_i\phi^i(0 \leq c_i < q, q \geq 64)$. It is unique and has length $l \leq n + 5$.*

2) **Method 1**
Firstly, the Frobenius expansion of $k = \sum^{l} -1_{i=0} c_i\phi^i(0 \leq c_i < q)$ is obtained. Then, each coefficient c_i is represented as a binary string $(c_{i,m-1}c_{i,m-2}, \ldots, c_{i,1}c_{i,0})$. Finally kP can be computed via the formula:

$$\begin{aligned} kP &= \sum_{i=0}^{l-1} c_i\phi^i(P) \\ &= \sum l - 1_{i=0} \sum_{j=0}^{m-1} c_{i,j}2^j\phi^i(P) \\ &= \sum l - 1_{i=0} 2^j \sum_{j=0}^{m-1} c_{i,j}c_{i,j}\phi^i(P). \quad (1) \end{aligned}$$

Let bit string $a = (a_{w-1}, a_{w-2}, \ldots, a_1, a_0)$ and S_a $=a_{w-1}\phi^{w-1}(P)+a_{w-2}\phi^{w-2}(P)+\ldots+a_1\phi(P)+a_0P$, Equation (1) can be varied to Equation (2) as :

$$kP = \sum_{j=0}^{m-1} 2^j \sum_{i=0}^{l-1} c_{i,j}\phi^i(P), \quad (2)$$

where

$$\begin{aligned} T_j &= \sum_{i=0}^{l-1} c_{i,j}\phi^i(P) \\ &= \sum_{i=0}^{\lceil m/w \rceil - 1} \phi^{wi} S_{(c_{wi+w-1,j}, c_{wi+w-2,j}, \cdots, c_{wi+1,j}, c_{wi,j})}. \end{aligned}$$

Algorithm 2 is the programming description of Method 1.

Algorithm 2 Algorithm 2 (Method 1)

1: Input
2: Integer k, point P
3: Output
4: kP

Pre-computation and Storages:

- Frobenius expansion of $k = \sum_{i=0}^{l-1} c_i\phi^i$ $(0 \leq c_i$ ¡ $q)$
- Binary string $(c_{i,m-1}, c_{i,m-2}, \ldots, c_{i,1}, c_{i,0})$ of each coefficient c_i.
- $S_a = a_{w-1}\phi^{w-1}(P) + a_{w-2}\phi^{w-2}(P) + \ldots + a_1\phi(P) + a_0P$ for all $(a_{w-1}, a_{w-2}, \ldots, a_1, a_0) \in \{0, 1\}^w$, where w is a chosen window size.

Process:

a. Q = O;
b. for $j = m$-1 downto 0 do
  i. for $i = \lceil l/w \rceil - 1$ downto 0 do
    A. $T = \phi^w(T)$;
    B. $a = (c_{wi+w-1,j}, c_{wi+w-2,j}, \ldots, c_{wi+1,j}, c_{wi,j})$;
    C. $T = T + S_a$
  ii. $Q = Q + T$;
c. return Q;

From the algorithm definition, we have the following theorems.

Theorem 3. *Let A stand for point addition, D for Doubling and Φ for Frobenius map ϕ, the total number of operations of Algorithm 2 is $TO \approx mD + (1 - 1/2^w)(mn/w)A + (mn)\Phi$.*

Proof. One by one counting in Algorithm 2, it can be easily obtained that the total number of operations in Algorithm 2.

$$\begin{aligned} TO &= (m-1)D + (1 - 1/2^w) \\ &\quad +(l - 1/2^w)(\lceil l/w \rceil m - 1)A \\ &\quad +((\lceil l/w \rceil - 1)wm)\Phi. \end{aligned}$$

Since l ¡ $n+5$ following Theorem 2, then

$$\begin{aligned} TO &= (m-1)D + (1-1/2^w)(\lceil (n+5)/w \rceil m - 1)A \\ &\quad + ((\lceil (n+5)/w \rceil - 1)wm)\Phi \\ &\approx mD + (l - 1/2^w)(mn/w)A + (mn)\Phi. \end{aligned}$$

□

Theorem 4. *The number of storages required by Algorithm 2 is* $2^w - w - 1$.

Proof. The number of possible values of S_a is 2^w. However, as the map ϕ is almost free (only twice m-bit left-shift) when normal basis representation is utilized on the finite field, then $\phi^i (i = 0, 1, \ldots, w-1)$ and O needn't be stored. Hence the theorem holds. □

3) **Method 2**

In fact, we may take all the binary strings of the Frobenius expansion coefficients as a binary matrix. Method 1 can be looked as a column combination. On the other hand, when raw combination is applied on the matrix, Method 2 is proposed. In Method 2, as in Method 1, firstly, the Frobenius expansion of $k = \sum_{i=0}^{l-1} c_i \phi^i (-q/x < c_i \leq q/2)$ is obtained. Then, each coefficient $|c_i|$ is represented as a binary string $(c_{i,m} c_{i,m-3}, \ldots, c_{i,1} c_{i,0})$. Finally kp can be computed via the formula:

$$\begin{aligned} kP &= \sum_{i=0}^{l-2} c_i \phi^j (P) \\ &= \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \varepsilon_i c_{i,j} 2^j \phi^i (P), \end{aligned} \tag{3}$$

where $\varepsilon_i = c_i / |c_i|$.

Let bit string $a = (a_{w-1}, a_{w-2}, \ldots, a_1, a_0)$ and $S_a = a_{w-1} 2^{q-1} P + a_{w-2} x^{w-2} P + \ldots + a_1 2P + a_0 P$, then Equation (3) will be modified to Equation (4):

$$\begin{aligned} kP &= \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \varepsilon_i c_{i,j} 2^j \phi^i (P) \\ &= \sum_{r=0}^{\lceil m/w \rceil - 1} 2^r T'_r \end{aligned} \tag{4}$$

where

$$\begin{aligned} T_r &= \sum_{j=rw}^{rw-1} 2^{j-rw} \sum_{i=0}^{l-1} \varepsilon_i c_{i,j} \phi^i (P) \\ &= \sum_{i=0}^{l-1} \varepsilon_i \phi^i S_{(c_{i,wr+w-1}, c_{wr+w-2}, \ldots, c_{i,wr+1}, c_{i,wr})} \end{aligned}$$

Algorithm 3 is the programming description of Method 2.

Algorithm 3 Algorithm 3 (Method 2)

1: Input
2: Integer k, point P
3: Output
4: kP

Pre-computation and Storages:

a. Frobenius expansion of
$$k = \sum_{i=0}^{l-1} c_i \phi^i (-q/2 \leq c_i \leq q/2);$$
b. Binary string $(c_{i.m-1}, c_{i,m-2}, \ldots, c_{i,1} c_{i,0})$ of each coefficient c_i.
c. $S_a = a_{w-1} 2^{q-1} P + a_{w-2} x^{w-2} P + \ldots + a_1 2P + a_0 P$ for all $a = (a_{w-1}, a_{w-2}, \ldots, a_1, a_0) \in \{0,1\}^w$, where w is a chosen window size.

Process:

a. $Q = \mathrm{O}$;
b. for $j = \lceil m/w \rceil$-1 downto 0 do
   i. for $Q = 2^w Q, T = O$;
   ii. for $i = l - 1$ downto 0 do
      A. $T = \phi(T)$; $\varepsilon_i = c_i / |c_i|$;
      B. $a = (c_{i,jw+w-1}, c_{i,jw+w-2}, \cdots, c_{i,jw+1}, c_{i,jw})$;
      C. $T = T + \varepsilon_i S_a$.
   iii. $Q = Q + T$;
c. return Q;

Based on Method 2, we have the following theorems.

Theorem 5. *Let A stand for point addition, D for Doubling and Φ for Frobenius map ϕ, the total number of operations of Algorithm 3 is*

$$TO \approx mD + (l - 1/2^w)(mn/w)A + (mn/w)\Phi.$$

Proof. From the procedures of Algorithm 3, it can be easily counted one by one that the total number of operations

$$\begin{aligned} TO &= ((\lceil m/w \rceil - 1)w)D \\ &\quad + ((1 - 1/2^w)(l-1)\lceil m/w \rceil)A \\ &\quad + (l-1)\lceil m/w \rceil \Phi. \end{aligned}$$

From Theorem 1, we have $l \leq n + 3$, then

$$\begin{aligned} TO &= ((\lceil m/w \rceil - 1)w)D \\ &\quad + ((1 - 1/2^w)(l-1)(n+3-1)\lceil m/w \rceil)A \\ &\quad + ((n+3-1)\lceil m/w \rceil)\Phi \approx mD \\ &\quad + (1 - 1/2^w)(mn/w)\Phi. \end{aligned}$$

□

Theorem 6. *Theorem 6 The number of storages required by Algorithm 3 is* 2^w *- 2.*

Proof. The number of possible values of S_a is 2^w. It is obvious that O and P needn't be stored. Thus the theorem holds. □

If normal basis representation is applied in the finite field, which makes the map ϕ almost free, Algorithm 2 is more efficient. Otherwise, Algorithm 3 is more efficient, especially when the improvement of [3] is utilized to compute $2^w P$.

4 The Proposed Method

The idea of Lee et *al.'s* methods is like this. First, the Frobenius expansion of k is obtained. Then each coefficient of the expansion is represented as a binary string and all these bits build up the coefficient matrix. Finally, the matrix is deal with longitudinal w-bit combination by Algorithm 2, and transverse w-bit combination by Algorithm 3. In Algorithm 2, if $S_a = 0$, then the addition can be saved. In situation that the amount of storage is limited, the value of w needs to be very small. In particular, if $w = 2$, JSF of c_i and c_{i+1} can be applied to increase the probability of getting $s_a = 0$ and thus reduce the point additions in the computation of kP. This forms the idea of our proposed Method 1. On the other hand, if the storage resource is rich, a combination of the transverse w positions of the JSF of c_i and c_{i+1} is used in our proposed Method 2. We not only give a detailed description or our methods, but also have an accurate evolution on the number of atomic operations and storages. Furthermore, we do a comparison on both methods between ours and Lee at el.'s.

4.1 The Proposed Method 1

Let $k = \sum_{i=0}^{l-1} c_i \phi^i (-q/2 < c_i \leq q/2)$ be the Frobenius expansion of k as before. If $l \bmod 2 = 1$, set $c_l = 0$. Let $s = \lceil (l-1)/2 \rceil$, and JSF is applied on every coefficient pair of $(c_{2x}, c_{2x+1})(x = 0, 1, \ldots, s)$ to obtain the coefficient matrix

$$c = \begin{bmatrix} c_{0,m-1} & \cdots & c_{0,0} \\ \vdots & \vdots & \vdots \\ c_{2s+1,m-1} & \cdots & c_{2s+1,0} \end{bmatrix}$$

.

Let $a = (a_1, a_0)$, and $s_a = a_1/phi(P) + a_0 P$, then we have the Equation (5):

$$kP = \sum_{j=0}^{m-1} \sum_{i-0}^{l-1} c_{i,j}/phi^i(P) = \sum_{j=0}^{l-1} 2^j T_j \qquad (5)$$

The Equation (5) can be programmed as Algorithm 4, which is our proposed Method 1.

Pre-computation and Storages:

Algorithm 4 Algorithm 4 (proposed Method 1)

1: Input
2: Integer k, point P
3: Output
4: kP

- Frobenius expansion of

$$k = \sum_{i=0}^{l-1} c_i \phi^i (-q/2 < c_i \leq q/2).$$

- JSF of (c_{2x}, c_{2x+1}) for $(x = 0, 1, \ldots, s)$.
- Coefficient matrix $c = \begin{bmatrix} c_{0,m-1} & \cdots & c_{0,0} \\ \vdots & \vdots & \vdots \\ c_{2s+1,m-1} & \cdots & c_{2s+1,0} \end{bmatrix}$.
- $S_a = a_1\phi(P) + a_0 P$ for $(a_1, a_0) \in \{0, \pm 1\}^2$ (only $\phi(P) \pm P$ needs to be stored).

Process:

1) $Q = \mathrm{O}$;
2) for $j = m$-1 downto 0 do
   a. for $Q = 2Q, T = O$;
   b. for $i = s$ downto 0 do
      i. $T = \phi^2(T)$;
      ii. $a = (c_{2i+1,j}, c_{2i,j})$;
      iii. $T = T + S_a$
   c. $Q = Q + T$;
3) return Q;

In Algorithm 4, we have the following theorems about the computing speed and the storages.

Theorem 7. *Let A stand for point addition, D for Doubling and* Φ *for Frobenius map* ϕ*, the total number of operations in Algorithm 4 is* $TO \approx mD + (mn/4)A + (mn)\Phi$.

Proof. From property JSF-4, we have $P(S_a = O) = 1/2$. From the procedure, the total number of atomic operations in Algorithm 4 is

$$TO = (m-1)D + (1 - 1/2)(\lceil l/2 \rceil)A + ((\lceil l/2 \rceil - 1)2m)\Phi.$$

Following Theorem 1, we have $l \leq n + 3$, thus

$$TO = (m-1)D + (1-1/2)(\lceil (n+3)/2 \rceil m - 1)A + ((\lceil (n+3)/2 \rceil m - 1)A.$$

□

Theorem 8. *The amount of storage required by Algorithm 4 is 2.*

Table 3: Comparison of our method with Lee's on Method 1

	TO	Storages
Lee's Scheme (w=2)	$mD + (3nm/8)$ $A + (mn)\Phi$	1
The Proposed Scheme	$mD + (mn/4)$ $A + (mn)\Phi$	2

Proof. Only $\phi(P) \pm P$ need to be stored. If $a \in \{(0,0),(\pm1,0),(0,\pm1)\}$, it is obviously that S_a need not be stored. If $a = (-1,1)$ then $S_a = -(\phi(P) - P)$. Similarly, if $a = (-1,-1)$, $S_a = -(\phi(P) + P)$. □

Comparing with Algorithm 2 for the case $w = 2$, about $mn/8$ additions are reduced with the cost of an additional storages.

4.2 The Proposed Method 2

If the amount of storage is not too restricted, one can choose Method 2. Let the coefficient matrix

$$c = \begin{bmatrix} c_{0,m-1} & \cdots & c_{0,0} \\ \vdots & \vdots & \vdots \\ c_{2s+1,m-1} & \cdots & c_{2s+1,0} \end{bmatrix}$$

be the same as the proposed Method 1.

Let

$$a = \begin{bmatrix} a_{0,w} & \cdots & a_{0,0} \\ a_{1,w-1} & \cdots & a_{1,w-1} \end{bmatrix}$$

and

$$S_a = \sum_{j=0}^{w-1} s^j \sum_{i=0}^{l} a_{i,j}\phi^i(P),$$

then Equation (4) will be modified as Equation (6):

$$kP = \sum_{i=0}^{l-1}\sum_{j=0}^{m-1} c_{i,j}x^j\phi^i(P) = \sum_{r=0}^{\lceil m/w \rceil} 2^r T'_r \qquad (6)$$

where

$$T_r = \sum_{i=0}^{l-1} \phi^{2i} S_{(c_{i,2r+1}, c_{i,2r})}$$

Equation (6) is also used in our Method 2, which can be programmed as Algorithm 5.

Algorithm 5 Algorithm 5 (Proposed Method 2)

1: Input
2: Integer k, point P
3: Output
4: kP

Pre-computation and Storages:

- Frobenius expansion of

$$k = \sum_{i=0}^{l-1} c_i\phi^i (-q/2 < c_i \le q/2).$$

- JSF of (c_{2x}, c_{2x+1}) for $(x = 0, 1, \ldots, s)$.
- Coefficient matrix $c = \begin{bmatrix} c_{0,m-1} & \cdots & c_{0,0} \\ \vdots & \vdots & \vdots \\ c_{2s+1,m-1} & \cdots & c_{2s+1,0} \end{bmatrix}$.
- Compute

$$S_a = \sum_{j=0}^{w-1} 2^j \sum_{i=0}^{l} a_{i,j}\phi^i(P)$$

$$a = \begin{bmatrix} a_{0,w-1} & \cdots & a_{0,0} \\ a_{1,w-1} & \cdots & a_{1,w-1} \end{bmatrix}$$

and $a_{i,j} \in \{0, \pm 1\}$-. If $b = -a$ then S_b need not to be stored as $S_b = -S_a$.

Process:

1) $Q = \mathrm{O}$;
2) for $j = \lceil m \rfloor$-1 downto 0 do
   a. for $Q = 2^w Q, T = O$;
   b. for $i = s$ downto 0 do
      i. $T = \phi^2(T)$;
      ii. $a = a = \begin{bmatrix} a_{2i,jw+w-1} & \cdots & a_{2i,jw} \\ a_{2i+1,jw+w-1} & \cdots & a_{2i+1,jw} \end{bmatrix}$;
      iii. $T = T + S_a$
   c. $Q = Q + T$;
3) return Q;

Based on Algorithm 5, we have the theorems below.

Theorem 9. *Let A stand for point addition, D for Doubling and Φ for Frobenius map ϕ, the total number of operations required by Algorithm 5 is $TO \approx mD + (1 - 1/2w)(mn/(2w)) + (mn/w)\Phi$.*

Proof. From property JSF-4, we have $P(S_a = O) = 1/2^w$. From the procedures of Algorithm 5, it can be easily obtained that the total number of operations is $TO = ((\lceil m/w \rceil - 1)w)D + ((1-1/2^w)(\lfloor l - 1/2 \rfloor)\lceil m/w \rceil)A + (\lfloor (l - 1/2) \rfloor)\lceil m/w \rceil \Phi$. By Theorem 1, we have $l \le n + 3$. Hence

$$\begin{aligned} TO = {} & ((\lceil m/w \rceil)w)D \\ & + ((1 - 1/2^w)(\lfloor (n + 3 - 1)/2 \rfloor)\lceil m/w \rceil)A \\ & + ((\lfloor (n + 3 - 1)/2 \rfloor)\lceil m/w \rceil)\Phi \approx mD \\ & + (1 - 1/2^w)(mn/w)\Phi. \end{aligned}$$

□

Table 4: Comparison of our method with Lee's on Method 2

	Total Operations	Strorages	
Our	mD+(1-1/2^w) (mn/2w) A+ (mn/w)Φ	16	w=2
		62	w=3
		608	w=4
Lee's	mD+(1-1/2^w)(mn/w)A+(mn/w) Φ	2	w=2
		6	w=3
		14	w=4

Theorem 10. *The number of storages required by Algorithm 5 is as below:*

1) If $w = 2$, it is 16;

2) If $w = 3$, it is 62;

3) If $w = 4$, it is 308.

Proof. The number of possible values of S_a is 3^{2w}. With the restriction of JSF-1, JSF-2 and JSF-3, the number decreases a lot. Since only either one of a and $-a$ needs to be stored as well as $\phi(P)$ and P needn't be stored. Thus it can be checked one by one when $w = 2, 3, 4$ respectively to get result of the theorem. □

When the improvement of [3] is utilized to compute 2^wP, Algorithm 5 can be further accelerated. Comparing with Algorithm 3, about $(1-1/2^w)(\mathrm{mn}/2w)$ additions are saved. However, the amount of storage required increases substantially. Thus it is suitable for small w. For example, Algorithm 5 with $w = 2$ has the same computational complexity as Algorithm 3 with $w = 5$ with a lower amounts of storages. The same result can be obtained for $w = 3$ of Algorithm 5 to $w = 7$ of Algorithm 3,also $w = 4$ of Algorithm 5 to $w = 9$ of Algorithm 3. However, for $w > 4$, the number of storages is too large and our proposed Method 2 has no practical meaning. For this reason, the storages number needed for $w > 4$ needn't to be cared.

5 Conclusion

Based on the coefficient matrix of the Frobenius expansion of the integer k when computing the scalar multiplication kP of elliptic curve cryptography over $GF(2^{mn})$ with a medium size m, the idea of longitudinal w-bit combination and transverse w-bit combination are separately proposed by Lee *et al.* to form Algorithms 2 and 3 to speed up the computation. For Algorithm 2, we apply the JSF form on the coefficients to obtain the coefficient matrix and propose our Method 1. It is suitable for limited storage situation and saves $mn/8$ point additions with an additional storage when compared with Algorithm 2 of $w = 2$. Based on proposed Method 1, the idea of transverse w-position combination is utilized to form our Method 2. It saves about $(1 - 1/2^w)$ $(mn/2w)$ point additions when compared with Algorithm 3. The value of w can be selected to balance the computational speed and the storage requirements. As the amount of storage increases substantially with w, it is practical for relatively small w. For example this algorithm with $w = 2$ has the same result as Algorithm 3 with $w = 5$, but the storage required is fewer. The value of w can also be selected as $w = 3$ or $w = 4$ if the storages resource is rich. Nevertheless, it has no any practical meaning for $w > 4$ as the storages number is too large.

Acknowledgements

The work described in this paper was supported by The Project Supported by Guangxi Science Foundation (0991079) , ShenZhen Bureau of Science Technology and Information Fundamental research plan 2009(Research and Implementation on Key Technology of trust Wifi-3G cross-platform Communication System) and open grant from of Key Lab of Computer Networks and Information Security of Ministry of Education, Xidian University (2008CNIS-03).

References

[1] A. Z. Abdulan, "High speed modular divider based on GCD algorithm," *Proceeding of ICICS'07*, pp. 189-200, Zhengzhou, China, 2008.

[2] S. Arno, and F.S. Wheeler, "Signed digit representation of minimal hamming weight," *IEEE Transactions on Computers*, no. 42, pp. 1007-1010, 1993.

[3] M. Ciet, M. Joye, K. Lauter, and P. Montgomery, "Trading inversion for multiplication in elliptic curve cryptography". (http://eprint.iacr.org/2003/)

[4] M. Ciet, T. Lange, F. Sica, and J. Quisquater, "Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms," *Proceeding of Advances in Cryptology (Eurocrypt'03)*, LNCS 2656, pp. 388-400, Springer-Verlag, 2003.

[5] H. Cohen, "A course in computational algebraic number theory," *Graduate Texts in Mathematics*, vol. 138, pp. 233-235, Springer-verlag, 1996.

[6] A. Daly, W. Marnane, T. Kerinsy, "Fast modular division for application in ECC on reconfigurable logic," *Proceeding of FPL'03*, pp. 786-795, Springer-Verlag, 2003.

[7] V. S. Dimitrov, L. Imbert, and P. K. Mishra, "Fast elliptic curve point multiplication using double-base chains," *Cryptology ePrint Archive*, Report 2005/069, 2005.

[8] Y. Ding, K. W. Wong, and Y. M. Wang, "A w-NNAF method for the efficient computation of scalar multiplication in elliptic curve cryptography," *Applied Mathematics and Computation*, vol. 167, no. 1, pp. 81-93, 2004.

[9] R. P. Gallant, J. L. Lambert, and S. A. Vanstone, "Faster point multiplication on elliptic curves with effcient endomorphisms," *Proceedings of Crypto' 01*, LNCS 2139, pp. 190-200, Springer-Verlag, 2001.

[10] D. M. Gordon, "A survey of fast exponentiation methods," *Journal of Algorithms*, vol. 27, no. 1, pp. 129-146, 1998.

[11] R. R. Goundar, K. I. Shiota, and M. Toyonaga, "SPA resistant scalar multiplication using golden ratio addition chain method," *International Journal of Applied Mathematics,* vol 38, no.7, pp. 38-42, June 2008.

[12] J. Guajardo, C. Paar, "Itoh-tsuji inversion in standard basis and its application in cryptography and codes," *Design, Codes and Cryptography*, vol. 25, no. 2, pp. 207-216, 2002.

[13] D. Hankerson, J. Lopez, and A. Menezes, "Software implementation of elliptic curve cryptography over binary fields," *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 1-24, 2000.

[14] K. C. Hoon, S. Kwon, and H. C. Pyo, "Efficient bit-serial systolic array for division over GF(2^m) elliptic curve cryptosystem applications," *Proceedings of the 2003 International Symposium on Circuits and Systems*, pp. 252-255, IEEE, Los Alamitos,2003.

[15] Inforsec company ECC liary. (http://www.hids.com.cn/data.asp)

[16] H. C. Kim, and H. C. Pyo, "High-speed division architecture for GF(2m)," *Electronics Letters*, vol. 38, no. 15, pp. 835-836, 2002.

[17] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, no. 48, pp. 203-209, 1987.

[18] N. Koblitz. "CM curves with good cryptographic properties," *Proceedings Crypto '91*, pp. 279-287, Springer-Verlag, 1992.

[19] T. Lange, *Efficient Arithmetic on Hyperelliptic Koblitz Curves*, PhD Thesis, University of Essen, 2001.

[20] D. H. Lee, S. Chee, S. C. Hwang, and J. C. Ryou, "Improved scalar multiplication on Elliptic curve defined over F2^{mn}," *ETRI Journal*, vol. 26, no. 3, pp. 241-251, June 2004.

[21] D. Liu, and Y. Q. Dai, "A new algorithm of elliptic curve multi-scalar multiplication," *Chinese Journal of Computers*, vol. 31, no. 7, pp.1131-1137, July 2008.

[22] J. Lopez, and R. Dahab, "Improved algorithm for elliptic curve arithmetic over GF(2^n)," *Proceeding of Selected Areas in Cryptography (SAC'98)*, LCNS 1556, pp. 201-212, 1999.

[23] J. Lopez, and R. Dahab, *An Overview of Elliptic Curve Cryptography*, Technical report, Institute of Computing,State University of Compinas, Brazil, 22nd of May 2000. (http://citeseer.nj.nec.com/333066.html)

[24] V. Mäuller, "Efficient point multiplication for elliptic curves over special optimal extension fields," *Public-Key Cryptography and Computational Number Theory*, pp. 197-207, Warschau, Poland, Sep. 11-15, 2000.

[25] V.S. Miller, "Use of elliptic curves in cryptography," *Proceeding of Advances in Cryptology (Crypto'85)*, LNCS 218, pp. 417-426, Springer-Verlag, 1986.

[26] V. Muller, " Fast multiplication on elliptic curves over small fields of characteristic two," *Journal of Cryptology* , vol. 11 pp. 219-234, 1998,.

[27] J. Solinas, "Efficient arithmetic on Koblitz curves," *Designs, Codes, and Cryptography*, pp. 195-249, 2000.

[28] J. A. Solinas, *11Low-weight Binary Representations for Pairs of Integers*, Technical Report CORR 2001-41, CACR. (www.cacr.math.uwaterloo.ca/techreports/2001/corr2001-41)

[29] K. W. Wong, E. C.W. Lee, L.M. Cheng, and X. F. Liao, " Fast elliptic scalar multiplication using new double-base chain and point halving," *Applied Mathematics and Computation*, vol. 183, no. 2, pp. 1000-1007, Dec. 15, 2006.

[30] K. W. Wong, E. C. W. Lee, L. M. Cheng, and X. F. Liao, "Fast elliptic scalar multiplication using new double-base chain and point halving". (http://eprint.iacr.org/2006/124.pdf)

[31] D. Yong, and G. Feng, "High speed modular divider based on GCD algorithm over GF(2^m)," *Journal on Communications*, vol. 29, no. 10, pp. 199-204, Oct. 2008.

Ding Yong was born on June 1975 in Chongqing China. He graduated with a B.E degree from Dept. of Mathematics, Sichuan University, China, in 1998. He received the M.S degree and the PhD degree in Xidian University, China, in 2003 and 2005, respectively. Now, he is a Research Fellow in Dept. of Computer Science, City University of Hong Kong. He is also an Assistant Professor in School of Mathematics and Computational Science, Guilin University of Electronic Technology. His research interests are cryptography and network security.

Yin-Fang Hong was born on Oct. 1980 in Hunan China. She is now a postgraduate student in School of Mathematics and Computational Science in Guilin University of Electronic Technology. Her research interests is public key cryptography.

Wei-Tao Wang was born in 1987. He is now a graduate student in School of Mathematics and Computational Science in Guilin University of Electronic Technology.

Yuan-Yuan Zhou was born in 1988. She is now a graduate student in School of Mathematics and Computational Science in Guilin University of Electronic Technology.

Xiao-Yang Zhao was born in 1989. He is now a graduate student in School of Mathematics and Computational Science in Guilin University of Electronic Technology.