Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 37(1), 2010, Pages 27–34
ISSN: 1223-6934

Elliptic Curve Cryptosystems and Scalar Multiplication

Nicolae Constantinescu

Abstract. One of the most used cryptosystems in the world is the RSA system. Its popularity is due to its high security level. In the last decades, the studies have shown that the cryptosystems based on elliptic curves have the same security level as the RSA system. Besides that, the elliptic curve cryptosystems have a higher efficiency and they use shorter keys. In this paper we describe basics of the elliptic curve cryptosystems. We, also, present methods (based on the Lee et al. methods) of the scalar multiplications of the elliptic curves. In the end, we propose an efficient method for simultaneous multiple point multiplication.

2010 Mathematics Subject Classification. Primary 94A55; Secondary 11T71, 68P25.
Key words and phrases. Elliptic Curve, Cryptosystem, Scalar Multiplication.

1. Introduction

The elliptic curves are an area of mathematics and have been discovered for almost a century. Neal Koblitz [21] and Victor Miller [20] have, independently, proposed, for the first time, the use of the elliptic curves in cyptography. This was almost three decades ago, in 1985. Since then, they are widely studied and the cryptographic systems based on elliptic curves become more and more popular.

Definition 1.1. *An elliptic curve over a field K is given by:*

$$E:\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_i \in K$. The above equation is named the Waierstrass equation.

Definition 1.2. *The discriminant of an elliptic curve given in the Waierstrass form is:*

$$\Delta = d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6$$

where:

$$\begin{aligned} d_2 &= a_1 + 4a_2 \\ d_4 &= 2a_4 + a_1a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \end{aligned}$$

and $\Delta \neq 0$.

For two extension fields of K the points of the curve are:

$$E(L) = \{(x, y) \in L \times L | E = 0\} \cup O$$

where O is the infinity point.

Received December 02, 2009. Revision received February 06, 2010.

If $K = F_p$ where $p > 3$ is a prime the Waierstrass equation can be simplified to:

$$E : y^2 = x^3 + ax + b$$

The discriminant of this curve is $\Delta = -16(4a^3 + 27b^2)$. If we have the point $P(x, y)$ then the inverse will be $-P(x, -y)$. If we have $P(x_1, y_1)$ and $Q(x_2, y_2)$ then $P + Q = R(x_3, y_3)$ is given by:

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

where $\lambda = \frac{y_1 - y_2}{x_1 - x_2}$. For doubling a point $2P(x_3, y_3)$ we use the formulas:

$$x_3 = \lambda^2 - 2x_1$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

where $\lambda = \frac{3x_1^2 + a}{2y_1}$.

If x is replaced with x/z and y with y/z, where $z \neq 0$ we obtain the equation:

$$y^2 z = x^3 + axz^2 + bz^3$$

The infinity point is $O(0, 1, 0)$ and the inverse of $P(x, y, z)$ is $-P(x, -y, z)$.

The elliptic curves over a binary field are given by:

$$E : y^2 = x^3 + ax + b$$

The discriminant is $\Delta = b$, and for a point $P(x, y)$ the inverse is $-P(x, x + y)$. Addition and doubling are computed in the same way as on the prime curves. The projective coordinates are, also, used in the same way as on the prime curves. The only difference is that for a point $P(x, y, z)$ the inverse is $P(x, x + y, z)$. The reader can learn more about arithmetic of elliptic curves in [15, 17, 16].

The elliptic curve cryptosystems are very efficient and they, also, have a high security level. The efficiency depends the most on the scalar multiplication. This operation has a high time-consuming. The computational speed of such an operation is influenced by:

- finite field operations;
- curve point operations;
- representation of the scalar k [1, 2].

The scalar multiplication kP, is in fact the adding of the point P to itself k times. That means:

$$kP = \underbrace{P + P + P + \ldots + P}_{k \text{ times}}$$

and $-kP = k(-P)$.

2. State of Art

Multiplication for general elliptic curves can be done with various methods [14, 13]: binary method [8], the addition-substraction method [8], the k-ary method [9, 10], the Montgomery method [18]. From these general methods the Montgomery methods is considered to be the fastest [19]. If an elliptic curve admits an efficient endomorphism, its use can speed up scalar multiplication. For special elliptic curves, the multiplier

k can be represented with the Frobenius map. Some of the methods which use the Frobenius map were proposed by Neal Koblitz [11] and Volker Muller [12].

Lee *et al.* presented two methods for computing scalar multiplications. The two algorithms accelerate the computation of kP, where P is a point of the elliptic curve over $GF(2^{mn})$[5]. In this paper, we discuss only the popular one. Its popularity is due to the fact that the storages are reduced. In both their methods Lee *et al.* used the Frobenius map.

Definition 2.1. *Frobenius map*

Let E be an elliptic curve over F_q with q elements. The q-th power Frobenius map ϕ_q on $E(F_q)$ is given by:

$$\phi_q : (x, y) \mapsto (x^q, y^q)$$

Then we have:

$$\#E(F_q) = q + 1 - t$$

where t is named the trace of ϕ_q. The above equation is equivalent with [3]:

$$\phi_q^2 - t\phi_q + q = 0 \text{ in the ring of the endomorphisms of } E$$

Theorem 2.1. *Let E be a non-supersingular elliptic curve over F_q. We denote E_m a curve regarded over the extension field F_{q^m}, where $m \geq 2$. Then, the multiplication of an integer k on E_m is given by:*

$$k = \sum_{i=0}^{m+2} c_i \phi^i$$

where $c_i \in \{j \in Z | -q/2 < j \leq q/2\}$.

Definition 2.2. *Frobenius expansion*

If E from the above theorem is defined over $GF(q)$, where $q = 2$ or $q = 3$, to obtain a Frobenius expansion of k we consider E to be defined over $GF(q^2)$. Then the Frobenius expansion is:

$$k = \sum_{i=0}^{\lceil \frac{m}{2} \rceil} c_i \phi_{q^2}^i$$

where $c_i \in \{j \in Z | -q^2 < j \leq q^2/2\}$

An algorithm for Frobenius expansion is described in [18]. The following theorem has be proven in [4].

Theorem 2.2. *For an integer $k > 0$ the Frobenius extension is $k = \sum_{i=0}^{t-1} c_i \phi_q^i (-q/2 < c_i \leq q/2, q \geq 64)$. The extension is unique and has the length $t \leq m + 3$.*

Besides Frobenius map, Lee *et al.*, also, applied Joint sparse form (JSF). The JSF was invented by Jerry Solinas of the NSA [7]. The form is applied to obtain more double zero positions for a pair of binary integers. For example, if we want to compute $13x + 15y$, in binary we have:

$$\begin{aligned} 13 &= 1011 \\ 15 &= 1111 \end{aligned}$$

We observe that every column has at least one 1. If we compute the operation as $(17 - 4)x + (16 - 1)y$ we have:

$$\begin{aligned} 17 &= 10001 \\ 4 &= 00100 \\ 16 &= 10000 \\ 1 &= 00001 \end{aligned}$$

An algorithm for implementing JSF is presented in [6].

Lee et. al. idea was, also, based on Muller's method described bellow:

Muller's method is used for computing jP where $1 \leq j \leq q/2$. Thus, the scalar multiplication is given by:

$$kP = \left(\sum_{j=0}^{l_1-1} c_j \phi^j \right)(P)$$

$$kP = \phi(\ldots \phi(\phi(c_{l_1-1}P) + c_{l_1-2}P) \ldots + c_1 P) + c_0 P$$

This is applicable only to a small sized m.

Lee *et al.* method has the following steps:

(1) Compute the Frobenius extension of $k = \sum_{i=0}^{m-1} c_i \phi^i (0 \leq c_i < q)$;
(2) The coefficients c_i are represented as a binary strig $(c_{i,n-1}, c_{i,n-2}, \ldots, c_{i,1}, c_{i,0})$;
(3) Compute $kP = \sum_{i=0}^{m-1} c_i \phi^i(P) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} c_{i,j} 2^j \phi^i(P)$;

To simplify the last computation we take $a = a_{w-1}, a_{w-2}, \ldots, a_1, a_0$, a bit string, and $S_a = a_{w-1}\phi^{w-1}(P) + a_{w-2}\phi^{w-2}(P) + \ldots + a_i\phi(P) + a_0 P$. Then, the last step can be computed as:

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

where T_j is given by:

$$T_j = \sum_{i=0}^{m-1} c_{i,j} \phi^i(P) = \sum_{i=0}^{\lceil \frac{n}{w} \rceil - 1} \phi^{wi} S_{c_{wi+w-1,j}, \ldots, c_{wi,j}}$$

The following algorithm was described in [6]:

Algorithm 1 Lee et. al. algorithm for scalar multiplication

1: $Q = O$
2: **for** $j = n - 1$ downto 0 **do**
3: $\quad Q = 2Q$
4: $\quad T = O$
5: $\quad$ **for** $i = \lceil m/w \rceil - 1$ downto 0 **do**
6: $\quad\quad T = \phi^w(T)$
7: $\quad\quad a = (c_{wi+w-1,j}, c_{wi+w-2,j}, \ldots, c_{wi+1,j}, c_{wi,j})$
8: $\quad\quad T = T + S_a$
9: $\quad$ **end for**
10: $\quad Q = Q + T$
11: **end for**
12: return Q

The input for this algorithm are the integer k and the point $P \in E$, while the output is the multiplication kP. Also, the precomputations and the storages for this algorithm are the following:

- $k = \sum_{i=0}^{m-1} c_i \phi^i (0 \leq c_i < q)$ (Frobenius expansion of k);
- for each c_i compute $c_{i,n-1}, c_{i,n-2}, \ldots, c_{i,1}, c_{i,0}$;
- compute $S_a = a_{w-1}\phi^{w-1}(P) + a_{w-2}\phi^{w-2}(P) + \ldots + a_1\phi(P) + a_0 \forall (a_{w-1}, \ldots, a_0) \in \{0,1\}^w$.

An improved method for this algorithm was proposed in [6]. The main ideea is to decrease the value of w. If w is small enough the amount of storage is considerably improved. The authors take $w = 2$ and apply JSF of c_i and c_{i+1}. This increases the probability that $S_a = 0$. The steps of the algorithm are described below:

(1) Compute the Frobenius expansion of $k = \sum_{i=0}^{m-1} c_i \phi^i (-q/1 < c_i \leq q/2)$.
(2) If $m \ mod \ 2 = 1$ then $c_i = 0$
(3) $s = \lfloor (m-1)/2 \rfloor$
(4) Apply JSF on each pair (c_{2x}, c_{2x+1}) $\forall x = 0, 1, \ldots, s$
(5) $a = (a_1, a_0)$ and $S_a = a_1\phi(P) + a_0 P$
(6) Compute $kP = \sum_{j=0}^{n-1} 2^j \sum_{i=0}^{m-1} c_{i,j}\phi^i(P) = \sum_{j=0}^{n-1} 2^j T_j$ where $T_j = \sum_{i=0}^{m-1} c_{i,j}\phi^i(P) = \sum_{i=0}^{\lfloor m/2 \rfloor - 1} \phi^{2i} S_{(c_{2i+1,j}, c_{2i,j})}$

The input and the output are the same as in the Lee *et. al.* algorithm: integer k, point P and, respectively, kP. The precomputation and the storage are:

- $k = \sum_{i=0}^{m-1} c_i \phi^i (-q/2 < c_i \leq q/2)$ (Frobenius expansion)
- JSF of $(c_2 x, c_{2x+1} \forall (x = 0, 1, \ldots, s))$
- $S_a = a_1\phi(P) + a_0 P \forall (a_1, a_0) \in \{0, \pm 1\}^2$

The algorithm is:

Algorithm 2 The improved method of the Lee. et. al. algorithm

```
1: Q = O
2: for j = n − 1 downto 0 do
3:    Q = 2Q
4:    T = O
5:    for i = s downto 0 do
6:       T = φ²(T)
7:       a = (c_{2i+1,j}, c_{2i,j})
8:       T = T + S_a
9:    end for
10:   Q = Q + T
11: end for
12: return Q
```

3. Our Method

3.1. Premises.

The method we propose in this paper is for computing simultaneous multiple point multiplications. It is based on the Muller's method and on the algorithm described earlier.

The method has the input the integers k and l and the points of the elliptic curve P and R, and the output $kP + lR$. In fact we have to compute:

$$kP + lR = \underbrace{P + P + P + \ldots + P}_{k \text{ times}} + \underbrace{R + R + R + \ldots + R}_{l \text{ times}}$$

For computing $kP + lR$ we know that:

$$kP = \sum_{j=0}^{n-1} 2^j T_j$$
$$lR = \sum_{j=0}^{n-1} 2^j T'_j$$

where $T_j = \sum_{i=0}^{\lfloor m/2 \rfloor - 1} \phi^{2i} S_{(c_{2i+1,j}, c_{2i,j})}$ and $T'_j = \sum_{i=0}^{\lfloor m/2 \rfloor - 1} \phi^{2i} S_{(c'_{2i+1,j}, c'_{2i,j})}$. So, we have:

$$kP + lR = \sum_{j=0}^{n-1} 2^j (T_j + T'_j) = \sum_{j=1}^{n-1} \sum_{i=0}^{\lfloor m/2 \rfloor - 1} \phi^{2i} (S_{(c_{2i+1,j}, c_{2i,j})} + S_{(c'_{2i+1,j}, c'_{2i,j})})$$

3.2. Parameters.

The precomputation and the storage for our method are:

- $k = \sum_{i=0}^{m-1} c_i \phi^i (-q/2 < c_i \leq q/2)$ (Frobenius expansion for k)
- $l = \sum_{i=0}^{m-1} c'_i \phi^i (-q/2 < c'_i \leq q/2)$ (Frobenius expansion for l)
- JSF of $(c_{2x}, c_{2x+1}\ \forall x = 0, 1, \ldots, s)$
- JSF of $(c'_{2x}, c'_{2x+1}\ \forall x = 0, 1, \ldots, s)$
- $S_a = a_1 \phi(P) + a_0 P \forall (a_1, a_0) \in \{0, \pm 1\}^2$
- $S_{a'} = a'_1 \phi(P) + a'_0 P \forall (a'_1, a'_0) \in \{0, \pm 1\}^2$

3.3. Algorithm.

A simple algorithm for our method is described below. It represents an improvement brought to the algorithm 1 for the particular case $w = 2$.

Algorithm 3 Simultaneous multiple point multiplication

1: $Q = O$
2: **for** $j = n - 1$ downto 0 **do**
3: $\quad Q = 2Q$
4: $\quad T = O$
5: $\quad T' = O$
6: $\quad$ **for** $i = s$ downto 0 **do**
7: $\quad\quad T = \phi^2(T)$
8: $\quad\quad T' = \phi^2(T')$
9: $\quad\quad a = (c_{2i+1,j}, c_{2i,j})$
10: $\quad\quad a = (c'_{2i+1,j}, c'_{2i,j})$
11: $\quad\quad T = T + S_a$
12: $\quad\quad T' = T' + S_{a'}$
13: $\quad$ **end for**
14: $\quad Q = Q + T + T'$
15: **end for**
16: return Q

3.4. Comparison.

For Lee's improved algorithm presented earlier, the total number of operations is: $nD+A((nm/4)+(nm)\Phi)$ where A means point addition, D doubling and Φ Frobenius map. The demonstration is described in [6]. That means that for computing kP we need $nD + A(nm/4) + (nm)\Phi$ operations. So, for computing kP and lR, applying this algorithm two times, we will have $2(nD + A((nm/4) + (nm)\Phi))$ operations. For computing $kP + lR$ with the method we proposed the number of operations will be smaller. This is because $Q = 2Q$ will be computed for n times, unlike $2n$ times. So, by applying our method we will have up to $nD + 2(A((nm/4) + (nm)\Phi))$. This may not seem a big difference, but when k and l are large numbers, the computing time is considerable improved. The following image illustrates the two complexities of the algorithms.

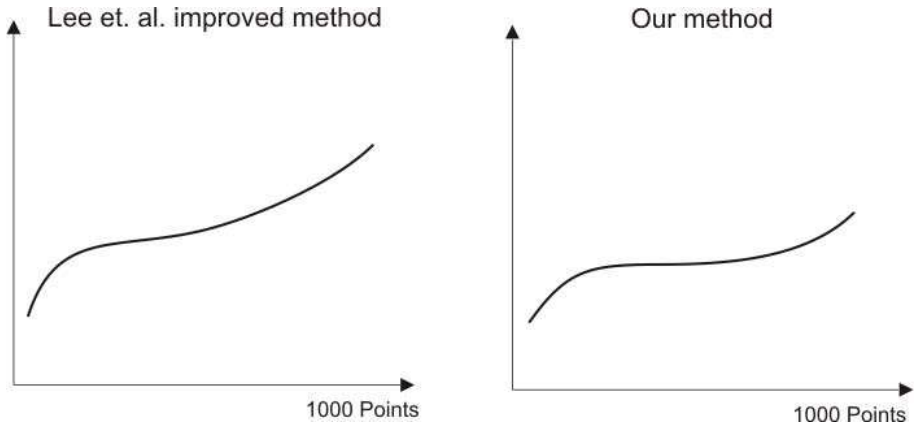

FIGURE 1

The complexity is presented for 1000 points from the elliptic curve. If the number of points increases, the difference between the two figures will be more obvious.

4. Conclusions

In the last years, the cryptosystems based on elliptic curves are increasingly used. This is because their security level and the efficiency one are very high. The complexity of these cryptographic systems depends the most on the scalar multiplications. The Frobenius map and the JSF are often used to decrease the operations' number for multiplying. The presented algorithms are some of the most optimized methods. All three algorithms described were proposed for elliptic curves defined over subfields. If the size of subfields is too small, it is hard to find good cryptographic curves. So, it is recommended to use subfields like F_{2^m}. The simultaneous multiple point multiplication is often used for replacing the simple multiplication. Our method efficiently computes $kP + lR$. The algorithm is practical for relatively small w. The method can be applied not only for $w = 2$, but, also, for $w = 3$ or $w = 4$. In this case, the storage resource is rich. If $w > 4$ the method cannot be applied because the storage is highly increased.

References

[1] D. Yong and G. Feng, High speed modular divider based on GCD algorithm over GF(2m), *Journal on Communications* **29** (2008), no. 10, 199–204.

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

[3] J. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, 1992.

[4] K.W. Wong, E.C.W. Lee, L.M. Cheng and X. Liao, Fast elliptic scalar multiplication using new double-base chain and point halving, *Applied Mathematics and Computation* **183** (2006), no. 2, 1000–1007.

[5] D.H. Lee, S. Chee, S.C. Hwang and J.-C. Ryou, Improved Scalar Multiplication on Elliptic Curve Defined over $2_{F^m n}$, *ETRI Journal* **26** (2004), no. 3.

[6] D. Yong, H. Yin-fang, W. Wei-tao, Z. Yuan-yuan and Z. Xiao-yang, A Combination of Joint Sparse Form and Frobenius Map in Scalar Multiplication of Elliptic Curve over $GF(2^{mn})$, *DBTA* (2009), 707–710.

[7] J.A. Solinas, Low-Weight Binary Representations for Pairs of Integers, *Technical Report CORR* **41** (2001).

[8] F. Morain and J. Olivos, Speeding up the Computations on an Elliptic Curve Using Additions-Substraction Chains, *Inform. Theory. Appl.* **24** (1990), 531–543.

[9] J. Guajardo and C. Paar, Efficient Algorithms for Elliptic Curve Cryptosystems, *Proc. Crypto '97* (1997), 342–356.

[10] K. Koyama and Y. Tsuruoka, Speeding up Elliptic Cryptosystems by Using Signed Binary Window Method, *Proc. Crypto '92* (1993), 43–56.

[11] N. Koblitz, CM Curves with Good Cryptographic Properties, *Proc. Crypto'91* (1992), 279-287.

[12] V. Muller, Fast Multiplication on Elliptic Curve over Small Field of Characteristic Two, *J. of Cryptology* **11** (1998), 219–234.

[13] Y. Ding, Kwok-wo Wong and Yu-min Wang, A w-NNAF Method for the Efficient Computation of Scalar Multiplication in Elliptic Curve Cryptography, *Applied Mathematics and Computation* **167**, no. 1, 81–93.

[14] W. Meier and O. Staffelbach, Efficient Multiplication on Certain Non-supersingular Elliptic Curves, *Proc. Crypto '92* (1993), 333-344.

[15] H. Cohen and G. Frey, Handbook of Elliptic and Hyperelliptic Curve Cryptography, *CRC* (2006), 267–385.

[16] I. Blake, G. Seroussi and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 2002.

[17] D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curves Cryptography*, Springer, 2003.

[18] D.H. Lee, S. Chee, S.C. Hwang and J.-C. Ryou, Improved Scalar Multiplication on Elliptic Curves Defined over F_2^{mn}, *ETRI Journal* **26** (2004), no. 3.

[19] D. Hankerson, J.L. Hernandez and A. Menezes, Software Implementation of Elliptic Curve Cryptography over Binary Fields, *Cryptographic Hardware Embedded System (CHES 2000)* (2000), 1–24.

[20] V. Miller, Uses of elliptic curves in cryptography, *Advances in Cryptology, CRYPTO 85* (1986), 417–426.

[21] N. Koblitz, Elliptic curve cryptosystems, *Mathematics of Computation*, **48** (1987), 203–209.

(Nicolae Constantinescu) Department of Mathematics, University of Craiova,
Al.I. Cuza Street, No. 13, Craiova RO-200585, Romania, Tel. & Fax: 40-251412673
E-mail address: nikyc@central.ucv.ro