Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 862508, 7 pages
http://dx.doi.org/10.1155/2013/862508

Research Article

A Novel Elliptic Curve Scalar Multiplication Algorithm against Power Analysis

Hongming Liu, Yujie Zhou, and Nianhao Zhu

Shanghai Jiao Tong University, Shanghai 200240, China

Correspondence should be addressed to Hongming Liu; liuhongming@sjtu.edu.cn

Received 13 November 2012; Revised 10 March 2013; Accepted 12 March 2013

Academic Editor: Jun-Juh Yan

Nowadays, power analysis attacks are becoming more and more sophisticated. Through power analysis attacks, an attacker can obtain sensitive data stored in smart cards or other embedded devices more efficiently than with any other kind of physical attacks. Among power analysis, simple power analysis (SPA) is probably the most effective against elliptic curve cryptosystem, because an attacker can easily distinguish between point addition and point doubling in a single execution of scalar multiplication. To make elliptic curve scalar multiplication secure against SPA attacks, many methods have been proposed using special point representations. In this paper, a simple but efficient SPA-resistant multiscalar multiplication is proposed. The method is to convert the scalar into a nonadjacent form (NAF) representation at first and then constitute it in a new signed digit representation. This new representation is undertaken at a small precomputation cost, as each representation needs just one doubling and 1/2 additions for each bit. In addition, when combined with randomization techniques, the proposed method can also guard against differential power analysis (DPA) attack.

1. Introduction

Since being proposed independently by Koblitz [1] and Miller [2] in the mid 1980s, elliptic curve cryptosystem (ECC) has been widely applied in public key cryptography, especially in pairing cryptosystems [3, 4]. This is due to ECC using a much shorter key size than other traditional public key cryptosystems such as RSA to provide a corresponding level of security. For instance, 160 bit ECC provides about the equivalent level of security as 1024 bit RSA [5]. Due to the shorter key length, higher speed, and lower power consumption, ECC has been attractive for wireless and smart card applications which have limited bandwidth and storage resources. The security of ECC is based on the hardness of the discrete logarithm problem (DLP) on an elliptic curve called elliptic curve discrete logarithm problem (ECDLP) [6]. Given a scalar multiplication $kP = Q$, where k is an integer, and P, Q are the points on an elliptic curve, according to ECDLP, if k is large enough, then it is unable to calculate k when the values of P and Q are given. When ECC is implemented in the wireless or smart card devices, they are very vulnerable to power analysis attacks [7, 8].

In power analysis attacks, which were proposed by Kocher et al. [8], an attacker can obtain the secret key stored inside a device by monitoring the cryptographic device's power consumption. Generally speaking, simple power analysis (SPA) [7] and differential power analysis (DPA) [8] are the two main types of power analysis attacks. SPA can observe secret information through analysis on a single execution of a cryptographic operation, while DPA may need many executions and to analyze them using a statistical process. As different operations executed by the device consume different amounts of time and power, the different time and power consumptions can be used to determine which operations were performed in what order. In the case of scalar multiplication, it may be possible for an attacker to distinguish which parts of the operation were performed by a point doubling, and which parts were performed by a point addition, although he has no knowledge about the private keys. With this the attacker can obtain the secret key, from the acquired information, for using as the scalar in the elliptic curve scalar multiplication.

In the past decade, various scalar multiplication algorithms that resist SPA or DPA have been proposed, for

example, [9–17]. In [10], Coron first generalized a DPA attack to the elliptic curve cryptosystem and introduced double-and-add always algorithm to resist SPA and randomization method to resist DPA. Since then, many countermeasures, which are based on randomization and scalar recoding, have been proposed. Reference [11] proposed the randomized addition-subtraction chains method, and [17] introduced the randomized window method. Lee first proposed a SPA-resistant countermeasure based on multiscalar multiplication [13]. Then, [14–16] not only introduced multi-scalar multiplication to resist SPA, but also used randomization method to resist DPA. However, those methods provide security at the cost of efficiency. In this paper, we propose a novel efficient SPA-resistant multi-scalar multiplication method and combine it with a randomization method to resist DPA.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to ECC, scalar multiplication, and previous SPA-resistant algorithms. Section 3 describes the proposed scalar multiplication algorithm. Section 4 compares the performance of our strategies with the previous countermeasures. Finally, Section 5 concludes this paper.

2. Preliminaries

2.1. Elliptic Curve Arithmetic. This subsection presents a brief introduction to ECC. For extended details, the reader can refer to [6, 18]. Let $GF(p)$ be a finite prime field. An elliptic curve E over $GF(p)$ can be defined by the Weierstrass equation:

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

where $a_1, a_2, a_3, a_4, a_6 \in GF(p)$. The set of points on an elliptic curve E and the point at infinity (denoted by O) form an Abelian group under a point addition operation. The formula for computing point operation consists of two basic operations: the elliptic curve addition (ECADD) when computing $P+Q$ according to a group addition rule when two points P and Q on the curve are given and P is not equal to Q, and the elliptic doubling (ECDBL) when computing $2P$ when a point P is given. This needs expensive filed inversions in the computation of point operations when using (x, y) known as affine coordinates to represent the points on the curve E. So, the most efficient implementations adopt representations of the form $(X : Y : Z)$, known as projective coordinates, including standard projective coordinates, Jacobian projective coordinates, Chudnovsky Jacobian coordinates, and Lopez-Dahab projective coordinates [6].

2.2. Scalar Multiplication. Scalar multiplication is the basic operation in ECDSA signature [19] and ECDH key agreement [20] protocols. The operation calculates the multiples of a point $kP = P + P + \cdots + P$ (k times), where P is a point on curve E and k is an integer scalar. As the most time consuming operation in the previously mentioned protocols, many algorithms have been proposed to improve the efficiency of scalar multiplication during the past decade. Among them, the nonadjacent form (NAF) [21] is the standard one. An NAF of a positive integer k is an expression

$$k_{\text{NAF}} = \sum_{i=0}^{l-1} k_i 2^i, \quad (2)$$

where $k_i \in \{0, \pm 1\}$, $k_{l-1} \neq 0$, and no two consecutive digits of k_i are nonzero. The computation of NAF of a positive integer k is described as in Algorithm 1. Then, it is possible to compute the scalar multiplication using NAF method following Algorithm 2.

Each positive k has a unique NAF. Among all signed binary representations, NAF(k) has the fewest nonzero digits. It is known that the average density of nonzero bits of NAF is approximately 1/3. This means that scalar multiplication using NAF needs n ECDBL + (n/3) ECADD.

Input: Positive integer $k, k = (k_{l-1}, k_{l-2}, \ldots, k_0)_2$
Output: $k_{\text{NAF}}, k_{\text{NAF}} = (k_l, k_{l-1}, k_{l-2}, \ldots, k_0)_{\text{NAF}}$
(1) $i = 0$.
(2) While ($k > 0$) do
  (2.1) if (k is odd) then
  (2.2) $k_i = 2 - (k \bmod 4)$;
  (2.3) else
  (2.4) $k_i = 0$;
  (2.5) $k = (k - k_i)/2$;
  (2.6) $i = i + 1$;
(3) Return (k).

Algorithm 1: NAF of a positive integer k.

Input: Positive integer k, $P \in E$
Output: $Q = kP$
(1) Use Algorithm 1 to compute k_{NAF}.
(2) $Q = O$.
(3) For i from $l - 1$ downto 0 do
  (3.1) Q = 2Q.
  (3.2) If $k_i = 1$ then $Q = Q + P$;
  (3.3) If $k_i = -1$ then $Q = Q - P$;
(4) Return (Q).

Algorithm 2: Binary NAF method for scalar multiplication.

2.3. Previous Algorithms

2.3.1. Ciet and Joye's Algorithm. This algorithm [14] uses the variant of Shamir's double ladder to compute the multi-scalar multiplication $k1P + k2Q$. The main difference is to insert a dummy operation in the computation. So, each loop includes one doubling and one addition, and the operation order is DADADADADA in Algorithm 3. Hence, one point doubling and one point addition per bit is needed.

2.3.2. Lee's Algorithm. To resist SPA, Lee improved the simultaneous scalar multiplication [21] in [13]. He changed the

Table 1: Transformation rules of Lee's algorithm.

(k_{i+1}, m_{i+1})	(k_i, m_i)	(k'_{i+1}, m'_{i+1})	(k'_i, m'_i)
(0, 0)	(0, 0)	(0, 1)	(0, −2)
(0, 1)	(0, 0)	(0, 2)	(0,−2)
(1, 0)	(0, 0)	(1, 1)	(0, −2)
(1, 1)	(0, 0)	(1, 0)	(0, 2)

values of (k_i, m_i) when $(k_i, m_i) = (0, 0)$ to construct another adequate digit pair with at least one non-zero digit. Of course, the adjacent pair (k_{i+1}, m_{i+1}) should be modified as well. The transformation rules can be described as in Table 1.

After the transformation, the digit pair (k_i, m_i) cannot be all zero. Therefore, the modified simultaneous scalar multiplication was proposed by Lee to resist SPA; see Algorithm 4. Obviously, the cost of Lee's algorithm is also one point doubling and one point addition per bit.

2.3.3. Zhang, Chen, Xiao's Algorithm. This algorithm [16] proposes four scalar multiplication algorithms against power analysis. Those algorithms are all based on the highest-weight binary form (HBF) of the scalars and randomization to resist power analysis. Although those four countermeasures have no dummy operations, the efficiency of them is similar to Ciet and Joye's algorithm. They also almost need one point doubling and one point addition per bit. One of these algorithms can be seen as follows in Algorithm 5.

2.3.4. Liu, Tan, and Dai's Algorithm. Liu et al. also propose a multi-scalar multiplication to resist SPA in [15]. The difference is that they use a joint sparse form (JSF) to represent a pair of integers and process two or three JSF columns each time. Although the processed column number may be different, the algorithm always performs four point doublings and two point additions in each loop. This means that it is not possible for useful information related to the private key to be obtained by the attacker through SPA.

Next, $k = 213$, $m = 408$ are selected as a simple example. The JSF of (213, 408) is

$$\begin{aligned} 213 &= \begin{pmatrix} 0 & 1 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 1 \end{pmatrix} \\ 408 &= \begin{pmatrix} 1 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & 0 & 0 \end{pmatrix}. \end{aligned} \tag{3}$$

Then, this algorithm processes the JSF columns as follows:

$$\begin{matrix} \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & -1 \end{pmatrix} & \begin{pmatrix} 0 & -1 & -1 \end{pmatrix} & \begin{pmatrix} 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} -1 & -1 \end{pmatrix} & \begin{pmatrix} 0 & -1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 \end{pmatrix}. \end{matrix} \tag{4}$$

That is to say, it needs four iterations to complete the whole operation; so, sixteen point doubling and eight point addition operations are required. The theoretical analysis and simulation results show that this algorithm needs 1.384 point doublings and 0.692 point additions per bit.

3. The New Algorithm

Based on the algorithms mentioned earlier, a simple but efficient scalar multiplication algorithm is proposed to resist known power analysis attacks in this section. The method first transforms the NAF of the multi-scalar (k, m) then combines it with the window method and modifies the value of the digit pair with all zero digits; so, the new algorithm can be obtained.

Input: $P, k = (k_{l-1}, k_{l-2}, \ldots, k_0)_2, S,$
$d = (d_{l-1}, d_{l-2}, \ldots, d_0)_2$
Output: $Q = [k]P + [d]S$
(1) $R_1 = P; R_2 = S; R_3 = P + S;$
(2) $j = 2d_{l-1} + k_{l-1}; R_0 = R_j$
(3) For i from $l - 2$ downto 0 do
(3.1) $R_0 = 2R_0, R_1 = P;$
(3.2) c = $!(k_i \,||\, d_i)$, $j = 2d_i + k_i;$
(3.3) $R_c = R_c + R_j;$
(4) Return (R_0).

Algorithm 3: Ciet and Joye's algorithm.

Input: $P, k = (k_{l-1}, k_{l-2}, \ldots, k_0)_2, S, m = (m_{l-1}, m_{l-2}, \ldots, m_0)_2$
Output: $Q = [k]P + [m]S$
(1) (k, m) is transformed to (k', m') as Table 1 rule;
(2) Pre-computation
$T[0, 1] = S; T[1, 0] = P; T[1, 1] = P + S;$
$T[0, 2] = 2S; T[0, -2] = -2S;$
(3) $R = T[k'_{l-1}, m'_{l-1}];$
(4) For i from $l - 2$ downto 0 do
(4.1) $R = 2R;$
(4.2) $R = R + T[k'_i, m'_i];$
(5) Return (R).

Algorithm 4: Lee's algorithm.

Input: $P, k = (k_{l-1}, k_{l-2}, \ldots, k_0)_2$
Output: $Q = [k]P$
(1) Select a random r;
$r = (r_{l-1}, r_{l-2}, \ldots, r_0)_2;$
$k' = k - r;$
(2) HBF(r, l);
(3) For i from 0 downto $l - 1$
If $r + k'_i = 0$
$(k'_{l-1}, \ldots, k'_{i+1}) + k'_i, k'_i = -k'_i$
(4) Construct a pre-computation table T
$T = O; T[1] = P; T[-1] = -P;$
$T[2] = 2P; T[-2] = -2P;$
(5) For i from $l - 1$ downto 0 do
(5.1) $T = 2T;$
(5.2) $T = T + T[r + k'_i];$
(6) Return (T).

Algorithm 5: Zhang, Chen, Xiao's algorithm.

3.1. Scalar Representation. In a scalar multiplication, it can be seen that each bit requires at least a point doubling, while

Table 2: Values of SHW(w) and SPN(w) for different window sizes.

w	2	3	4	5	6
SHW(w)	1	2	2	3	3
SPN(w)	2	5	10	21	42

Table 3: Values of MHW(w) and MPN(w) for different window sizes.

w	2	3	4	5	6
MHW(w)	2	4	4	6	6
MPN(w)	12	60	220	924	3612

the number of point addition varies in different algorithms. Therefore, the best way to improve the efficiency of the scalar multiplication is to reduce the number of point addition. In this approach, a window method with the NAF form is used to reduce the number of point addition.

First, the window method with the NAF form for a single scalar k is described. Let w be the window size, SHW(w) the maximum Hamming weight, and SPN(w) the point number of the precomputation table in each window. The NAF of a scalar k is denoted by k_{NAF}, and generally it can be represented as (2), where l is the bit length of the NAF of k. When using the window method, it can be represented as $\sum_{i=0}^{m-1} k_i' 2^i$, where $m = l/w$ and w bit $k' \in S_w$, in which S_w is the set of all possible w-bit parts of the NAF integers. Consider

$$\mathrm{SHW}(w) = \begin{cases} \dfrac{w}{2} & (w \text{ is even}), \\ \left\lfloor \dfrac{w}{2} \right\rfloor + 1 & (w \text{ is odd}), \end{cases} \tag{5}$$

$$\mathrm{SPN}(w) = \begin{cases} 2^{(w-1)} + 2^{(w-3)} + \cdots + 2^1 & (w \text{ is even}), \\ 2^{(w-1)} + 2^{(w-3)} + \cdots + 2^0 & (w \text{ is odd}). \end{cases} \tag{6}$$

In Table 2, we only list the values of SHW(w) and SPN(w) for w from 2 to 6, but from (6), it can be seen that SPN(w) rises by times with the increase of w.

Next, the window method with the NAF form for multi-scalar (k, m) is introduced, where w is the window size, MHW(w) is the maximum Hamming weight, and MPN(w) is the point number of pre-computation table in each window (Table 3). Now, in each window, there are two scalars; so, MHW(w) is double SHW(w), but MPN(w) is much larger than SPN(w). MPN(w) is the combination of two numbers ($\mathrm{SPN}_k(w)$, $\mathrm{SPN}_m(w)$). Consider

$$\mathrm{MHW}(w) = \begin{cases} \dfrac{w}{2} & (w \text{ is even}), \\ \left\lfloor \dfrac{w}{2} \right\rfloor + 1 & (w \text{ is odd}), \end{cases} \tag{7}$$

$$\begin{aligned} \mathrm{MPN}(w) = {} & \mathrm{SPN}_k(w) + \mathrm{SPN}_m(w) \\ & + \mathrm{SPN}_k(w) * \mathrm{SPN}_m(w) * 2. \end{aligned} \tag{8}$$

Table 4: Transformation rules of new algorithm.

(m_{i+3}, m_{i+2})	(m_{i+1}, m_i)	(m'_{i+3}, m'_{i+2})	(m'_{i+1}, m'_i)
(0, 0)	(0, 0)	(0, 1)	(−2, 0)
(0, −1)	(0, 0)	(−1, 0)	(2, 0)
(0, 1)	(0, 0)	(1, 0)	(−2, 0)
(−1, 0)	(0, 0)	(0, −1)	(−2, 0)
(1, 0)	(0, 0)	(0, 1)	(2, 0)

From (8), it can be seen that MPN(w) rises exponentially with an increase of w. So, in this paper, $w = 2$ was selected as the window size.

When $(k_{i+1}, k_i) = (0, 0)$ and $(m_{i+1}, m_i) = (0, 0)$, no point addition is performed. Hence, an attacker can determine this case through SPA. To assist SPA, $(k_{i+1}, k_i) = (0, 0)$ and $(m_{i+1}, m_i) = (0, 0)$ should be converted, so that a real point addition happens. This is to say that $(k_{i+1}, k_i) = (0, 0)$ and $(m_{i+1}, m_i) = (0, 0)$ are converted to another digit pair with at least one non-zero digit. In this paper, we select to transform $(m_{i+1}, m_i) = (0, 0)$. Of course, the adjacent pair (m_{i+3}, m_{i+2}) should be considered as well. According to the NAF coding rule, there are five possible cases related to the digit pair (m_{i+3}, m_{i+2}). The transformation rules can be described as in Table 4.

After the transformation, the digit pair (m'_{i+1}, m'_i) adds two more cases, but the pre-computation table only adds one more point due to the different symbol between the two cases.

3.2. Proposed New Multiscalar Multiplication Algorithm. Now, the new algorithm to calculate $[k]P + [m]S$ based on the new representation mentioned earlier can be described. The algorithm has a uniform doubling and adding operation, but no dummy operation.

From Algorithm 6, it can be found that two doublings and one addition are performed in each window. It is assumed that the power consumption of subtraction is the same as the addition. There are fifteen points in the pre-computation table.

Next, a simple example that shows how Algorithm 6 works is described, and the NAF of (209, 416) is

$$\begin{aligned} 209 &= (0 \ \ 1 \ \ 0 \ \ -1 \ \ 0 \ \ 1 \ \ 0 \ \ 0 \ \ 0 \ \ 1), \\ 416 &= (1 \ \ 0 \ \ -1 \ \ 0 \ \ 1 \ \ 0 \ \ 0 \ \ 0 \ \ 0 \ \ 0). \end{aligned} \tag{9}$$

After transformation according to Table 4, the new representation of (209, 416) is

$$\begin{aligned} 209 &= (0 \ \ 1 \ \ 0 \ \ -1 \ \ 0 \ \ 1 \ \ 0 \ \ 0 \ \ 0 \ \ 1), \\ 416 &= (1 \ \ 0 \ \ -1 \ \ 0 \ \ 0 \ \ 1 \ \ 2 \ \ 0 \ \ 0 \ \ 0). \end{aligned} \tag{10}$$

Then, the process of Algorithm 6 computing $209P + 416S$ is illustrated in Table 5.

3.3. Proposed DPA-Resistant Scalar Multiplication Algorithm. Among the various DPA countermeasures [22], random key splitting is the most common method to resist DPA. The scalar k can be split in at least two different ways; one is

Table 5: Example of Algorithm 6.

i	(k_{i-1}, k_{i-2}) (m_{i-1}, m_{i-2})	Operation	R
9	(0, 1) (1, 0)	$R = T[1, 2]$	$P + 2S$
7	(0, −1) (−1, 0)	$R = 2R$; $R = 2R$; $R = R - T[1, 2]$	$3P + 6S$
5	(0, 1) (0, 1)	$R = 2R$; $R = 2R$; $R = R + T[1, 1]$	$13P + 25S$
3	(0, 0) (2, 0)	$R = 2R$; $R = 2R$; $R = R + T[0, 4]$	$52P + 104S$
1	(0, 1) (0, 0)	$R = 2R$; $R = 2R$; $R = R + T[1, 0]$	$209P + 416S$

Table 6: Performance comparison of algorithms.

	Point number in Pre-Computation table	Computations per bit	
		Doublings	Additions
Algorithm in [14]	3	1	1
Algorithm in [13]	5	1	1
Algorithm in [16]	5	1	1
Algorithm in [15]	5	1.384	0.692
Algorithm 7	4	1	0.5

$k = (k - r) + r$, and the other is $k = \lfloor k/r \rfloor r + (k \bmod r)$, where r is random and the length of r is the same of k. In this paper, the first way $k = k1 + k2$ was chosen, where $k1 = k - r$, $k2 = r$. It can be observed that this method is the same as multi-scalar approach. Then, a similar method as in Algorithm 6 can be used to compute the scalar multiplication. The difference is that the point S is equal to P. Of course, the transformation rule and pre-computation table are also different. To assist an SPA, $k1$ and $k2$ when $(k1_{i+1}, k1_i) + (k2_{i+1}, k2_i) = (0, 0)$ should be converted, so that a real point addition happens. Algorithm 7 describes this in detail.

It can be seen that there are only four points in the pre-computation table for Algorithm 7, but the program sequence is also DDA. Therefore, due to the uniform operation sequence, it can resist SPA, and to ensure that there is no correlation between two times, a random r was inserted. Of course, the attacker cannot obtain any information through DPA.

4. Performance Comparison

In this section, the performance of Algorithm 7 is analyzed and compared with previous algorithms. In Algorithm 7, each loop processes two bits and has two point doubling and one point addition. Each bit needs one point doubling and 1/2 point addition. In order to show the performance of Algorithm 7, a comparison with previous methods is listed in Table 6.

Input: P, k, S, m
Output: $Q = [k]P + [m]S$
(1) Compute the NAF form of (k, m);
(2) Add leading 0's in (k, m) so that the number of bits in (k, m) is multiple of 2, $k = (k_{l-1}, k_{l-2}, \ldots, k_0)_{\text{NAF}}$, $m = (m_{l-1}, m_{l-2}, \ldots, m_0)_{\text{NAF}}$;
(3) $i = 0$;
(4) While $(i < (l - 2))$ do
(4.1) if $((k_{i+1}, k_i) = (0, 0)$ and $(m_{i+1}, m_i) = (0, 0))$ then
(4.2) if $((m_{i+3}, m_{i+2}) = (0, 0))$ then
(4.3) set $(m_{i+3}, m_{i+2}) = (0, 1)$ and $(m_{i+1}, m_i) = (-2, 0)$;
(4.4) else if$((m_{i+3}, m_{i+2}) = (0, -1))$ then
(4.5) set $(m_{i+3}, m_{i+2}) = (-1, 0)$ and $(m_{i+1}, m_i) = (2, 0)$;
(4.6) else if $((m_{i+3}, m_{i+2}) = (0, 1))$ then
(4.7) set $(m_{i+3}, m_{i+2}) = (1, 0)$ and $(m_{i+1}, m_i) = (-2, 0)$;
(4.8) else if $((m_{i+3}, m_{i+2}) = (-1, 0)$ then
(4.9) set $(m_{i+3}, m_{i+2}) = (0, -1)$ and $(m_{i+1}, m_i) = (-2, 0)$;
(4.10) else
(4.11) set $(m_{i+3}, m_{i+2}) = (0, 1)$ and $(m_{i+1}, m_i) = (2, 0)$;
(4.12) else
(4.13) $i = i + 2$;
(5) Pre-computation
$T[0, 1] = S; T[0, 2] = 2S; T[0, 4] = 4S; T[1, 0] = P$; $T[2, 0] = 2P; T[1, 1] = P + S; T[1, 2] = P + 2S$; $T[1, -1] = P - S; T[1, -2] = P - 2S; T[2, 1] = 2P + S$; $T[2, 2] = 2P + 2S; T[2, -1] = 2P - S$; $T[2, -2] = 2P - 2S$;
(6) $R = T[(k_{l-1}, k_{l-2}), (m_{l-1}, m_{l-2})]$;
(7) For i from $l - 2$ downto 0 $i = i - 2$ do
(7.1) $R = 2R$;
(7.2) $R = 2R$;
(7.3) if $((k_{i-1}, k_{i-2}) < 0)$ then
(7.4) $R = R - T[-(k_{i-1}, k_{i-2}), -(m_{i-1}, m_{i-2})]$;
(7.5) else if $((k_{i-1}, k_{i-2}) == 0)$ then
(7.6) if $((m_{i-1}, m_{i-2}) < 0)$ then
(7.7) $R = R - T[(k_{i-1}, k_{i-2}), -(m_{i-1}, m_{i-2})]$;
(7.8) else
(7.9) $R = R + T[(k_{i-1}, k_{i-2}), (m_{i-1}, m_{i-2})]$;
(7.10) else
(7.11) $R = R + T[(k_{i-1}, k_{i-2}), (m_{i-1}, m_{i-2})]$;
(8) Return (R).

Algorithm 6: New multi-scalar multiplication algorithm.

According to [23], one projective elliptic doubling in prime case needs 4S (S denotes module square) and 4M (M denotes module multiplication), and one projective elliptic addition on prime case needs 4S and 12M. We resume S ≈ 0.8M [24]. Then, algorithm in [13, 14, 16] needs 8S + 16M ≈ 22.4M per bit, and algorithm in [15] needs about 20.4M, while Algorithm 7 only needs 6S + 10M ≈ 14.8M per bit. Hence, it Algorithm 7 can improve performance by at least 25%

Input: P, k
Output: $Q = [k]P$
(1) Select a random r, $k1 = k - r$, $k2 = r$;
(2) Compute the NAF form of $(k1, k2)$;
(3) Add leading 0's in $(k1, k2)$ so that the number of bits in $(k1, k2)$ is multiple of 2, $k1 = (k1_{l-1}, k1_{l-2}, \ldots, k1_0)_{\text{NAF}}$, $k2 = (k2_{l-1}, k2_{l-2}, \ldots, k2_0)_{\text{NAF}}$;
(4) $i = 0$;
(5) While $(i < (l - 2))$ do
(5.1) set $c = (k1_{i+1}, k1_i) + (k2_{i+1}, k2_i)$;
(5.2) if $(c = 0)$ then
(5.3) if $((k2_{i+3}, k2_{i+2}) > 0)$ then
(5.4) set $(k2_{i+3}, k2_{i+2}) - = (0, 1)$ and $(k2_{i+1}, k2_i) = (-2, 0)$;
(5.5) else
(5.6) set $(k2_{i+3}, k2_{i+2}) + = (0, 1)$ and $(k2_{i+1}, k2_i) = (2, 0)$;
(5.7) else
(5.8) $i = i + 2$;
(6) Pre-computation
$T[1] = P; T[2] = 2P; T[3] = 3P; T[4] = 4P$;
(7) if $((k1_{l-1}, k1_{l-2}) + (k2_{l-1}, k2_{l-2}) = 0)$ then
$R = O$;
else
$R = T[(k1_{l-1}, k1_{l-2}) + (k2_{l-1}, k2_{l-2})]$;
(8) For i from $l - 2$ downto 0 $i = i - 2$ do
(8.1) $c = (k1_{i-1}, k1_{i-2}) + (k2_{i-1}, k2_{i-2})$;
(8.2) $R = 2R$;
(8.3) $R = 2R$;
(8.4) if $(c < 0)$ then
(8.5) $R = R - T[-c]$;
(8.6) else
(8.7) $R = R + T[c]$;
(9) Return (R).

Algorithm 7: DPA-resistant scalar multiplication algorithm.

compared with previous algorithms, and, sacrifices a small amount of memory to store pre-computation points.

5. Conclusion

In this paper, a simple but efficient elliptic scalar multiplication against power analysis attacks has been presented. First, we analyze previous algorithms which can resist SPA or DPA. Then, we present the new multi-scalar multiplication to endure SPA. This algorithm is based on NAF and processes two columns each loop. When computing scalar multiplication kP, we adopt a random number r to split scalar k and combine it with the multi-scalar multiplication. So, the new DPA-resistant scalar multiplication algorithm is proposed in this work. The proposed DPA-resistant scalar multiplication algorithm not only can resist SPA and DPA, but also provides good performance at a cost of only a few storage spaces.

References

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[2] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology: Proceedings of Crypto '85*, vol. 218 of *Lecture Notes in Computer Science*, pp. 417–426, Springer, Berlin, Germany, 1986.

[3] K. A. Shim and S. S. Woo, "Cryptanalysis of tripartite and multi-party authenticated key agreement protocols," *Information Sciences*, vol. 177, no. 4, pp. 1143–1151, 2007.

[4] L. Wang, Z. Cao, X. Li, and H. Qian, "Simulatability and security of certificateless threshold signatures," *Information Sciences*, vol. 177, no. 6, pp. 1382–1394, 2007.

[5] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," in *Advances in Cryptology (ASIACRYPT '98)*, vol. 1514 of *Lecture Notes in Computer Science*, pp. 51–65, Springer, Berlin, Germany, 1998.

[6] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer Professional Computing, Springer, New York, NY, USA, 2004.

[7] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other system," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '96)*, vol. 1109 of *Lecture Notes in Computer Science*, pp. 104–113, Springer, 1996.

[8] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '99)*, vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397, Springer, 1999.

[9] T. Izu and T. Takagi, "A fast parallel elliptic curve multiplication resistant against side channel attacks," in *Public Key Cryptography (PKC 2002)*, vol. 2274 of *Lecture Notes in Computer Science*, pp. 280–296, Springer, 2002.

[10] J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Proceedings of the 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES '99)*, vol. 1717 of *Lecture Notes in Computer Science*, pp. 292–302, Springer, 1999.

[11] E. Oswald and M. Aigner, "Randomized addition-subtraction chains as a countermeasure against power attacks," in *Proceedings of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES '01)*, vol. 2001 of *Lecture Notes in Computer Science*, pp. 39–50, Springer, 2001.

[12] B. Moller, "Securing elliptic curve point multiplication against side-channel attacks," in *Proceedings of the 4th International Information Security Conference (ISC '01)*, vol. 2200 of *Lecture Notes in Computer Science*, pp. 324–334, Springer, October 2001.

[13] M. K. Lee, "SPA-resistant simultaneous scalar multiplication," in *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA '05)*, vol. 3481, pp. 314–321, Singapore, May 2005.

[14] M. Ciet and M. Joye, "(virtually) Free randomization technique for elliptic curve cryptography," in *Proceedings of the 5th International Conference on Information and Communications Security (ICICS '03)*, vol. 2836, pp. 348–359, 2003.

[15] D. Liu and Z. Tan Y Dai, "New elliptic curve multi-scalar multiplication algorithm for a pair of integers to resist SPA," in *Proceedings of the 4th International Conference on Information Security and Cryptology (Inscrypt '08)*, vol. 5487 of *Lecture Notes in Computer Science*, pp. 253–264, Springer, December 2008.

[16] N. Zhang, Z. Chen, and G. Xiao, "Efficient elliptic curve scalar multiplication algorithms resistant to power analysis," *Information Sciences*, vol. 177, no. 10, pp. 2119–2129, 2007.

[17] P. Y. Liardet and N. P. Smart, "Preventing SPA/DPA in ECC systems using the Jacobi form," in *Proceedings of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES '01)*, vol. 2162 of *Lecture Notes in Computer Science*, pp. 391–401, Springer, May 2001.

[18] *Advances in Elliptic Curve Cryptography*, Cambridge University Press, Cambridge, UK, 2005.

[19] *ANSI X9.62:2005, Public Key Cryptography for the Financial Service Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American National Standards Institute, 2005.

[20] *ANSI X9.63:2001, Public Key Cryptography for the Financial Service Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography*, American National Standards Institute, 2001.

[21] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Fla, USA, 1997.

[22] J. Fan and I. Verbauwhede, "An Updated survey on secure ECC implementations: attacks, countermeasures and cost," in *Cryptography and Security: From Theory to Applications*, vol. 6805 of *Lecture Notes in Computer Science*, pp. 265–282, Springer, 2012.

[23] *IEEE Std 1363-2000, IEEE Standard Specifications for Public-Key Cryptography*, Institute of Electrical and Electronics Engineers, New York, NY, USA, 2000.

[24] C. H. Lim and H. S. Hwang, "Fast implementation of elliptic curve arithmetic in GF(P^n)," in *Proceedings of the 3rd International Workshop on Practice and Theory in Public Key Cryptosystem (PKC '00)*, vol. 1751 of *Lecture Notes in Computer Science*, pp. 405–421, Springer, January 2000.