Randomized Signed-Scalar Multiplication of ECC to Resist Power Attacks

Jae Cheol Ha[1] and Sang Jae Moon[2]

[1] Division of Information Science, Korea Nazarene Univ., Cheonan, Choongnam, 330-718, Korea
jcha@kornu.ac.kr
[2] School of Electronic and Electrical Engineering Kyungpook National Univ., Taegu, 702-701, Korea
sjmoon@knu.ac.kr

Abstract. Recently it has been shown that smart cards as cryptographic devices are vulnerable to power attacks if they have no defence against them. Randomization on ECC scalar multiplication is one of the fundamental concepts in methods of defence against side-channel attacks. In this paper by using the randomization concept together with the NAF recoding algorithm, we propose an efficient countermeasure for ECCs against power attacks. The countermeasure provides a randomized signed-scalar representation at every scalar multiplication to resist DPA. To protect against SPA it additionally employs a simple SPA-immune addition-subtraction multiplication algorithm. Our analysis shows that it needs no additional computation load compared to the ordinary binary scalar multiplication, where the average number of doublings plus additions for a bit length n is $1.5n+O(1)$.

Keywords: Elliptic curve cryptosystems, Side-channel attack, Power analysis attack, SPA, DPA, Non-adjacent form.

1 Introduction

The use of elliptic curve cryptosystems(ECC) was first proposed in 1985 by Miller [1] and Koblitz[2]. ECCs can use much smaller sizes of key bits, typically around 160 bits which provides the same security level as a 1024 bit RSA. In addition ECCs have better performance in computation speed than other multiplicative groups of RSA and ElGamal type encryptions at the same security level. These advantages make ECCs more attractive for cryptographic implementation on smart cards because limited memory and computation capability are available on them. Unfortunately cryptographic systems on smart cards are much vulnerable to side-channel attacks[3,4,5,6,7,8,9,10,11,12,13,14,15] such as fault attacks, timing attacks and power attacks. Since power attacks are known to be the most practical and powerful especially to cryptosystems on smart cards, in this paper we discuss in building a countermeasure against power attacks.

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 551–563, 2003.

Kocher et al.[7] originally presented power attacks of the simple and differential power analyses(referred to as SPA and DPA, respectively). In SPA, a single power signal trace of a cryptographic execution is measured and analyzed to classify point doublings or additions over the power trace. On the other hand, a DPA attacker usually measures hundreds of power signal traces and divides them into two groups by using a classification criterion, and makes a subtraction between the two averaged values. Since the averaging can reduce noisy components and result in the amplification of small power differences occurred in the execution of an algorithm, DPA is in general more powerful than SPA. Coron[8] introduced three key concepts to build countermeasures against power attacks; randomization of the private exponent, blinding the point and randomized projective coordinates.

More recently Oswald and Aigner[9] randomized the binary algorithm itself to resist power attacks. They inserted a random decision in the process of building the addition-subtraction chain which had been originally utilized for speeding up the ordinary binary scalar multiplication of an elliptic curve point[16]. The speeding-up chain method was first introduced in 1990 by Morain and Olivos[17].

The contribution of this paper is to propose a simple and powerful countermeasure for ECC scalar multiplication against power attacks. It uses the randomization concept together with non-adjacent form(NAF) algorithm[18,19] to change an ordinary binary multiplication representation to a form of signed-scalar one. The proposal provides an differently randomized signed-scalar representation at every scalar multiplication to resist DPA. To defeat SPA it employs a simple SPA-immune addition-subtraction multiplication algorithm. In addition the new countermeasure seems to be able to make timing attacks very difficult to work because every execution time of scalar multiplication depends on every different signed-scalar representation. The analysis shows that our proposal needs no additional computation load compared to the ordinary binary scalar multiplication, where the average number of doublings plus additions for a large n is $1.5n + O(1)$.

This paper is organized as follows. The ECC ordinary scalar multiplication and power attacks are briefly described in section 2. We explain the non-adjacent form(NAF) recoding algorithm in section 3, and finally present the new countermeasure with analyses and comparisons.

2 Elliptic Curve Cryptosystems and Power Attacks

An elliptic curve is a set of points (x, y) which are solutions of a bivariate cubic equation over a field K. An equation of the form

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

where $a_i \in K$, defines an elliptic curve over K. As an example, if char $K \neq 2$ and char $K \neq 3$, the above equation can be transformed to

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

with $a, b \in K$. This curve has one point O at infinity, which is the identity element of the group.

Let $P = (x_1, y_1) \neq O$ be a point, inverse of P is $-P = (x_1, -y_1)$. Let $Q = (x_2, y_2) \neq O$ be a second point with $Q \neq -P$, the sum $P + Q = (x_3, y_3)$ can be calculated as

$$x_3 = \lambda^2 - x_1 - x_2$$

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

with

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}\ , & if \quad P \neq Q \\ \frac{3x_1^2 + a_4}{2y_1}\ , & if \quad P = Q. \end{cases}$$

To subtract the point $P = (x, y)$, one adds the point $-P$.

2.1 Binary Scalar Multiplication

The operation of adding a point P to itself k times is called scalar multiplication by k and denoted $Q = kP$. We usually make use of the binary algorithm for the computation of the scalar multiplication $Q = kP$. The binary algorithm is described in the following figure 1. The binary algorithm is the analogue of the square-and-multiply method for exponentiation. For validity and explanation see [16].

`Binary algorithm`
`Input:` A point P, an n-bit integer $k = \sum_{i=0}^{n-1} k_i\, 2^i$, $k_i \in \{0, 1\}$
`Output:` $Q = kP$

1.1 $Q = O$
1.2 for $i = n - 1$ to 0 by -1 do {
1.3 $Q = 2Q$
1.4 if ($k_i == 1$) then $Q = Q + P$ }
1.5 Return Q

Fig. 1. Binary scalar multiplication algorithm

If the occurrence of 1's of a scalar integer k is assumed $\frac{1}{2}$ in probability, the number of doublings in the algorithm is equal to n, and the average number of additions $\frac{n}{2}$, where n is the number of bits of k.

2.2 Power Attacks

Power attacks of SPA and DPA were introduced in [7]. An SPA consists in observing the power consumption of one single execution of a cryptographic algorithm. In ECC computing of $Q = dP$ as shown in figure 1, it might be

possible to distinguish a point addition from a measured power signal trace, because the step 1.4 of figure 1 is conducted over the period of $k_i = 1$. Figure 2 shows a countermeasure[8] to the SPA, where the instructions conducted during a cryptographic algorithm do not depend on the data being processed.

Notice that the algorithm of figure 2 no longer defends against DPA. The step of 2.5 in the figure 2 has a small different amount of power consumption depending on whether the bit is 1 or 0. Even though it is very hard to find such a small difference by SPA, DPA is a powerful technique that exploits secret information by statistical analyzing power consumption.

SPA resistant Binary algorithm
Input: A point P, an n-bit integer $k = \sum_{i=0}^{n-1} k_i\, 2^i$, $k_i \in \{0, 1\}$
Output: $Q[0] = kP$

2.1 $Q[0] = O$
2.2 for $i = n - 1$ to 0 by -1 do {
2.3 $Q[0] = 2Q[0]$
2.4 $Q[1] = Q[0] + P$
2.5 $Q[0] = Q[k_i]$ }
2.6 Return $Q[0]$

Fig. 2. Binary scalar multiplication algorithm immune to SPA

In order to be resistant to DPA, some countermeasures have been proposed[8, 9,10,11,12,13,14,15]. Three countermeasures for ECC were first suggested by Coron[8] : randomization of the private exponent, blinding the point P and randomized projective coordinates. Specific countermeasures on a Koblitz curve[12], on a Montgomery-form elliptic curve[13], on a Jacobi-form elliptic curve[14], and on a Hessian-form elliptic curve[15] were also proposed.

More recently, Oswald-Aigner[9] have proposed randomized addition- subtraction chains for an elliptic curve scalar multiplication as a countermeasure against power attacks. It randomizes the binary algorithm by using addition-subtraction chains which had been proposed by Morain-Olivos[17].

Our difference from Oswald-Aigner's work is to use a randomization on the NAF algorithm to resist DPA, and employ a simple SPA-immune scheme. Even though the idea seems somewhat straightforward, the result turns out to be a simple and powerful countermeasure against power attacks. Moreover it needs no additional computation load compared to the ordinary binary algorithm, while the randomized addition-subtraction chains proposed by Oswald-Algner needs approximately 9% more additions than the ordinary binary algorithm.

3 Non-adjacent Form(NAF) Recoding Algorithm

Since subtraction has the same load as addition in the elliptic curve group, the NAF representation in the form of addition-subtraction chain can reduce the number of point operations in ECCs compared to the ordinary binary representation. A minimum discussion about the NAF to describe the randomized signed-scalar representation will be given here. For details see [16,18,19,20,21].

Consider an integer representation of the form $d = \sum_{i=0}^{n} d_i\, 2^i$, $d_i \in \{\bar{1}, 0, 1\}$ where $\bar{1} = -1$. We call it a binary signed-digit representation. A non-adjacent form (NAF) has the lowest weight among all signed-digit representations of a given k. Notice that every integer k has each unique NAF. The NAF recoding number d of scalar k can be constructed by the NAF recoding algorithm of Reitwiesner[18] given in table 1. First the auxiliary carry variable c_0 is set to 0. Reitwiesner's algorithm computes d starting from the LSB of k and scanning two bits at a time to the left. The i-th NAF recorded digit d_i and $(i+1)$-th value of the auxiliary binary carry c_{i+1} for $i = 0, 1, 2, \cdots, n$ are successively produced using table 1[19].

Even though the number of bits of k is equal to n, the number of bits in the d can be $n+1$, As an example, when $k = (11110)$ in the binary form, we compute the NAF recoding number d as follows.

$$k = (11110) = 2^4 + 2^3 + 2^2 + 2^1 = 30$$

$$d = (1000\bar{1}0) = 2^5 - 2^1 = 30$$

Given a NAF recoding number d, the addition-subtraction scalar multiplication algorithm is given in figure 3. The number of doubling operations required can be at most 1 more than that of the ordinary binary algorithm. On the other hand, the number of subsequent additions or subtractions is simply equal to the number of non-zero bits of the NAF recoding number d. The average number of additions (or subtractions) for the bit length n can be reduced $\frac{n}{3}$ [20,21].

Table 1. NAF recoding method

Input			Output	
k_{i+1}	k_i	c_i	c_{i+1}	d_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	$\bar{1}$
1	1	0	1	$\bar{1}$
1	1	1	1	0

`Addition-Subtraction algorithm`
`Input:` A point P, an $(n+1)$-bit integer $d = \sum_{i=0}^{n} d_i 2^i$, $d_i \in \{\bar{1}, 0, 1\}$
`Output:` $Q = dP$

3.1 $Q = O$
3.2 for $i = n$ to 0 by -1 do {
3.3 $Q = 2Q$
3.4 if ($d_i == 1$) then $Q = Q + P$
3.5 if ($d_i == \bar{1}$) then $Q = Q - P$ }
3.6 Return Q

Fig. 3. Addition-subtraction scalar multiplication algorithm

4 The New Countermeasure Based on Randomized Signed-Scalar Representation

4.1 Randomized Signed-Scalar Representation

To prevent DPA, we intend to randomize the ECC scalar multiplication procedure. The randomization results in a signed-scalar representation which is not in the form of NAF. The randomized signed-scalar recoding algorithm can be built by employing the concept used in the NAF recoding algorithm as follows.

Note that we use auxiliary carry c_{i+1} in the NAF recoding algorithm. The carry c_{i+1} means an $(i+1)$-th carry with $c_0 = 0$, and d_i is an i-th NAF digit. Therefore, the concatenated $c_{i+1}d_i$ has the value of $c_{i+1}2^1 + d_i 2^0$. Therefore, we can consider that the representation $c_{i+1}d_i = 01$ has another identical representation, i.e. $c_{i+1}d_i = 1\bar{1}$, or the reverse.

To insert randomness in the NAF recoding algorithm, we generate a random number $r = (r_{n-1}r_{n-2}\cdots r_0)$ which is an n-bits integer. In our countermeasure algorithm, the random recorded digit d_i and next value of the auxiliary binary variable c_{i+1} for $i = 0, 1, 2, \cdots, n$ can be sequentially generated as shown in table 2.

As an example, if we have $k_{i+1}k_ic_i = 001$ and random bit $r_i = 0$, we take NAF recoding $c_{i+1}d_i = 01$. For $r_i = 1$, then we can choose another recoding $c_{i+1}d_i = 1\bar{1}$. However, two different choices have the same value. In all cases except when $k_{i+1}k_ic_i = 001$, $k_{i+1}k_ic_i = 010$, $k_{i+1}k_ic_i = 101$ or $k_{i+1}k_ic_i = 110$ the random signed-scalar recoding method is always adapted independent of r_i.

As a practical example, when $k = (111011110)$ in binary form, we compute a random recoding number d with $r = (101010011)$.
$k = (111011110) = 2^8 + 2^7 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 = 478$
$c = (1111111000), \quad r = (101010011)$
$d = (1000\bar{1}00\bar{1}10) = 2^9 - 2^5 - 2^2 + 2^1 = 478$
With a random number $r = (110101001)$, we compute another recoding number d as follows.
$c = (1110111100), \quad r = (110101001)$
$d = (100\bar{1}1000\bar{1}0) = 2^9 - 2^6 + 2^5 - 2^1 = 478$

Table 2. Random singed-scalar recoding method

Input				Output		
k_{i+1}	k_i	c_i	r_i	c_{i+1}	d_i	Remarks
0	0	0	0	0	0	NAF
0	0	0	1	0	0	NAF
0	0	1	0	0	1	NAF
0	0	1	1	1	$\bar{1}$	AF
0	1	0	0	0	1	NAF
0	1	0	1	1	$\bar{1}$	AF
0	1	1	0	1	0	NAF
0	1	1	1	1	0	NAF
1	0	0	0	0	0	NAF
1	0	0	1	0	0	NAF
1	0	1	0	1	$\bar{1}$	NAF
1	0	1	1	0	1	AF
1	1	0	0	1	$\bar{1}$	NAF
1	1	0	1	0	1	AF
1	1	1	0	1	0	NAF
1	1	1	1	1	0	NAF

The hardware or software implementation of randomized signed-scalar representation is not difficult. In hardware, it additionally needs a random number generator and a 4-input/2-output logic circuit to implement the operations of table 2. This computational load is negligible.

4.2 Analysis and Comparisons

It is possible to find the probability of each symbol in the new random recoding algorithm for a given value of k. We assume that an n-bit binary number k is uniformly distributed in the range $[0,\ 2^n-1]$. Thus each bit of k can be generated a value of zero or one with equal probability, i.e. $P(k_i = 0) = P(k_i = 1) = \frac{1}{2}$ for all i. It is also supposed that the each bit probability of zero or one for a random number r is 1/2. The random signed-scalar numbers produced by the new algorithm can be considered a finite Markov chain model which is a similar analysis method to that employed in [19]. In this paper, the state variables are taken to be the quadruplets (k_{i+1}, k_i, c_i, r_i). There are 16 states for 4-bit combinations of input as given in table 3.

For example, consider input state s_2 which represents $(k_{i+1}, k_i, c_i, r_i) = (0, 0, 1, 0)$. The output (c_{i+1}, d_i) can be calculated as (0, 1) using the table 3 and the next state is $(k_{i+2}, k_{i+1}, c_{i+1}, r_{i+1}) = (k_{i+2}, 0, 0, r_{i+1})$. By assuming $P(k_{i+2} = 0) = P(k_{i+2} = 1) = \frac{1}{2}$ and $P(r_{i+1} = 0) = P(r_{i+1} = 1) = \frac{1}{2}$, there are 4 transitions with equal probability from state $s_2 = (0, 0, 1, 0)$ to the states $s_0 = (0, 0, 0, 0)$, $s_1 = (0, 0, 0, 1)$, $s_8 = (1, 0, 0, 0)$ and $s_9 = (1, 0, 0, 1)$.

Table 3. State transition table for the random singed-scalar recoding algorithm

State		Output	Next state			
			(k_{i+2}, r_{i+1})			
s_i	(k_{i+1}, k_i, c_i, r_i)	(c_{i+1}, d_i)	(0,0)	(0,1)	(1,0)	(1,1)
s_0	(0,0,0,0)	(0,0)	s_0	s_1	s_8	s_9
s_1	(0,0,0,1)	(0,0)	s_0	s_1	s_8	s_9
s_2	(0,0,1,0)	(0,1)	s_0	s_1	s_8	s_9
s_3	(0,0,1,1)	$(1,\bar{1})$	s_2	s_3	s_{10}	s_{11}
s_4	(0,1,0,0)	(0,1)	s_0	s_1	s_8	s_9
s_5	(0,1,0,1)	$(1,\bar{1})$	s_2	s_3	s_{10}	s_{11}
s_6	(0,1,1,0)	(1,0)	s_2	s_3	s_{10}	s_{11}
s_7	(0,1,1,1)	(1,0)	s_2	s_3	s_{10}	s_{11}
s_8	(1,0,0,0)	(0,0)	s_4	s_5	s_{12}	s_{13}
s_9	(1,0,0,1)	(0,0)	s_4	s_5	s_{12}	s_{13}
s_{10}	(1,0,1,0)	$(1,\bar{1})$	s_6	s_7	s_{14}	s_{15}
s_{11}	(1,0,1,1)	(0,1)	s_4	s_5	s_{12}	s_{13}
s_{12}	(1,1,0,0)	$(1,\bar{1})$	s_6	s_7	s_{14}	s_{15}
s_{13}	(1,1,0,1)	(0,1)	s_4	s_5	s_{12}	s_{13}
s_{14}	(1,1,1,0)	(1,0)	s_6	s_7	s_{14}	s_{15}
s_{15}	(1,1,1,1)	(1,0)	s_6	s_7	s_{14}	s_{15}

Let T_{ij} be the probability of moving from state s_i to state s_j. From the above example we find that $T_{20} = T_{21} = T_{28} = T_{29} = \frac{1}{4}$ and $T_{0j} = 0$ for $j = 2, 3, 4, 5, 6, 7,\ 10, 11, 12, 13, 14, 15$ using table 3. By computing probabilities T_{ij} for all i and j, we can draw the one-step transition probability matrix of the chain as follows.

$$T = \begin{pmatrix}
\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4}
\end{pmatrix}$$

Let π_i denote the limiting probability of state s_i. It can be found by solving linear equations $\pi T = \pi$ with $\pi_0 + \pi_1 + \cdots + \pi_{15} = 1$. The solutions of these equations are $\pi_i = \frac{1}{16}$ for $0 \leq i \leq 15$. Therefore the probability of each digit in a random signed-scalar number d is found by summing the limiting probability π_i of the state according to each output. As an example, the states for $d_i = 0$ are 8 states for $i = 0, 1, 6, 7, 8, 9, 14, 15$. By summing each limiting probability we get

$$P(d_i = 0) = \pi_0 + \pi_1 + \pi_6 + \pi_7 + \pi_8 + \pi_9 + \pi_{14} + \pi_{15} = \frac{1}{2},$$

$$P(d_i = 1) = \pi_2 + \pi_4 + \pi_{11} + \pi_{13} = \frac{1}{4},$$

$$P(d_i = \bar{1}) = \pi_3 + \pi_5 + \pi_{10} + \pi_{12} = \frac{1}{4}.$$

It is interesting to note that the average number of non-zero digits in the randomized signed-scalar number d is equal to $\frac{n}{2}$. Therefore the average number of additions (subtractions) required by the randomized signed-scalar recoding algorithm is $\frac{n}{2}$, which is the same as in the ordinary binary algorithm.

We consider now some possible power attacks. In DPA, suppose that the difference between doubling and addition (subtraction) operations is not distinguishable with a single power measurement. Every time the scalar multiplication is performed, it traces a different computational path due to randomization of the scalar such as that performed by our countermeasure algorithm. Therefore it makes the DPA attacks infeasible. In addition, the intermediate values to be attacked are computed at different times, or sometimes not even calculated. This make the DPA bias signal useless.

In the SPA case, it is assumed that the distinction between doubling and addition (subtraction) is visible with one power consumption measurement. It would be used to identify the correct secret key as in an ordinary binary representation. However, attackers can not distinguish addition from subtraction in our algorithm, which makes it difficult to identify the correct key.

Any possible weakness from the viewpoint of SPA can totally destroy DPA immunity countermeasure. This clearly shows the importance of developing a good SPA countermeasure. We here present a countermeasure against SPA, which is the SPA-immune addition-subtraction scalar algorithm given in figure 4. The proposed method is modified using the SPA resistant binary algorithm described in figure 2. The scheme in figure 4 also makes the power consumption independent of the secret digits. Consequently, the SPA countermeasure adopted the random signed-scalar recoding method makes the power attacks infeasible since random exponent d changes at each new operation of the scalar multiplication.

At this point we briefly compare the efficiency of the new algorithm with other algorithms that exist. The comparison of the number of operations is shown in table 4, which includes major operations(additions and doublings) without data copying or selection. Firstly, we compare our algorithm with the unprotected ordinary binary algorithm as each faces power attacks. The expected numbers

```
SPA resistant Addition-Subtraction algorithm
Input:  A point P, an (n+1)-bit integer d = Σ_{i=0}^{n} d_i 2^i, d_i ∈ {1̄, 0, 1}
Output: Q[0] = dP
```

4.1 $Q[0] = O$
4.2 $P[0] = P,\ P[1] = P,\ P[\bar{1}] = -P$
4.3 for $i = n$ to 0 by -1 do {
4.4 $\quad Q[0] = 2Q[0]$
4.5 $\quad Q[1] = Q[0] + P[d_i]$
4.6 $\quad Q[\bar{1}] = Q[1]$
4.7 $\quad Q[0] = Q[d_i]$ }
4.8 Return $Q[0]$

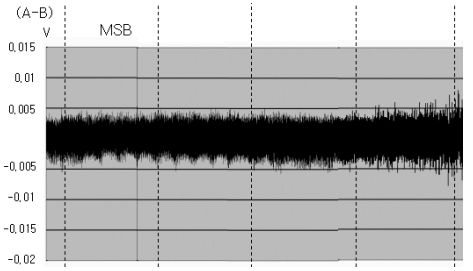
Fig. 4. SPA-immune addition-subtraction multiplication algorithm

of additions and doublings for an unprotected ordinary algorithm are $\frac{n}{2}$ and n respectively. As mentioned above, the NAF algorithm can reduce the additions (subtractions) to $\frac{n}{3}$ from $\frac{n}{2}$ of a ordinary binary algorithm. Our algorithm to resist DPA required $\frac{n}{2}$ additions and $(n+1)$ doublings which takes no extra time over using the unprotected ordinary binary algorithm. If this algorithm is compared with the unprotected NAF algorithm, it would be somewhat slower according to the above analysis. It is clear that the protected algorithms for SPA using the ordinary binary method required n additions and n doublings. Our countermeasure algorithm against DPA and SPA required almost exactly the same computation load.

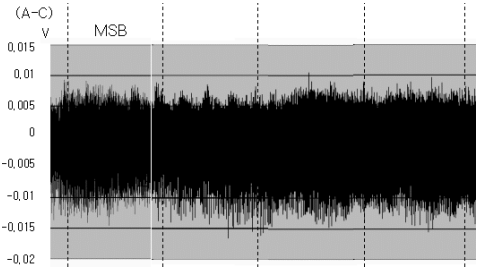
Table 4. Comparison of expected operations

Algorithm	Additions	Doublings	References
Unprotected ordinary binary	$\frac{n}{2}$	n	Fig. 1
Unprotected NAF	$\frac{n}{3}$	$n+1$	Fig. 3, Table 1
Protected ordinary binary against SPA	n	n	Fig. 2
Protected our algorithm against DPA	$\frac{n}{2}$	$n+1$	Fig. 3, Table 2
Protected our algorithm against DPA + SPA	$n+1$	$n+1$	Fig. 4, Table 2

Coren also proposed a modified binary scalar multiplication algorithm to resist SPA and three countermeasures against DPA[8]. We especially compare our countermeasure with the Coren's first DPA solution adopted SPA immuune algorithm as shown in figure 2. In his countermeasure, one needs to store an additional parameter the number of points $\#E(K)$, which is often not desirable. The main difference with our countermeasure is the bitlength of randomized key. If one selects a random number r of size m bits, in practice $m = 20$ bits, then his solution to resist DPA and SPA may increase computational operations up to $(n+m)$ additions and $(n+m)$ doublings.



(a) Difference between the correct one {1, 0, 0, X,···} and other scalar {1, 0, 1, X,···}



(b) Difference between the correct one {1, 0, 0, X,···} and the randomized signed-scalar representation

Fig. 5. The MESD attack with averaging over 300 traces for each scalar multiplication

4.3 Experimental Result

We experimentally applied the new countermeasure against a multiple-exponent, single-data (MESD) attack[11] for a simple test. The experimental result is shown in figure 5, where we assumed the attack was on the third digit {0} from the correct NAF scalar digits {1, 0, 0,···}. Since the step 4.4 and 4.5 of the multiplication algorithm procedure in figure 4 make a difference in power consumption, the averaged power peaks occur at the bit period right after the wrong guessing bit. The difference trace of (a) in figure 5 shows a lot of peaks over the period right after the third bit, where (a) is the difference between the averaged trace of the correct one {1, 0, 0, X,···} and a different representation {1, 0, 1, X,···}, and "don't care" is denoted by X. The occurrence of peaks implies that our guess is wrong in third bit. Therefore, the MESD attack is successful.

If we use the countermeasure, it generates a differently randomized signed-scalar representation at every multiplication execution. Due to the randomization, there are no peak appearances distinguishable between two power traces. It can make DPA including MESD infeasible. This is shown in (b) of figure 5.

5 Conclusion

The countermeasure provides a differently randomized signed-scalar representation at every multiplication execution so that it makes DPA infeasible. In addition it uses an addition-subtraction multiplication algorithm to interleave dummy operations to protect against SPA. It also seems to be able to defeat timing attacks because every execution time of a scalar multiplication changes according to every differently randomized signed-scalar representation. The structure of the countermeasure was analyzed using a finite Markov chain model. The result shows that it needs no additional computation load compared to the ordinary binary scalar multiplication, where the average number of doublings plus additions for a bit length n is $1.5n + O(1)$.

Acknowledgments. We would like to thank JongRok Kim and SungHyun Kim at the Samsung Electronics Corporation for their providing with experimental devices and helpful discussions. We are also grateful to the anonymous referees for their comments and suggestions.

References

1. V. S. Miller, "Use of elliptic curve in cryptography," In *Advances in Cryptology – CRYPTO '85*, LNCS 218, pp. 417–426, Springer-Verlag, 1986.
2. N. Koblitz, *Elliptic curve cryptosystems,* In Mathematics of Computation, vol. 48, pp. 203–209, 1987.
3. R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," In *Pre-proceedings of Security Protocol Workshop*, pp. 125–136, Springer-Verlag, April 1997.
4. Bell Press Release, "New threat model breaks crypto codes," Sept. 1996, available at URL `<http://www.bellcore.com/PRESS/ADVSRY96/facts.html>`.
5. D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," In *Advances in Cryptology – EUROCRYPT '97*, LNCS 1233, pp. 37–51, Springer-Verlag, 1997.
6. F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimbalu, and T. Ngair, "Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults," In *Pre-proceedings of Security Protocol Workshop*, pp. 115–124, Springer-Verlag, April 1997.
7. P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," In *Advances in Cryptology – CRYPTO '99*, LNCS 1666, pp. 388–397, Springer-Verlag, 1999.
8. J. S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," In *Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 292–302, Springer-Verlag, 1999.
9. E. Oswald and M. Aigner, "Randomized addition-subtraction chains as a countermeasure against power attacks," In *Cryptographic Hardware and Embedded Systems – CHES '01*, LNCS 2162, pp. 39–50, Springer-Verlag, 2001.
10. B. Möller, "Securing elliptic curve point multiplication against side-channel attacks," In *Information Security Conference – ISC'01*, LNCS 2200, pp. 324–334, 2001.

11. T.S. Messerges, E.A. Dabbish, and R.H. Sloan, "Power analysis attacks of modular exponentiation in smartcards," In *Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 144–157, Springer-Verlag, 1999.
12. M. A. Hasan, "Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz curve cryptosystems," In *Cryptographic Hardware and Embedded Systems – CHES '00*, LNCS 1965, pp. 93–108, Springer-Verlag, 2000.
13. K. Okeya and K. Sakurai, "Power analysis breaks elliptic curve cryptosystems even secure against the timing attack," In *Advances in Cryptology – INDOCRYPT '00*, LNCS 1977, pp. 178–190, Springer-Verlag, 2000.
14. P. Y. Liardet and N. P. Smart, "Preventing SPA/DPA in ECC systems using the Jacobi form," In *Cryptographic Hardware and Embedded Systems – CHES '01*, LNCS 2162, pp. 391–401, Springer-Verlag, 2001.
15. M. Joye and J. J. Quisquater, "Hessian elliptic curves and side-channel attacks," In *Cryptographic Hardware and Embedded Systems – CHES '01*, LNCS 2162, pp. 402–410, Springer-Verlag, 2001.
16. D. E. Knuth, *The art of computer programming*, vol 2: Seminumerical Algorithms, Reading, MA: Addison-Wesley, 2nd Edition, 1981.
17. F. Morain and J. Olivos, "Speeding up the computation on an elliptic curve using addition-subtraction chains," *Inform Theory Appl.*, vol. 24, pp. 531–543, 1990.
18. G. W. Reitwiesner, *Binary arithmetic*, Advances in Computers, 1:231–308, 1960.
19. O. Eğecioğlu and Ç. K. Koç, "Exponentiation using canonical recoding," *Theoretical Computer Science*, vol. 129, no. 2, pp. 407–417, 1994.
20. O. Eğecioğlu and Ç. K. Koç, "Fast modular exponentiation," In E. Arikan, editor, *Communication, Control, and Signal Processing : Proceedings of 1990 Bilkent International Conference on New Trends in Communication, Control, and Signal Processing*, pp. 188–194, Bilkent Univ. Ankara, Turkey, July 1990.
21. C. N. Zhang, "An improved binary algorithm for RSA," *Computer Math. Applic.*, vol. 25, no. 6, pp. 15–24, 1993.