Differential Power Analysis on Countermeasures Using Binary Signed Digit Representations

Tae Hyun Kim, Dong-Guk Han, Katsuyuki Okeya, and Jongin Lim

Side channel attacks are a very serious menace to embedded devices with cryptographic applications. To counteract such attacks many randomization techniques have been proposed. One efficient technique in elliptic curve cryptosystems randomizes addition chains with binary signed digit (BSD) representations of the secret key. However, when such countermeasures have been used alone, most of them have been broken by various simple power analysis attacks. In this paper, we consider combinations which can enhance the security of countermeasures using BSD representations by adding additional countermeasures. First, we propose several ways the improved countermeasures based on BSD representations can be attacked. In an actual statistical power analysis attack, the number of samples plays an important role. Therefore, we estimate the number of samples needed in the proposed attack.

Keywords: Elliptic curve cryptosystems, smart card, side channel attacks, binary signed digit representation, signal-to-noise ratio (SNR).

Manuscript received Aug. 18, 2006; revised Jan. 15, 2007.

This work was supported partly by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2006-(C1090-0603-0025)) and by the IT R&D program of MIC/IITA [2005-S-088-02, Development of security technology for secure RFID/USN service].

Tae Hyun Kim (phone: + 82 2 3290 4259, email: thkim@cist.korea.ac.kr) and Jongin Lim (email: jilim@korea.ac.kr) are with the Center for Information Security Technologies (CIST), Korea University, Seoul, Rep. of Korea.

Dong-Guk Han (phone: + 82 42 860 1784, email: christa@etri.re.kr) is with Information Security Research Division, ETRI, Daejeon, Rep. of Korea.

Katsuyuki Okeya (email: katsuyuki.okeya.ue@hitachi.com) is with Systems Development Laboratory, Hitachi, Ltd., Kawasaki, Japan.

I. Introduction

In consumer electronics, constrained devices, such as smart cards, mobile phones, and handheld computers can be used for efficient and secure communications. Therefore, the security of electronic applications in commercial and financial areas is very important for consumers and businesses. Cryptographic algorithms on such devices should be optimized in terms of high performance, low cost, low power, and low energy because of scarce resources. Above all, elliptic curve cryptosystems (ECCs) [1], [2] are suitable to implement on such devices because they achieve the same level of security as a finite-field-based cryptosystem with much shorter key size For example, a 160-bit elliptical curve digital signature algorithm (ECDSA) has almost the same security level as a 1,024-bit DSA.

In a cryptosystem implemented in an embedded device, cryptographic operations leak sensitive information related with the secret key stored in the device [3] via a side channel. The information includes computation timing, power consumption, electromagnetic radiation, and sound leaked during the execution of the cryptographic algorithms. The goal of most countermeasures against side channel attacks (SCAs) is to minimize the impact of the side channel information by introducing randomization techniques. For example, the paths taken by cryptographic algorithms could be randomized by a redundant representation of the secret key. We define a binary signed digit representation as a redundant representation.

Definition 1. For an integer k, the binary representation of k is represented by the set of digits $\{0, 1\}$:

$$k=(k_{n-1}\cdots\ k_0)_2, \qquad \text{where} \quad k_i\in\{0, 1\}.$$

A binary signed digit (BSD) representation of k is represented by the set of digits {-1, 0, 1}:

$$d=(d_n\cdots d_0)_2, \qquad \text{where } d_i\in\{-1, 0, 1\}.$$

In ECCs, a popular type of countermeasure against SCAs inserts random decisions when a BSD representation is chosen from the secret key. Especially, since the computation of the inverse of a point $P=(x, y)$ on an elliptic curve is negligible, this type of countermeasure on ECCs provides us with good performance/efficiency and security.

Definition 2. In elliptic curve scalar multiplication of ECCs, a method to provide protection against SCAs using a BSD representation of the secret key is called a BSD countermeasure.

For instance, this type of countermeasure[1] includes the Ebeid-Hasan countermeasure [4], the Ha-Moon countermeasure [5], the Oswald-Aigner countermeasure [6], and the countermeasure by Agagliate and others [7].

1. Previous Works

In [4] and [8], Ebeid and Hasan investigated the randomness of their own countermeasure as well as that of the Ha-Moon and Oswald-Aigner countermeasures. They showed that those BSD countermeasures can generate many binary signed digit representations from one binary secret key. This makes the secret key unpredictable in the context of side channel attacks. The prime purpose of BSD type countermeasures is to protect against differential power analysis (DPA) attacks.

However, most of BSD countermeasures have been broken by sophisticated simple power analysis (SPA) attacks under the assumption of the ability to distinguish between elliptic curve point addition and elliptic curve point doubling if only a BSD countermeasure is used against SCAs. For example, Okeya and Sakurai [9] and Walter [10] broke the simple and complex versions of the Oswald-Aigner countermeasure, respectively. Okeya and Han proposed an attack on the basic version of the Ha-Moon countermeasure [11]. In [12], Karlof and Wagner proposed an attack using the hidden Markov model, which is a cryptanalytic framework for countermeasures that utilize a probabilistic finite state machine. In [13], Han and others showed an invariant property of BSD representations.

Definition 3. An original BSD countermeasure can be combined with additional SPA or DPA countermeasures to improve the security of the BSD countermeasure. The combination is called an improved BSD countermeasure.

Since the previously mentioned attacks are based on SPA, protecting BSD countermeasures from them can simply be done by combining with an SPA countermeasure. Thus, the improved countermeasures combined with an SPA countermeasure may be secure against both SPA and DPA attacks.

2. Contribution of this Paper

In this paper, we will show that the combination of a BSD countermeasure with SPA countermeasures is vulnerable to a DPA attack which utilizes some statistical tools to reduce the unwanted noise. The effect of the statistical approach depends on the number of samples required in a DPA attack. Therefore, the role of the number of samples used in a DPA attack is very important. We will show how many samples are required in the proposed attack to obtain the same height as the peak of the DPA attack on naïve algorithms without any countermeasure. To estimate the number of samples, we use the signal-to-noise ratio (SNR) model introd uced by Messerges [14].

To further strengthen the security of BSD countermeasures, we will show that even when they are combined with a DPA countermeasure it does not provide sufficient security.

The previously described attacks are achieved on the computation from the most significant bit (MSB) to the least significant bit (LSB) of the secret key, that is, when the bits of the secret key are scanned from left to right, denoted by L-to-R. However, countermeasures such as the Oswald-Aigner countermeasure may utilize the computation from the LSB to the MSB of the secret key, in which the bits of the secret key are scanned from right to left, denoted by R-to-L. In this paper, we will investigate the security of both L-to-R and R-to-L computation.

This paper is organized as follows. In the next section, we briefly describe the principle of elliptic curve operations. In section III, we introduce side channel attacks and countermeasures on ECCs. In section IV, we propose a DPA attack on BSD countermeasures combined with some SPA countermeasure and show simulation results. In section V, we apply the proposed attack to R-to-L computation. In section VI, we extend the proposed attack to the refined power analysis (RPA) to attack BSD type countermeasures combined with some DPA countermeasure. In section VII, we investigate secure combinations between BSD and SPA or DPA countermeasures. In section VIII, we show a comparison with previous attacks against improved BSD countermeasures. Finally, we conclude in section IX.

II. Elliptic Curve Arithmetic

Let $K=F_q$ be the finite field with q elements where q is a power of a prime $p>3$. Elliptic curves over K can be

1) Note that since the BSD representations use the set of digits {-1,0,1}, in this paper, we do not deal with countermeasures based on window methods using randomized addition chains.

represented by

$$E(K) := \{(x, y) \in K \times K \mid y^2 = x^3 + ax + b\} \cup O$$

with $a, b \in K$, $4a^3 + 27b^2 \neq 0$, where O denotes the point at infinity, that is, the identity element of an elliptic curve group. Every elliptic curve is isomorphic to a curve of this form, and we call it the Weierstrass form. An elliptic curve $E(K)$ has an additive group structure. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two elements of $E(K)$ that are different from O. We have $-P_1 = (x_1, -y_1)$. If $-P_1 \neq P_2$, then the addition, $P_1 + P_2 = (x_3, y_3)$ is given with

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

where $\lambda = (y_2 - y_1)/(x_2 - x_1)$ for $P_1 \neq P_2$, and $\lambda = (3x_1^2 + a)/(2y_1)$ for $P_1 = P_2$.

If $P_1 \neq P_2$, it is called an elliptic curve addition (ECADD); otherwise, if $P_1 = P_2$, it is called an elliptic curve doubling (ECDBL) The computational costs of ECADD and ECDBL are usually different.

Let k be a positive integer, and let P and Q be points on an elliptic curve $E(K)$. The ECCs are based on the elliptic curve discrete logarithm problem (ECDLP), which is to find k from P and $Q = kP$. While ECDLP is a hard mathematical problem, computing $kP = P + \cdots + P$ from k and P is easy, where k is often secret. This operation is called a scalar multiplication. A standard method for computing the scalar multiplication is the double-and-add binary method. We show how to compute kP with this method. Let $(k_{n-1} \cdots k_1 k_0)_2$ be the binary representation of k, where $k_i = 0$ or 1. Therefore,

$$kP = k_{n-1} 2^{n-1} P + \cdots + k_1 2P + k_0 P \tag{1}$$

$$= 2(2(\cdots 2(2k_{n-1}P + k_{n-2}P) + \cdots) + k_1 P) + k_0 P. \tag{2}$$

From the above two equations, the binary scalar multiplication can be computed in two ways. The algorithm based on (1) scans the bits of the scalar k from LSB to MSB, or R-to-L. The other algorithm based on (2) scans the bits of the scalar k from MSB to LSB, or L-to-R.

Algorithm 1. L-to-R Binary Algorithm.
Input: A point P and $k = k_{n-1}2^{n-1} + \cdots + k_1 2 + k_0$, $k_i \in \{0,1\}$.
Output: $Q = kP$
1. $Q[0] = O$
2. For $j = n-1$ down to 0
   2.1. $Q[0] = \text{ECDBL}(Q[0])$
   2.2. if $k_j = 1$ then $Q[0] = \text{ECADD}(Q[0], P)$
3. Return $Q[0]$

Algorithm 2. R-to-L Binary Algorithm.
Input: A point P and $k = k_{n-1}2^{n-1} + \cdots + k_1 2 + k_0$, $k_i \in \{0,1\}$.
Output: $Q = kP$
1. $Q[0] = O$, $Q[1] = P$,
2. For $j = 0$ to $n-1$
   2.1. if $k_j = 1$ then $Q[0] = \text{ECADD}(Q[0], Q[1])$
   2.2. $Q[1] = \text{ECDBL}(Q[1])$
3. Return $Q[0]$

Algorithm 2 requires one additional arbitrary storage, namely, $Q[1]$ over algorithm 1. Moreover, it is not easy to extend algorithm 2 to window methods; therefore, algorithm 1 is generally preferred to algorithm 2.

Since the number of non-zero bits in the binary representation is on average $n/2$, the computational cost of the above algorithms is n ECDBLs and $n/2$ ECADDs. Since the computation of the inverse of a point P is negligible, we can reduce the number of ECADDs by using a BSD representation. The main idea of the conversion is to replace s consecutive 1's in the binary representation as follows:

$$1^s \rightarrow 10^{s-1}\bar{1}, \text{ where } \bar{1} = -1.$$

For example, the non-adjacent form (NAF) is one of many BSD representations, where non-zero digits cannot be adjacent. Note that every integer has several different binary signed representations, but the representation of NAF is unique. In the next section, we describe how to randomize the binary representation using signed digits.

III. Side Channel Attacks

There are two types of SCAs: active and passive attacks. Active attacks try to tamper with the internal circuitry of cryptographic devices. For example, fault-induction attacks will try to induce errors during computation. Passive attacks simply observe the behavior of devices during processing without disturbance. Passive attacks use information received through side channels, such as computation timing, power consumption, electromagnetic radiation, and sound coming out of the device.

Among passive attacks, power analysis attacks, which include SPA and DPA attacks, are among the most effective types of SCAs. An SPA attack directly interprets power consumption measurements collected during cryptographic operations. A DPA attack is very powerful in comparison to an SPA attack because a DPA attack analyzes the correlation between power consumption and specific key-dependent intermediate values which appear during computation with the secret key by using statistical tools and error correction techniques. Therefore, defense against DPA attacks may be much more difficult than SPA attacks.

1. SPA and Countermeasure

In this section, we describe an SPA attack on the binary algorithm for computing the scalar multiplication introduced in section II. Since the computational costs of ECADD and ECDBL are usually not equal, the power consumption traces of ECDBL and ECADD are not equal either. Therefore, we can easily distinguish these operations in a power consumption trace, and then recover the secret key k by corresponding DA into 1 and D into 0, where D and A denote the power consumption of ECDBL and ECADD, respectively.

One possible way to achieve resistance against SPA attacks is to insert dummy ECADDs during the standard binary method, first introduced by Coron [15]. Therefore, an operation sequence converted from power consumption is always $DA\cdots DA$ regardless of the secret key. An SPA-immune version of algorithm 1 is the following.

Algorithm 3. Double-and-Add-Always Algorithm.
Input: A point P and $k=k_{n-1}2^{n-1}+\cdots+k_1 2+k_0$, $k_i \in \{0,1\}$.
Output: $Q=kP$
1. $Q[0]=P$
2. For $j=n$-2 down to 0
   2.1. $Q[0]=\text{ECDBL}(Q[0])$
   2.2. $Q[1]=\text{ECADD}(Q[0],P)$
   2.3. $Q[0]=Q[k_i]$
3. Return $Q[0]$

Another option is to use the same formula for ECDBL and ECADD. Therefore, an operation sequence converted from power consumption is always $AA\cdots AA$ regardless of the secret key, where A is an operation of ECADD or ECDBL by the same formula.

The difference between countermeasures using a dummy operation and the indistinguishable operation is whether the length of operation sequences generated during a scalar multiplication is always fixed independent of the secret key. This characteristic affects the success of a DPA attack.

2. DPA and Countermeasure

Before a description of DPA, we first introduce power leakage models to construct a successful DPA attack.

A. Power Leakage Models

Messerges [14] proposed a power leakage model whose leaked information depends on the Hamming weight of the data being processed and confirmed that the model is valid under smart cards. Let the power consumption W be expressed as $W=\varepsilon \cdot H+L+N$, where H, ε, L, and N respectively represent the Hamming weight of the intermediate data result, the incremental amount of power for each extra 1 in the Hamming weight, the additive constant portion of the total power, and the noise. Note that noise N is assumed to be independent and have a zero mean.

Brier and others [16] introduced the Hamming distance model which is a general model of the Hamming weight model. The Hamming distance between D and R is defined by the number of flipping bits to go from R to D and denoted by $H(D \oplus R)$, where D is data and R is a reference state. They then modeled a linear relationship between the current consumption and $H(D \oplus R)$. It does not represent the entire consumption of a chip but only the data dependent part. All the remaining aspects of the power consumption of a chip are assigned to a term denoted as b, which is assumed to be independent from the other variables: b includes offsets, time dependency components, and noise. Therefore, the basic model for the data dependency can be written as

$$W = aH(D \oplus R)+b,$$

where a is a scalar gain between the Hamming distance and W, the power consumed.

We can achieve the same results under the Hamming distance model, but for simplicity we use the Hamming weight model.

B. DPA on Double-and-Add-Always Algorithm

In this section, we describe a DPA attack on algorithm 3 under the power leakage model even though it is secure against SPA attacks. The standard DPA attack is achieved by the following five steps.

Step 1. Measure power consumption.
Step 2. Guess a part of the secret key.
Step 3. Calculate hypothetical intermediate values.
Step 4. Split power consumption into two sets by a partitioning function.
Step 5. Subtract the averages of the two sets.

Suppose an attacker already knows the highest bits $(k_{n-1}\cdots k_{i+1})_2$ of the secret key k. The attacker's goal is to recover the next bit k_i. Without loss of generality, assume $k_i=0$. For $1\le r\le R$, the attacker inputs distinct points P_r into the device which uses algorithm 3 to compute $Q=kP_r$, and then measures the power consumption associated with the computation for kP_r. The power consumption at time t is denoted by $W_r(t)$. Since a hypothetical intermediate value calculated at step 2.1 after $j=i$ (the computation for the i-th bit of k) in algorithm 3 is $(k_{n-1}2^{n-i}+\cdots+k_{i+2}2^3 + k_{i+1}2^2)\cdot P$,[2] the attacker can divide $W_r(t)$ into two groups, S_0 and S_1, by a partitioning function, D(*):

$$S_0 = \{W_r(t) \mid D(*) = 0\},$$
$$S_1 = \{W_r(t) \mid D(*) = 1\},$$

where $|S_0| + |S_1| = R$. Let $D(*)$ be a Boolean selection function,

2) If the hypothesis is $k_i=1$ then the hypothetical intermediate value is $(k_{n-1}2^{n-i}+\cdots+k_{i+2}2^3 + k_{i+1}2^2+2)\cdot P$.

such that $D(*)=0$ if a specific bit of the hypothetical intermediate values is 0, and $D(*)=1$ if not. Then, the attacker can compute the differential power consumption $g(t)$ as

$$g(t)=\frac{1}{|S_1|}\sum_{W_r(t)\in S_1}^{|S_1|} W_r(t)-\frac{1}{|S_0|}\sum_{W_r(t)\in S_0}^{|S_0|} W_r(t).$$

If the hypothesis is correct, then the average power consumptions of S_0 and S_1 are, respectively, $\varepsilon(n\text{-}1)/2+L$ and $\varepsilon(n+1)/2+L$ by the Hamming weight model and the assumption that the noise has a zero mean. Finally, the attacker can recognize some peaks with amplitude ε in the differential power trace when the hypothetical intermediate value is manipulated. We call a peak DPA a bias signal. Otherwise, if the hypothesis is wrong, we cannot see any bias signal since the classification into S_0 and S_1 is meaningless. Once k_i is known, the remaining bits $(k_{i-1}\cdots k_0)_2$ are recursively recovered as in steps 2 to 5.

C. Hypothetical Intermediate Value

The success of the DPA attack depends on how the relation between hypothetical values and real power consumption is detected. Therefore, it is important that the attacker chooses the hypothetical intermediate value corresponding to a hypothesis for the secret key in the DPA attack. First, the attacker classifies power consumption into two classes according to a part of the hypothetical intermediate value. Second, the attacker analyzes side channel information, and checks whether the hypothetical intermediate value appears. Finally, the attacker reveals a portion of the secret key using the (dis)appearance of the hypothetical intermediate value.

In the DPA attack of section III.2.B, the hypothetical intermediate values corresponding to the hypothesis $k_i=0$ and $k_i=1$ are $(k_{n-1}2^{n-i}+\cdots+k_{i+2}2^3+k_{i+1}2^2)\cdot P$ and $(k_{n-1}2^{n-i}+\cdots+k_{i+2}2^3+k_{i+1}2^2+2)\cdot P$, respectively. The hypothetical intermediate values are not values chosen by an attacker. The attacker can only guess or predict hypothetical intermediate values using known plaintexts. Thus, the scenario of this DPA attack is the known plaintext attack.

D. Signal-to-Noise Ratio

For a DPA attack to be successful, the attacker must be able to detect the bias signal over the noise in practice. To reduce the unwanted noise in the power consumption, Messerges introduced filtering strategies [14] and proposed a model for the DPA signal-to-noise ratio. We use the concept of signal-to-noise ratio (SNR) to estimate the required number of samples.

Proposition 1 [14]. A DPA attack using R samples on an M-bit processor in algorithm 3 with signal size ε, average non-algorithmic noise variance σ^2, and percentage of algorithmic noise α, has a voltage intrasignal SNR that can be modeled by

$$SNR=\frac{\varepsilon\sqrt{R}}{\sqrt{8\sigma^2+\varepsilon^2(\alpha M+M-1)}}. \quad (3)$$

3. BSD Countermeasure

We now describe the basic idea of the BSD countermeasure, which is essentially proposed to counteract both SPA and DPA attacks. Considering the binary representation of an integer k as one of its BSD representations, different BSD representations for k can be obtained by replacing 01 with $1\bar{1}$ and vice versa and by replacing $0\,\bar{1}$ with $\bar{1}\,1$ and vice versa. For example, if $k=(01101)_2$, the different BSD representations for k are 01101, $011\bar{1}1$, $10\bar{1}01$, $1\bar{1}101$, and so forth. Therefore, it was believed to provide enough security against both SPA and DPA attacks if a BSD representation was randomly selected among many BSD representations. The lack of distinction between additions and subtractions provides resistance against SPA attacks, even though a distinction between doublings and additions/subtractions is possible. A different path by a randomized representation of the secret key provides resistance against DPA attacks because a target hypothetical intermediate value processed is not synchronized. Thus, desynchronization causes the DPA bias signal to be widely distributed at different times [17].

IV. New DPA Attacks on BSD Countermeasures Combined with SPA Countermeasures

In this section, we describe a DPA attack on the BSD countermeasure combined with SPA countermeasures. We can classify SPA countermeasures into two types according to synchronization or desynchronization of the target hypothetical intermediate value. One includes Coron's dummy method [15] and Montgomery ladder methods[3] [18], [19]. The other includes the unified code [20] and the side channel atomic blocks method [21]. Since the effect of desynchronization splits the amplitude of a DPA bias signal, the former is more vulnerable to DPA attacks than the latter. In this section, we describe an attack on the former.

Before describing the proposed DPA attack, we first introduce notations and assumptions.

1. Notations

R: The number of executions that an attacker observes to

3) The Montgomery ladder [22] was originally proposed to speed up scalar multiplication in the context of elliptic curves. It has been extended to different settings, such as countermeasures against SCAs.

measure power consumption, that is, the number of samples.

R^S: The smallest number of executions to detect the bias signal over the noise in SNR formula (3) against algorithm 3. If R is chosen as $R \geq R^S$, then an attacker can break algorithm 3.

R^M: The maximum number of executions available in a DPA attack for an attacker. Note that R^M depends on the computational power and ability of an attacker. By definition, $R^M \geq R$.

P_r: For $1 \leq r \leq R$, the r-th point into the smart card using a target scalar multiplication algorithm.

F_r^i: The real intermediate value which is actually calculated at the step of ECDBL after $j=i$ (the computation for the i-th bit of d) in an algorithm using BSD representations for the r-th input point P_r.

H_r^i: A hypothetical intermediate value corresponding to the attacker's guess for the i-th bit of the secret key k_i and the r-th input point P_r.

p: The appearance probability of a hypothetical intermediate value. For a large R, $p \approx \Pr[H_r^i = F_r^i]_{1 \leq r \leq R}$.

2. Assumptions

Let $k = k_{n-1}2^{n-1} + \cdots + k_1 2 + k_0$, be the n-bit binary secret key where $k_i = 0$ or 1, and let $d = d_n 2^n + \cdots + d_1 2 + d_0$ be the $(n+1)$-bit random recoded number generated from k by a random recoding method, where $d_i = 0$ or ± 1. Note that k and d are obviously the same number, but they have different representations. Let $k_{[s,t]}$ and $d_{[s,t]}$ denote partial segments from the s-th bit to the t-th bit of k and d, respectively. Namely,

$$k_{[s,t]} := k_t 2^{t-s} + \cdots + k_{s+1} 2 + k_s, \text{ where } 0 \leq s \leq t \leq n-1,$$

$$d_{[s,t]} := d_t 2^{t-s} + \cdots + d_{s+1} 2 + d_s, \text{ where } 0 \leq s \leq t \leq n.$$

In algorithm 3, the appearance probability for all hypothetical intermediate values is either 0 or 1. Note, however, that the appearance probability for a hypothetical intermediate value when using BSD representations depends on the secret key and a random recoding method used in BSD type countermeasures. Thus, it is not a fixed and predictable value.

Suppose an attacker already knows the highest bits $(k_{n-1} \cdots k_{i+1})_2$ of the secret key k. The attacker's goal is to recover the next bit k_i. We make some more precise assumptions to describe the framework of our attack.

Assumption 1. We assume that the scalar multiplication is performed by L-to-R computation using the fixed secret key k but for different representations and different inputs P.

For some elliptic curve schemes like ECDSA, two-pass ECDH, or two-pass ECMQV [23], collecting multiple power consumption curves for the fixed secret key may be impossible. However, for some other schemes like ECIES, single-pass ECDH, or single-pass ECMQV [23], it is possible[4]. Therefore, the assumption is reasonable. We can repeatedly measure power consumption under assumption 1.

3. Properties of BSD Representations

We justify that the following property holds in all BSD countermeasures. That is, the consequence of the following proposition means that it does not depend on recoding techniques.

Proposition 2. $d_{[i,n]}$ is either $k_{[i,n-1]}$ or $k_{[i,n-1]}+1$.

Proof. $d = d_{[i,n]}2^i + d_{[0,i-1]}$ and $k = k_{[i,n-1]}2^i + k_{[0,i-1]}$. As $d=k$, $(d_{[i,n]} - k_{[i,n-1]})2^i = k_{[0,i-1]} - d_{[0,i-1]}$. As $-2^i < k_{[0,i-1]} - d_{[0,i-1]} < 2^{i+1}$, $-1 < (k_{[0,i-1]} - d_{[0,i-1]})/2^i < 2$. Here, $(k_{[0,i-1]} - d_{[0,i-1]})/2^i$ must be an integer since it is equal to $d_{[i,n]} - k_{[i,n-1]}$. Hence, $d_{[i,n]}$ is either $k_{[i,n-1]}$ or $k_{[i,n-1]}+1$. □

From proposition 2, we can obtain a relation between the i-th bit k_i of the secret key and the corresponding intermediate value (F_r^i).

Fact 1.

If $k_i=0$, then F_r^i is either $(2^2 k_{[i+1,n-1]})P_r$ or $(2^2 k_{[i+1,n-1]}+2)P_r$.

If $k_i=1$, then F_r^i is either $(2^2 k_{[i+1,n-1]}+2)P_r$ or $(2^2 k_{[i+1,n-1]}+4)P_r$.

There are only three intermediate values. The intermediate value $F_r^i = (2^2 k_{[i+1,n-1]})P_r$ only appears when $k_i=0$, and $F_r^i = (2^2 k_{[i+1,n-1]} + 4)P_r$ only appears when $k_i=1$. However, $F_r^i = (2^2 k_{[i+1,n-1]} + 2)P_r$ is related to both $k_i=0$ and $k_i=1$. The above relation helps us to predict a hypothetical intermediate value in the scalar multiplication algorithm using BSD representations. For example, the hypothetical intermediate value corresponding to the hypothesis for $k_i=0$ may be one of $(2^2 k_{[i+1,n-1]})P_r$ and $(2^2 k_{[i+1,n-1]} + 2)P_r$. The hypothetical intermediate value for recovering k_i should be $(2^2 k_{[i+1,n-1]})P_r$, since $(2^2 k_{[i+1,n-1]} + 2)P_r$ can appear regardless of $k_i=0$ or 1. Note that a hypothetical intermediate value is not chosen by an attacker; rather, it is computable or predictable using the known input point P_r and the previously known key bits $(k_{n-1} \cdots k_{i+1})_2$.

4. DPA on BSD type Countermeasure with Fixed Path

In this section, we show that BSD countermeasures with a fixed path such that the target hypothetical intermediate value is manipulated at the fixed time as in Coron's dummy method and Montgomery ladder methods are vulnerable to DPA attack.

For simplicity, we mainly deal with Coron's dummy method as an SPA countermeasure, but the proposed DPA attack can be

4) In both single-pass ECDH and single-pass ECMQV, one of the ephemeral DH/MQV keys is a long term static public key. This is kept constant.

easily applied to Montgomery ladder methods. The following addition-subtraction algorithm is an example of an SPA countermeasure using a dummy method.

Algorithm 4. L-to-R-Addition-Subtraction Algorithm.
Input: A point P and $k=k_{n-1}2^{n-1}+\cdots+k_1 2+k_0$, $k_i \in \{0,1\}$.
Output: $Q=kP$
1. Convert k to $d=d_n 2^n+\cdots+d_0$, where $d_i \in \{\pm 1,0\}$
2. $Q[0]=O$, $R[0]=P$, $R[1]=P$, $R[2]=-P$
3. For $j=n$ down to 0
   3.1. $Q[0]=\mathrm{ECDBL}(Q[0])$
   3.2. $Q[1]=\mathrm{ECADD}(Q[0],R[1-d_j])$
   3.3. $Q[0]=Q[|d_j|]$
4. Return $Q[0]$

In the original Montgomery ladder methods, if the i-th intermediate pair is $(iP, (i+1)P)$, the next intermediate pair is updated by $(2iP, (2i+1)P)$ for $k_{i-1}=0$ and $((2i+1)P, 2(i+1)P)$ for $k_{i-1}=1$. Therefore, the Montgomery ladder methods cannot be directly combined with BSD representations. However, if we use the triple $((i-1)P, iP, (i+1)P)$ instead of $(iP, (i+1)P)$, the modified Montgomery ladder method can work with BSD representations.

A. SNR for Probabilistic Appearance of Hypothetical Intermediate Value

Since the role of the number of samples used in DPA attacks is very important in practice, we estimate the number of samples required in the DPA attack on a BSD countermeasure with a fixed path to obtain the same height as the amplitude of the DPA bias signal on unprotected algorithms using the SNR model.

As algorithm 4 uses a randomized BSD representation, the amplitude of DPA bias signals may decrease due to the probabilistic appearance of hypothetical intermediate values. Therefore, the SNR model for the BSD countermeasure should be modified as follows.

Proposition 3. Assume that the computational environment is the same as proposition 1. If the appearance probability of a hypothetical intermediate value is p, then the SNR is

$$SNR=\frac{\varepsilon p\sqrt{R}}{\sqrt{8\sigma^2+\varepsilon^2(\alpha M+M-p)}}. \quad (4)$$

The SNR is approximately p times larger than the original. In other words, to obtain the same SNR, the required samples are $1/p^2$ times larger than the original.

Proof. Recall that SNR is defined as the ratio of the average signal divided by its standard deviation. If a hypothetical intermediate value appears with the probability p, then the DPA bias signal is p times larger than the original; $E[S]=\varepsilon p$, where S stands for the bias signals. Non-algorithmic noise does not depend on the input data, that is, $4\sigma^2/R$, whereas algorithmic noise can be seen as $(M\text{-}S/\varepsilon)$ random bits, namely, $p(M\text{-}1)+(1\text{-}p)M=(M\text{-}p)$ random bits. Thus, it is easy to see that the variance of the signal is $V[S]=4(\sigma^2+\varepsilon^2(M\text{-}p)/4)/R$. If the hypothetical intermediate value does not appear, we can consider such a case as noise; $E[N]=0$, and the variance of the noise is $V[N]=4(\sigma^2+\varepsilon^2\alpha M/4)/R$. Hence, in the current case, the average is εp, and the standard deviation for the sum of two variances is

$$\sqrt{8\sigma^2+\varepsilon^2(\alpha M+M-p)}/\sqrt{R}.$$

In other words, SNR satisfies (1). □

Remark 1. Since the number of samples R needed is determined by the probability p, if R/p^2 is bigger than R^M or less than R^S, then the attacker may not obtain a useful signal over the noise even if the hypothesis for the secret key was right.

Under the Hamming weight model, the amplitude of the bias signal depends on the number of bits output by the partitioning function $D(*)$. Thus, a possible way to increase the amplitude of the DPA bias signal is to increase the number of bits output by $D(*)$ to more than one bit. That is, S_0 and S_1 in section III.2.B are changed so that output d bits of $D(*)$ are all zero or one.

$$S_0=\{W_r(t)\mid D(*)=0^d\},$$
$$S_1=\{W_r(t)\mid D(*)=1^d\}.$$

Thus, the DPA bias signal becomes $d\varepsilon$. This approach is called d-bit DPA [14]. The SNR for the d-bit DPA on the BSD countermeasures can be written as

$$SNR=\frac{\varepsilon dp\sqrt{R}}{\sqrt{8\sigma^2+\varepsilon^2(\alpha M+M-pd)}}.$$

The proof is similar to that of proposition 3, except that the algorithmic noise for the bias signal is changed to $p(M\text{–}d)+(1\text{–}p)M=(M\text{–}pd)$ random bits. Thus, it is easy to see that the variance of the bias signal is $V[S]=4(\sigma^2+\varepsilon^2(M\text{–}pd)/4)/R$.

B. DPA on L-to-R-Addition-Subtraction Algorithm

Assume the attacker first classifies power consumption by the hypothetical intermediate point $H_r^i=(2^2 k_{[i+1,n-1]})P_r$ corresponding to the hypothesis for $k_i=0$.

Proposition 4. If the appearance probability of the hypothetical intermediate value corresponding to the target bit k_i is p (>0), the use of R^S/p^2 samples enables the attacker to recover k_i. That is, if a significant bias signal is visible, $k_i=0$; otherwise, $k_i=1$.

Proof. First we discuss the case of k_i=0. When algorithm 4 manipulates the i-th bit of d generated from k, $2k_{[i+1,n-1]}$ or $2k_{[i+1,n-1]}+1$ is computed because of proposition 2. The next iteration of the flow computes $4k_{[i-1,n-1]}P$ or $(4k_{[i-1,n-1]}+2)P$. Note that the former is the hypothetical intermediate value. From the assumption, the appearance probability of the hypothetical intermediate value is p. Proposition 3 shows that the use of R^S/p^2 samples enables the attacker to recognize a visible DPA bias signal because of the assumption for his/her capability. Since the bias signal shows that the hypothesis is correct, the attacker reveals k_i=0. The discussion on the case for k_i=1 is similar. □

By proposition 4, if the attacker uses R samples such that $R \geq R^S/p^2$ then he/she can recover k_i. This proposition makes the DPA attack on BSD countermeasures combined with SPA countermeasures theoretically feasible. However, in practice, the attacker may not find any appreciable bias signal over noise, even though the hypothesis for the secret key is right. We consider the following two cases.

Problem 1. The probability p=0. Namely, the actual intermediate value is always $F_r^i=(2^2k_{[i+1,n-1]}+2)P_r$ during R executions. Therefore, the attacker may not know whether k_i is 0.

Problem 2. The probability p is so small that $R=R^S/p^2>R^M$. It implies that the attacker cannot practically obtain $R=R^S/p^2$ samples needed to determine k_i.

There is a further problem. The probability p is not a fixed value, and it changes with each iteration, so it is difficult for the attacker to predict the probability p in advance because, in fact, the probability depends on the secret key and the used random recoding method. Therefore, it seems hard to determine the exact number of R samples needed in the DPA attack such that $R \geq R^S/p^2$.

We now describe how to solve these problems. Assume that we always use the maximum number of samples R^M to recover k_i, that is, $R=R^M$. The smallest probability p that an attacker can recognize a DPA bias signal is sqrt(R^S/R^M) by proposition 3, where sqrt(x) denotes the positive square root of x. Let the smallest appearance probability p=sqrt(R^S/R^M) such that an attacker can detect a DPA bias signal be denoted as LB. It means a lower bound of the appearance probability for detecting the signal. In other words, $p \geq LB$ is equivalent to $R^M \geq R^S/p^2$, that is, the number of samples used, R^M, is enough to detect a useful bias signal. Note that the value of LB depends on the ability of an attacker because it is closely related with the size of R^M.

We propose a new DPA attack strategy based on fact 1. Assume that an attacker uses R^M samples, or $R=R^M$. Suppose the attacker already knows the highest bits $(k_{n-1}\cdots k_{i+1})_2$ of the secret key k. The attacker's goal is to recover the next bit k_i. Then we have a new DPA attack strategy as follows.

Step 1. With the hypothetical intermediate value $H_r^i=(2^2k_{[i+1,n-1]})P_r$, execute steps 2 to 5 as given in section III.2.B. If some useful bias signal appears over noise, or $p \geq LB$, then k_i=0.

Step 2. Else, with the other hypothetical intermediate value $H_r^i=(2^2k_{[i+1,n-1]}+4)P_r$, execute steps 2 to 5 as given in section III.2.B. If some useful bias signal appears over noise, or $p \geq LB$, then k_i=1.

Step 3. Otherwise, we cannot determine whether k_i is 0 because the actual intermediate value $F_r^i=(2^2k_{[i+1,n-1]}+2)P_r$ is operated with high probability.

Whereas steps 1 and 2 recover k_i, step 3 does not determine k_i. To solve this problem, we extend the hypothesis for the secret key to several consecutive bits instead of one bit. For simplicity, we guess two consecutive bits at once. For all $(k_i,k_{i-1})_2$, the actual intermediate point F_r^{i-1} is as follows.

Fact 2.

If $(k_i,k_{i-1})_2=(00)_2$
then F_r^{i-1} is either $(2^3k_{[i+1,n-1]})P_r$ or $(2^3k_{[i+1,n-1]}+2)P_r$.
If $(k_i,k_{i-1})_2=(01)_2$
then F_r^{i-1} is either $(2^3k_{[i+1,n-1]}+2)P_r$ or $(2^3k_{[i+1,n-1]}+4)P_r$.
If $(k_i,k_{i-1})_2=(10)_2$
then F_r^{i-1} is either $(2^3k_{[i+1,n-1]}+4)P_r$ or $(2^3k_{[i+1,n-1]}+6)P_r$.
If $(k_i,k_{i-1})_2=(11)_2$
then F_r^{i-1} is either $(2^3k_{[i+1,n-1]}+6)P_r$ or $(2^3k_{[i+1,n-1]}+8)P_r$.

From the above fact, we can find some useful relations.

1. $F_r^{i-1}=(2^3k_{[i+1,n-1]})P_r$ and $F_r^{i-1}=(2^3k_{[i+1,n-1]}+8)P_r$ only appear in the cases of $(k_i,k_{i-1})_2=(00)_2$ and $(11)_2$, respectively.
2. If $(2^3k_{[i+1,n-1]}+2)P_r$ and $(2^3k_{[i+1,n-1]}+6)P_r$ appear, then we can believe k_i=0 and 1, respectively.

Besides these cases, if the appearance probabilities of two actual intermediate values in the same class in fact 2 are larger than LB, we can recover both bits of the secret key. For example, if the appearance probabilities for both $H_r^{i-1}=(2^3k_{[i+1,n-1]}+2)P_r$ and $(2^3k_{[i+1,n-1]}+4)P_r$ satisfy the condition $p \geq LB$, then we can be convinced that $(k_i,k_{i-1})_2=(01)_2$. Suppose an attacker uses $R^M=9R^S$, that is, LB=1/3, and p=0.6 when $H_r^{i-1}=(2^3k_{[i+1,n-1]}+2)P_r$ and p=0.4 when $H_r^{i-1}=(2^3k_{[i+1,n-1]}+4)P_r$. The attacker can detect $(k_i,k_{i-1})_2=(01)_2$ because these two probabilities p, 0.6 and 0.4, are greater than LB=1/3. By recursively processing in this way, we can recover the remaining bits.

C. Simulation

In the original DPA attack on the standard binary method with n-bit secret keys, we can detect a visible DPA bias signal by the hypothesis for one bit of the secret keys if the hypothesis

is correct. To recover one bit of the secret key, we perform steps 2 to 5 as given in section III.2.B. The original DPA attack requires n times hypotheses for one bit of the secret keys. That is, the total number of hypotheses required to recover the whole n-bit key is n. For example, if we want to recover a 160-bit key in the original DPA attack on algorithm 3, then we perform steps 2 to 5 160 times and step 1 of section III.2.B one time.

In our DPA attack, we use the hypothesis for partial segments of the secret keys (namely, consecutive bits of the secret keys) instead of one bit. Thus, the complexity of our DPA attack depends on the maximum length of the hypotheses for partial segments of the secret keys and the total number of the hypotheses required in the DPA attack. We estimate the maximum length of the hypotheses for partial segments of the secret keys and the total number of hypotheses tried in the proposed DPA attack to completely recover the secret keys by computing the appearance probability of hypothetical intermediate values from software written in C language.

We carried out simulations on the Ha-Moon countermeasure, the Ebeid-Hasan countermeasure, and that by Agagliate and others.

For the simulation, we implemented the Ha-Moon method, the Ebeid-Hasan method, and that by Agagliate and others with algorithm 4 on typical microprocessors, Pentium IV/2GHz (32bit μP; Windows XP, MSVC). A 160-bit secret key was randomly chosen. We obtained 10,000 random recoded numbers generated from the secret key by random recoding methods. We computed the appearance probability of two intermediate values for all bits using the given 10,000 random recoded numbers, that is, $\Pr[d_{[i,n]}=k_{[i,n-1]}]$ and $\Pr[d_{[i,n]}=k_{[i,n-1]}+1]$. For each LB, we tried to recover the secret key by the proposed DPA attack. From the result of the above step, we counted the maximum length of hypotheses for partial segments of the secret key and the total number of hypotheses to completely recover the 160-bit secret key. We computed the average of the maximum length of hypotheses for partial segments of the secret keys and the total number of hypotheses for each LB, that is, the sum of the maximum length of hypotheses/10,000 and the sum of the total number of hypotheses/10,000.

Table 1 shows the total number of hypotheses in the proposed DPA attack on the BSD countermeasures with algorithm 4 according to the ability to obtain R^M.

As shown in Table 1, if we use $100R^S$ samples, we can recover a 160-bit secret key used in the Ha-Moon countermeasure, the Ebeid-Hasan countermeasure, and the countermeasure by Agagliate and others by performing steps 2 to 5 of the DPA in section III.2.B, 868, 478, and 2,893 times, respectively.

Table 1. The number of hypotheses needed in the proposed DPA attack to recover 160-bit keys.

R^M	Countermeasures		
	Ha-Moon [5]	Ebeid-Hasan [4]	Agagliate et al. [7]
$2R^S$	16,484	1,511	4,004
$5R^S$	2,716	711	2,917
$10R^S$	1,987	596	2,915
$20R^S$	1,480	543	2,898
$50R^S$	1,019	580	2,896
$100R^S$	868	478	2,893

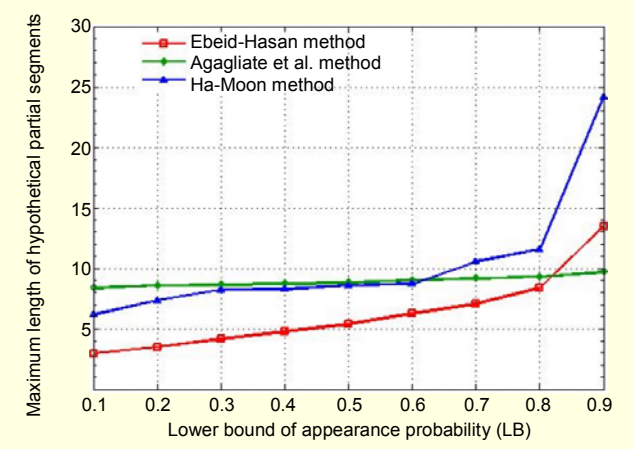

Fig. 1. Maximum length of hypotheses for each lower bound of the appearance probability for 160-bit secret keys.

Figure 1 shows the maximum consecutive bits of hypotheses to reveal the secret keys depending on the computational power of an attacker (LB). If an attacker has the capability to guess 10 consecutive bits, then the samples needed to obtain the same SNR as proposition 1 are about $2R^S$ for the Ha-Moon countermeasure, $1.4R^S$ for the Ebeid-Hasan countermeasure, and R^S for the countermeasure proposed by Agagliate and others. The countermeasure by Agagliate and others could be broken by using the same number of samples as the ordinary DPA attack.

From Fig. 1 and Table 1, we can derive the following conclusion. Suppose we use $10R^S$ samples. Then, $LB=\text{sqrt}(R^S/(10R^S)) \approx 0.32$. For the Ha-Moon method, the attacker executes hypotheses 1,987 times, and the length to be guessed at once is 10 bits at most. For the Ebeid-Hasan method, the attacker executes hypotheses 596 times, and the length to be guessed at once is 710 bits at most. For the method proposed by Agagliate and others, the attacker executes hypotheses 2,915 times, and the length to be guessed at once is 9 bits at most. This results show the practicality of the proposed DPA attack.

V. Application to R-to-L Computation

Until now we have assumed that algorithms for computing the scalar multiplication scan the bits of the secret key from MSB to LSB. However, the Oswald-Aigner countermeasure [6] works from LSB to MSB. In this section, we apply the proposed DPA attack to BSD countermeasures processed from LSB to MSB (R-to-L).

Algorithm 5. R-to-L-Addition-Subtraction Algorithm.
Input: A point P and $k=k_{n-1}2^{n-1}+\cdots+k_1 2+k_0$, $k_i \in \{0,1\}$.
Output: $Q = kP$
1. Convert k to $d=d_n 2^n+\cdots+d_0$, where $d_i \in \{\pm 1,0\}$
2. $Q[0] = O$, $R[0] = -P$, $R[1] = P$, $Q[2] = P$
3. For j=0 to n
   3.1. $Q[1]$ = ECADD($Q[0]$,$R[d_j+1]$)
   3.2. $R[2]$ = ECDBL($R[2]$)
   3.3. $R[1] = R[2]$, $R[0] = -R[2]$
   3.4. $Q[0] = Q[|d_j|]$
4. Return $Q[0]$

The following proposition is easily derived from proposition 2.

Proposition 5. $d_{[0,i]}$ is either $k_{[0,i]}$ or $k_{[0,i]} - 2^{i+1}$.

For $j=i$, $Q[0]= k_{[0,i]}P$ or $(k_{[0,i]} - 2^{i+1})P$ at step 3.4 in algorithm 5. Then, the intermediate values calculated at step 3.1 in the next loop are $k_{[0,i]}P + 2^{i+1}P$ or $(k_{[0,i]} - 2^{i+1})P + 2^{i+1}P$ if $d_{i+1} = 0$ or 1, and $k_{[0,i]}P - 2^{i+1}P$ or $(k_{[0,i]} - 2^{i+1})P - 2^{i+1}P$ if $d_{i+1} = -1$.

As a result, we have the following relation between k_i and the intermediate values. Suppose an attacker already knows the lowest bits $(k_{i-1}\cdots k_0)_2$ of the secret key k. The attacker's goal is to recover the next bit k_i.

Fact 3.
If k_i=0, then the intermediate value is one of $(k_{[0,i-1]} + 2^{i+1})P_r$, $k_{[0,i-1]}P_r$, $(k_{[0,i-1]} - 2^{i+1})P_r$, or $(k_{[0,i-1]} - 2^{i+2})P_r$.
If k_i=1, then the intermediate value is one of $(k_{[0,i-1]} + 3\cdot 2^i)P_r$, $(k_{[0,i-1]} + 2^i)P_r$, $(k_{[0,i-1]} - 2^i)P_r$, or $(k_{[0,i-1]} - 3\cdot 2^i)P_r$.

The intermediate values for k_i=0 are totally different from those for k_i=1; therefore, in R-to-L algorithms we can use one of four intermediate values as the hypothetical intermediate value. That is, the hypothetical intermediate value could be $(k_{[0,i-1]} + 2^{i+1})P_r$, $k_{[0,i-1]}P_r$, $(k_{[0,i-1]} - 2^{i+1})P_r$, or $(k_{[0,i-1]} - 2^{i+2})P_r$ for hypothesis k_i=0.

Proposition 6. In algorithm 5, the number of hypotheses tried to recover the i-th bit of the secret key bit k_i is a maximum of four since there are four intermediate values. As there is no collision between intermediate values for k_i=0 and 1, we first use $(k_{[0,i-1]} + 2^{i+1})P_r$, $k_{[0,i-1]}P_r$, $(k_{[0,i-1]} - 2^{i+1})P_r$, and $(k_{[0,i-1]} - 2^{i+2})P_r$. If we detect a visible bias signal over noise, then k_i is 0; otherwise, k_i is 1.

Therefore, the attack against R-to-L algorithms can recover the secret key bit by bit, using a procedure similar to the ordinary DPA attack. Clearly, the attack on algorithm 5 is simpler and more efficient than that on algorithm 4.

Remark 2. Since the Oswald-Aigner countermeasure [6] utilizes an R-to-L algorithm, it is very easily broken by the proposed DPA attack on algorithm 5.

VI. Extended RPA Attack

The proposed attack is included in the class of general DPA attacks. The BSD countermeasure can further be combined with some DPA countermeasures in order to strengthen the security of BSD countermeasures. The DPA countermeasure using randomized point representation methods is one of most efficient classes of DPA countermeasures. This class includes the randomized projective coordinate [15] and random isomorphism methods [24]. However, Goubin proposed the refined power analysis (RPA) using "special points" $(x, 0)$ and $(0, y)$ which cannot be randomized by randomized point representation methods [25]. If we use the special point as a hypothetical intermediate value, improved BSD countermeasures combined with randomized point representation methods could be broken.

Note that other notations and assumptions are the same as those in the previous sections with the exception of combining algorithm 4 with a DPA countermeasure. We can then find differences between DPA and RPA attacks. The RPA attack is achieved by the following four steps.

Step 1. Guess a part of the secret key.
Step 2. Calculate the input point corresponding to the guess.
Step 3. Measure power consumption.
Step 4. Compute the mean of the power consumption.

The first step of the RPA attack is to guess the secret key. To recover one bit of the secret key, an attacker has to perform steps 1 to 4. Therefore, the power consumption needed to recover the next bit has to be re-measured.

1. SNR of RPA in BSD Countermeasures

Similar to the DPA attack on algorithm 4 described in propositions 2 and 3, we introduce a proposition which deals with SNR for the RPA attack on the addition-subtraction algorithm with the DPA countermeasure using randomized point representation methods.

Proposition 7. Assume that the computational environment is the same as in proposition 3. If the appearance probability of the special point P_0 is q, then

$$SNR = \frac{\varepsilon Mq\sqrt{R}}{\sqrt{8\sigma^2 + \varepsilon^2(\alpha M + M - Mq)}}. \quad (6)$$

The SNR is approximately q times larger than the original RPA. In other words, in order to obtain the same SNR as the original, the required samples are $1/q^2$ times larger than the original.

Note that the proof is similar to that of proposition 3 (see theorem 3 in [14]). Actually, the appearance probability p of the hypothetical intermediate value in the proposed DPA attack is exactly the same as the appearance probability q of the special point P_0.

Remark 3. Proposition 7 shows that an attacker needs approximately R/M^2 samples to obtain the same SNR as in proposition 3 (for the proposed DPA attack).

2. Adaptively Chosen Plaintext Attack

The RPA attack is an adaptively chosen plaintext attack. All hypothetical intermediate values in the RPA attack are chosen by an attacker. Since the RPA attack requires the special point for detecting a specific bit of the secret key, the observed samples cannot be reused. That is, in detecting each bit, the attacker has to measure the power consumption for new plaintexts. Therefore, in the proposed RPA attack, the maximum number of samples that an attacker can use for each hypothesis is R^M/(the number of hypotheses) on average. Thus, LB=sqrt(R^S/R^M) used in the DPA attack of the previous section is replaced with sqrt((R^S×the number of hypotheses)/R^M). For a more successful attack we can use smaller samples if R^M/(the number of hypotheses) > R^M/M^2, and larger samples if not, but the total number of samples should be less than R^M.

VII. Security Estimation

As described in the previous sections, the basic BSD countermeasures are vulnerable to various attacks; therefore, they should be combined with additional countermeasures to resist SPA and DPA attacks. In this section, we investigate secure combinations of BSD with SPA or DPA countermeasures.

1. Combining SPA Countermeasures

When BSD countermeasures are combined with some SPA countermeasures, we estimate the security of combinations between them. One possible way to strengthen the security of BSD countermeasures is to combine them with an SPA countermeasure such that a hypothetical intermediate value is manipulated at the same time, such as Coron's dummy method and Montgomery ladder methods, Another possibility is to combine them with an SPA countermeasure such that a hypothetical intermediate value is manipulated at different times like the unified code and the side channel atomic blocks. We have shown that the combination of the BSD countermeasure and the fixed procedure type countermeasures is vulnerable to the proposed DPA attack.

We now consider the effect of the randomized path of computations. This has the effect of dispersing the amplitude of a DPA bias signal across the differential trace because of desynchronization of the time at which the target hypothetical intermediate values have been processed. However, the methods to reconstruct the original bias signal, such as the total power summation attack [14] and window-DPA [26] proposed analyzing the time randomization techniques, such as inserting random delays and randomly rearranging the order of operations; however, the number of samples required in DPA attacks sharply increases. In the reconstruction methods, it is assumed that the original bias signal is spread over consecutive clocks. However, since the bias signal caused by the randomized path is spread over consecutive elliptic operations it is spread over very large clocks. Moreover, the bias signal decreased by the appearance probability of hypothetical intermediate values is again decreased by this desynchronization by the randomized path. The DPA bias signal contains excessive noise, and the number of samples needed in the DPA attack becomes extremely large. This makes the attack practically infeasible [17]. The effect of the desynchronization may provide enough security, even though the entire bias signal is distributed over some clocks.

2. Combining DPA Countermeasures

To further strengthen the security of BSD countermeasures, we can combine them with DPA countermeasures. One DPA countermeasure blind inputs points by adding a random point as in Coron's second method [15] and the random initial point method [27]. Another approach changes the representation of input points before the beginning of scalar multiplications, as in the randomized projective coordinate [15] and random isomorphism methods [24]. Whereas randomized point representation DPA countermeasures are vulnerable to the proposed RPA attack, the random point blinding DPA countermeasure is not vulnerable.

In Table 2, "Secure" and "Insecure" mean that the

Table 2. Secure combinations between the BSD type and SPA or DPA countermeasures.

SPA countermeasures		DPA countermeasures	
FPT [15], [18], [19]	IOT [20], [21]	RPRT [15], [24]	RPBT [15], [27]
Insecure	Secure	Insecure	Secure

countermeasure is vulnerable or invulnerable to the proposed attack, respectively; FPT and IOT denote fixed procedure types, such as Coron's dummy method [15] or Montgomery ladder methods [18], [19] and the indistinguishable operation type such as the unified code [20] or the side channel atomic blocks method [21] as SPA countermeasures, respectively; RPRT and RPBT denote randomized point representation methods, such as the randomized projective coordinate [15] and the random isomorphism method [24] and random point blinding methods, such as Coron's second method [15] and the random initial point method [27] as DPA countermeasures, respectively.

Remark 4. If BSD countermeasures are combined with an indistinguishable type of operation using the same addition formulae [20], [21] or with a random point blinding type, then BSD countermeasures may be invulnerable to the proposed attack in addition to the previous attacks.

VIII. Comparison

To analyze the original version or the improved version of BSD countermeasures there have been many attacks. The hidden Markov model (HMM) attack proposed by Karlof and Wagner is a cryptanalytic framework for countermeasures which utilizes a probabilistic finite state machine [12]. The HMM attack is available under the assumption of the ability to distinguish between ECADD and ECDBL. Thus, the HMM attack seems unable to break the combined BSD and SPA countermeasures.

Recently, Fouque and others [28] and Sim and others [29] proposed attacks on the full version of the Ha-Moon countermeasure composed of a random recoding method based on the NAF recoding technique and an SPA-immune algorithm using dummy operations. The model proposed by Fouque and others is based on collision detection, which is rather different from the usual DPA attack model in practice. To find collisions, their attack utilizes the multiple exponent single data (MESD) technique [14]. The MESD requires that an attacker has two identical devices with the same algorithm, one with an unknown secret key and the other with a secret key by the attacker. To recover the unknown secret key, we compare the power consumption of the two devices. If the power consumption for each device is similar, then the secret keys are equal; otherwise, the secret keys differ. Such a situation may be less practical than the zero exponent multiple data (ZEMD) technique, which only requires a device with an unknown secret key.

However, the proposed DPA attack and the attack by Sim and others utilize the ZEMD technique [14]. The difference between them is the examination of the appearance probability of intermediate values. The attack by Sim and others works under the assumption that the probability is always 1/2. Unfortunately, by our simulation, this is not true. In fact, the probability depends on both the random recoding method and the secret key used in scalar multiplications. Therefore, sometimes the attack by Sim and others cannot obtain useful information from the measured power consumption even though the key hypothesis is right.

In Table 3, "Feasible" means that the attack can break the combined BSD countermeasure and "Infeasible" means that the attack cannot break the combined BSD countermeasure. In Table 2, FPT, IOT, RPRT, and RPBT are denoted.

Table 3. Comparison to previous works.

Attaks	Fouque et al. [28]	Sim et al. [29]	HMM [12]	Proposed Attack
FPT	Feasible	Infeasible	Infeasible	Feasible
IOT	Infeasible	Infeasible	Infeasible	Infeasible
RPRT	Infeasible	Infeasible	Infeasible	Feasible
RPBT	Infeasible	Infeasible	Infeasible	Infeasible
R-to-L	Feasible	Infeasible	Feasible	Feasible
Model	MESD	ZEMD	HMM	ZEMD

IX. Concluding Remarks

In this paper, we have estimated the security of BSD countermeasures combined with SPA or DPA countermeasures. We have further extended the proposed attack to R-to-L computation. In order to show the practicality of the proposed attack, we have examined the number of samples needed in the DPA attack since this affects the success of statistical power attacks. Simulations were performed, and the results verify that the proposed attack is practicable.

References

[1] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. Comp*., vol. 48, 1987, pp. 203-209.

[2] V.S. Miller, "Use of Elliptic Curves in Cryptography," *Advances in Cryptology - CRYPTO '85*, 1986, LNCS 218, pp. 417-426.

[3] C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Advances in Cryptology - CRYPTO*, LNCS 1666, 1999, pp. 388-397.

[4] N. Ebeid and A. Hasan, "On Randomizing Private Keys to Counteract DPA Attacks," *Selected Areas in Cryptography - SAC*, LNCS 3006, 2004, pp. 58-72.

[5] J. Ha and S. Moon, "Randomized Signed-Scalar Multiplication of

ECC to Resist Power Attacks," *Cryptographic Hardware and Embedded Systems - CHES*, LNCS 2523, 2002, pp. 551-563.

[6] E. Oswald and M. Aigner, "Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks," *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, 2001, pp. 39-50.

[7] S. Agagliate, P. Guillot, and O. Orcière, "A Randomized Efficient Algorithm for DPA Secure Implementation of Elliptic Curve Cryptosystems," *The Proc. of Workshop on Coding and Cryptography 2003 - WCC*, 2003, pp. 11-19.

[8] N. Ebeid and A. Hasan, "Analysis of DPA Countermeasures Based on Randomizing the Binary Algorithm," Technical Report of the University of Waterloo, no. CORR 2003-14. http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-4.ps.

[9] K. Okeya and K. Sakurai, "On Insecurity of the Side Channel Attack Countermeasure Using Addition-Subtraction Chains under Distinguishability between Addition and Doubling," *Australasian Conf. on Information Security and Privacy - ACISP 2002*, LNCS 2384, 2002, pp. 420-435, 2002.

[10] C.D. Walter, "Issues of Security with the Oswald-Aigner Exponentiation Algorithm," *Topics in Cryptology - CT-RSA*, LNCS 2964, 2004, pp. 208-221.

[11] K. Okeya and D-G. Han, "Side Channel Attack on Ha-Moon's Countermeasure of Randomized Signed Scalar Multiplication," *INDOCRYPT 2003*, LNCS 2904, 2003, pp. 334-348.

[12] C. Karlof and D. Wagner, "Hidden Markov Model Cryptanalysis," *Cryptographic Hardware and Embedded Systems - CHES*, LNCS 2779, 2003, pp. 17-34.

[13] D-G Han, K. Okeya, T.H. Kim, Y.S. Hwang, B. Kim, and Y-H. Park, "Enhanced Exhaustive Search Attack on Randomized BSD Type Countermeasure," *IEICE Trans. Fundamentals, Special Section on Discrete Mathematics and Its Applications*, vol. E89-A, no. 5, May 2005, pp. 1316-1327.

[14] T.S. Messerges, "Power Analysis Attacks and Countermeasures for Cryptographic Algorithms," PhD dissertation, Univ. of Illinois at Chicago, 2000.

[15] J.S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," *Cryptographic Hardware and Embedded Systems - CHES*, LNCS 1717, 1999, pp. 292-302.

[16] E. Brier, C. Clavier, and F. Oliver, "Correlation Power Analysis with a Leakage Model," *Cryptographic Hardware and Embedded Systems - CHES 2004*, LNCS 3156, 2004, pp. 16-29.

[17] S. Mangard, "Hardware Countermeasures against DPA] - A Statistical Analysis of Their Effectiveness," *Topics in Cryptology - CT-RSA 2004*, LNCS 2964, 2004, pp. 222-235.

[18] T. Izu and T. Takagi, "A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks," *Public Key Cryptography – PKC*, LNCS 2274, 2002, pp. 280-296.

[19] M. Joye and S-M. Yen, "The Montgomery Powering Ladder," *CHES*, LNCS 2523, 2002, pp. 291-302.

[20] E. Brier and M. Joye, "Weierstrass Elliptic Curves and Side-Channel Attacks," *Public Key Cryptography - PKC 2002*, LNCS 2274, 2002, pp. 335-345.

[21] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity," *IEEE Trans. Computers*, vol. 53, no. 6, 2004, pp. 760-768.

[22] P.L. Montgomery, "Speeding the Pollard and Elliptic Curve Methods of Factorization," *Mathematics of Computation*, vol. 48, no. 177, Jan. 1987, pp. 243-264.

[23] ANSI X9.63, Public Key Cryptography for the Financial Services Industry: Elliptic Curve Key Agreement and Transport Protocols, draft, 1999.

[24] M. Joye and C. Tymen, "Protections against Differential Analysis for Elliptic Curve Cryptography: An Algebraic Approach," *Cryptographic Hardware and Embedded Systems - CHES 2001*, LNCS 2162, 2001, pp. 377-390.

[25] L. Goubin, "A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems," *Public Key Cryptography - PKC*, LNCS 2567, 2002, pp. 199-211.

[26] C. Clavier, J.S. Coron, and N. Dabbous, "Differential Power Analysis in the Presence of Hardware Countermeasures," *Cryptographic Hardware and Embedded Systems - CHES*, LNCS 1965, 2000, pp. 252-263.

[27] H. Mamiya, A. Miyaji, and H. Morimoto, "Efficient Countermeasures against RPA, DPA, and SPA," *Cryptographic Hardware and Embedded Systems - CHES*, LNCS 3156, 2004, pp. 343-356.

[28] P.A. Fouque, F. Muller, G. Poupard, and F. Valette, "Defeating Countermeasures Based on Randomized BSD Representations," *Cryptographic Hardware and Embedded Systems - CHES*, LNCS 3156, 2004, pp. 312-327.

[29] S.G. Sim, D.J. Park, and P.J. Lee, "New Power Analyses on the Ha-Moon Algorithm and the MIST Algorithm," *Int'l Conf. Information and Communication Security - ICICS 2004*, LNCS 3269, 2004, pp. 291-304.

Tae Hyun Kim received the BS degree in mathematics from University of Seoul, Seoul, Korea, in 2002, and the MS degree in engineering in information security from Korea University, Seoul, Korea, in 2004. He was a visiting researcher in the School of Systems Information Science in Future University, Hakodate, Japan, from April 2006 to September 2006. He is currently a PhD student at Korea University and is also a researcher with the Center for Information Security Technologies (CIST).

Dong-Guk Han received his BS degree in mathematics from Korea University in 1999, and his MS degree in mathematics from Korea University in 2002. He received his PhD in engineering in information security from Korea University in 2005. He was a post-doctoral researcher with Future University, Hakodate, Japan. After finishing his PhD course, he was an exchange student with the Department of Computer Science and Communication Engineering of Kyushu University from April 2004 to March 2005. He has been a senior researcher with Electronics and Telecommunications Research Institute since June 2006. He is a member of KIISC, IEEK, and IACR.

Katsuyuki Okeya received the BS degree from Toyama University in 1994 and the MS degree from Kyushu University in 1996. He worked on the development of information security with the Software Division of Hitachi Ltd. from 1998 to 2001. He received the PhD degree in engineering from Kyushu University in 2004. He has conducted research on cryptography and information security with the Systems Development Laboratory of Hitachi Ltd. since 2001. He received the IPSJ Best Paper Award from the Information Processing Society of Japan in 2004. Dr. Okeya is a member of the Information Processing Society of Japan; the Institute of Electronics, Information, and Communication Engineers; the Japan Society for Industrial and Applied Mathematics; and the Mathematical Society of Japan.

Jongin Lim received the BS, MS, and DS degrees from Korea University, Seoul, in 1980, 1982, and 1986, respectively. He is currently a professor of the Graduate School of Information Security of Korea University and a dean of the Center for Information Security Technology (CIST). His current research interests include cryptanalysis, theory of cryptography, and cyber-law.