# Practical Second-Order Correlation Power Analysis on the Message Blinding Method and Its Novel Countermeasure for RSA

HeeSeok Kim, Tae Hyun Kim, Joong Chul Yoon, and Seokhie Hong

**Recently power attacks on RSA cryptosystems have been widely investigated, and various countermeasures have been proposed. One of the most efficient and secure countermeasures is the message blinding method, which includes the RSA derivative of the binary-with-random-initial-point algorithm on elliptical curve cryptosystems. It is known to be secure against first-order differential power analysis (DPA); however, it is susceptible to second-order DPA. Although second-order DPA gives some solutions for defeating message blinding methods, this kind of attack still has the practical difficulty of how to find the points of interest, that is, the exact moments when intermediate values are being manipulated. In this paper, we propose a practical second-order correlation power analysis (SOCPA). Our attack can easily find points of interest in a power trace and find the private key with a small number of power traces. We also propose an efficient countermeasure which is secure against the proposed SOCPA as well as existing power attacks.**

**Keywords: RSA cryptosystems, side channel attacks, message blinding method, BRIP, second-order DPA.**

Manuscript received May 1, 2009; revised Sept. 8, 2009; accepted Oct. 28, 2009.

This work was supported by the Second Brain Korea 21 Project.

HeeSeok Kim (phone: +82 10 4749 3425, email: 80khs@korea.ac.kr) and Seokhie Hong (corresponding author, phone: +82 2 3290 4894, email: hsh@cist.korea.ac.kr) are with the Center for Information Security Technologies (CIST), Korea University, Seoul, Rep. of Korea.

Tae Hyun Kim (email: thkim7042@gmail.com) is with Institute Attached to ETRI, Daejeon, Rep. of Korea.

Joong Chul Yoon (email: joongchul.yoon@samsung.com) is with the System LSI Division, Samsung Electronics, Yongin, Rep. of Korea.

doi:10.4218/etrij.10.0109.0249

## I. Introduction

RSA is a public key cryptosystem that is widely used in a growing number of embedded applications such as smart cards. Even though these systems are proven to be secure with mathematical tools, they could be vulnerable to physical attacks using additional information via side channels. For the last few years, increased research has focused on these implementation vulnerabilities. These sorts of attacks are referred to as side channel attacks (SCAs) and were first introduced by Kocher [1].

In the categories of SCAs, one actively researched technique used on RSA cryptosystems is the power analysis attack, which was first published by S. Messerges and others [2]. Power analysis attacks on RSA cryptosystems are classified into two kinds of attacks, simple power analysis (SPA) and differential power analysis (DPA). While an SPA can expose secret information by observing the power consumption of a single execution of a cryptographic algorithm, a DPA is more sophisticated and powerful. It finds secret information by a statistical analysis of many executions of the same algorithm with different inputs without physical decapsulation of the target device. In a simple example of SPA, a secret key can be easily exposed in the binary exponentiation algorithm by distinction between the squaring signal and the multiplication signal. However, DPA is more powerful and complicated than SPA because it can lead to the recovery of the secret key even in extraordinary cases: first when it is impossible to distinguish between the squaring signal and the multiplication signal, that is, when a countermeasure based on atomicity [3] is used, and second, when an exponentiation is computed by a predetermined fixed path, such as Coron's dummy method [4] and the Montgomery ladder (ML) method [5].

Messerges and others introduced three DPA attacks on the binary modular exponentiation algorithm: single-exponent multiple-data (SEMD), multiple-exponent single-data (MESD), and zero-exponent multiple-data (ZEMD) [2]. Amiel and others advanced power analysis techniques using correlation on the Chinese remainder theorem (CRT), RSA, and a non-CRT RSA [6]. Both DPA and correlation power analysis (CPA) attacks use some relation between power consumption and intermediate values appearing according to part of the secret key. The reason why power analysis attacks can recover the secret key is that the power consumption signal from a CMOS cryptographic device is strongly related to the internal state of the device.

DPA attacks can be precluded by eliminating or hiding the relationship between the power consumption and the internal state. There are two popular methods to achieve this goal: the exponent blinding method and the message blinding method. In the exponent blinding method, the exponent is randomized or split. The exponent randomization (ER) randomly changes the exponent by adding the order $\varphi(n)$, in which $x^d$ mod $n$ is computed by $x^{d+r\varphi(n)}$ mod $n$ [4]. The disadvantage of ER is that it requires the additional system parameter $\varphi(n)$ and additional computation overhead for random number $r$. What is worse, its security is also controversial [7]. The exponent splitting (ES) method [8] computes $x^r x^{d-r}$ mod $n$, and the improved exponent splitting (IES) method [9] computes $(x^r)^{\lfloor d/r \rfloor} x^{(d \bmod r)}$ mod $n$ instead of $x^d$ mod $n$, where $\lfloor u \rfloor$ and $r$ are the largest integer less than or equal to $u$ and a random number, respectively. ES has the disadvantages of a doubled computation time compared to the case without countermeasures as well as the storage for $r$ and $d - r$. The disadvantage of IES is that it requires the inverse of a random number, which takes large computational time and memory.

The second alternative for preventing power analysis attacks is the message blinding method. In elliptic curve cryptography, the binary-with-random-initial-point (BRIP) is one of the most efficient and popular message blinding methods to preclude SPA, DPA, and refined power analysis (RPA) because the inversion operation is almost free [10], [11]. However, we cannot apply BRIP to RSA directly as this would require an expensive inversion operation. To reduce the cost of the inversion, Amiel and Feix proposed a modified BRIP algorithm for RSA using Montgomery multiplication [12]. This algorithm must be designed not to expose the random number by SPA when generating the random number and its inverse. This vulnerability was also well explained in [12].

Although the BRIP algorithm is designed to be secure against SPA and DPA, this algorithm may still be vulnerable to a second-order DPA (SODPA), such as the Okeya-Sakurai style attack, theoretically proposed in elliptic curve cryptosystems (ECCs) [13]. The weakness is that one operand in the multiplication is fixed and the same pre-computed value is used when the same bits of the secret key are manipulated. This kind of problem can be found in various countermeasures using the message blinding method to thwart first-order DPAs, which utilize left-to-right type exponentiation based on window techniques. Even if SODPAs are proven to be effective in theory, a practical issue remains: how to find points of interest corresponding to the time to use or load fixed values. The practicality of SODPAs is determined by how easy it is to precisely find these points.

In this paper, we propose a new second-order correlation power analysis (SOCPA) which uses correlation coefficients between power traces to be able to analyze modified BRIP algorithms. This new attack can find points of interest with ease and does not require a profiling stage; therefore, it is practical. Besides, this attack requires fewer power traces than Okeya-Sakurai style SODPA. In addition, even when additional countermeasures such as random time delay and random clocks are added to the message blinding methods, our algorithm can find the secret key. Experimental results support these assertions. Our attack can be extended to various countermeasures such as the binary method and the ML method using the message blinding method. Because this attack does not require any plaintext or ciphertext, CRT RSA is also vulnerable.

To resist the proposed SOCPA as well as the previous power analysis attacks, we propose a new countermeasure. This new countermeasure uses the message blinding methods rather than the exponent blinding method to minimize variation of existing systems. Our countermeasure is designed by modifying the ML method, but it does not incur much computation overhead.

The remainder of this paper is organized as follows. In section II, we present the theoretical background for exponentiation in RSA and for power analysis on a naive algorithm and the BRIP algorithms. Our proposed attack and experimental results are presented in section III. In section IV, we introduce a novel secure countermeasure against most power analysis attacks. Finally, we conclude this article in section V.

## II. Related Work

### 1. Modular Montgomery Multiplication and the RSA Exponentiation Algorithm

Montgomery multiplication is a technique which allows an efficient implementation of modular reduction without explicitly carrying out the classical modular reduction step. Let $m$ be a positive integer, and let $R$ be an integer such that $R$ $(=w^t) > m$, $gcd(m, R) = 1$, where $w$ is word size, $t$ is the smallest integer such that $w^t > m$, and $gcd$ denotes the greatest common divisor. To describe the Montgomery multiplication algorithm, we first define the $m$-residue of an integer $A < m$ as '$A$' = $AR$ mod $m$. Here, '$A$' is the Montgomery representation of $A$. Given two $m$-residues '$A$' and '$B$', the Montgomery multiplication is defined as the $m$-residue '$z$' = Mont('$A$', '$B$') = '$A$''$B$'$R^{-1}$ mod $m$, where $z = AB$ mod $m$. Algorithm 1 describes Montgomery multiplication.

```
Algorithm 1. Montgomery multiplication
Input: $A\ (=a_{t-1}a_{t-2}\cdots a_0)_w$, $B\ (=b_{t-1}b_{t-2}\cdots b_0)_w$, $m\ (=m_{t-1}m_{t-2}\cdots m_0)_w$,
       $R = w^t$ and $m' = -m_0^{-1}$ mod $w$.
Output: $ABR^{-1}$ mod $m$.
  1. $S = 0$ (Notation: $S\ (=s_t s_{t-1}\cdots s_0)_w$)
  2. For $i = 0$ to $t-1$ do
    2.1.  $u_i = (s_0 + b_i a_0)m'$ mod $w$.
    2.2.  $S = (S + b_i A + u_i m)/w$.
  3. If $S \geq m$ then $S = S - m$.
  4. Return $S$.
```

Since conversion from an ordinary residue to an $m$-residue, computation of $m'$, and conversion back to an ordinary residue are time-consuming, it is not a good idea to use the Montgomery multiplication algorithm when only a single modular multiplication is performed. It is more suitable when several modular multiplications with respect to the same modulus are needed. Such is the case when one needs to compute a modular exponentiation in RSA. Algorithm 2 represents an RSA exponentiation algorithm using Montgomery multiplication.

```
Algorithm 2. RSA exponentiation algorithm using Montgomery multiplication
Input: $X$, $m$, and $d = (d_{n-1}d_{n-2}\cdots d_1 d_0)_2$ where $n$ is the size of modulus $m$.
Output: $X^d$ mod $m$.
  1. $S_0 = S_1 = \mathrm{Mont}(X, R^2)$.
  2. For $i = n-2$ down to 0 do
    2.1.  $S_0 = \mathrm{Mont}(S_0, S_0)$.
    2.2.  If $d_i = 1$, then $S_0 = \mathrm{Mont}(S_0, S_1)$.
  3. $S_0 = \mathrm{Mont}(S_0, 1)$.
  4. Return $S_0$.
```

## 2. BRIP Countermeasure for RSA

The message blinding method defends against first-order DPA attacks because intermediate values expected by an attacker are blinded by a random number at each execution. The BRIP algorithm is the most popular message blinding method. It was originally proposed for ECC because an inversion operation is almost free in this setting. An RSA derivative of the BRIP algorithm computes $(X^d U^{(11'1'\cdots1')_2}$ mod $m)U^{1'}$ mod $m$, where 1' denotes –1 and $U$ is a random number newly generated for each execution of the exponentiation $X^d$. The computation of $X^d U^{11'1'\cdots1'}$ mod $m$ is accomplished by the simultaneous exponentiation algorithm. The only problem of the BRIP algorithm for RSA is how to compute the inverse $U^{1'}$ of the random number $U$. Amiel and Feix with Ciet and Feix's idea solved this problem by using Montgomery multiplication Mont(1,1)=$R^{-1}$ mod $m$ [12]. Therefore, when $U$ is $R^v$ with a relatively short exponent $v$, $U^{1'}$ can be easily computed by $(\mathrm{Mont}(1,1))^v$. To avoid analysis methods like the doubling attack [14] and the collision attack [12], we recommend using at least a 48-bit random value $v$. These analyses succeed when an attacker can have one pair of traces which are blinded by same masking value. When a 48-bit random value is used, 14 million traces are required in order to find collisions that have a probability of 0.5 according to fact 2.27 of [15]. However, it is practically impossible for any attacker to collect this many power traces. Algorithm 3 shows an RSA version of the BRIP countermeasure using this idea.

```
Algorithm 3. BRIP algorithm for RSA
Input: $X$, $m$, and $d = (d_{n-1}d_{n-2}\cdots d_1 d_0)_2$ where $n$ is the size of modulus $m$.
Output: $X^d$ mod $m$.
  1. If $X = 1$ then return 1. Else if $X = -1$ then return $1 - 2d_0$.
  2. Generate a random value $v$.
  3. Compute $U = R^v$ mod $m$ and $U_0 = (\mathrm{Mont}(1,1))^v = R^{-v}$ mod $m$ using
     SPA resistant exponentiation.
  4. Compute $U = \mathrm{Mont}(U, R^2)$.
  5. Compute $U_0 = \mathrm{Mont}(U_0, R^2)$ and $U_1 = \mathrm{Mont}(\mathrm{Mont}(X, R^2), U_0)$.
  6. For $i = n-1$ down to 0 do.
    6.1.  $U = \mathrm{Mont}(U, U)$.
    6.2.  $U = \mathrm{Mont}(U, U_{d_i})$.
  7. $U = \mathrm{Mont}(U, U_0)$
  8. Return $\mathrm{Mont}(U, 1)$.
```

## 3. Second-Order DPA on BRIP Countermeasure

SODPA is well-known to the analysis method on the DPA countermeasure. The method of [16] can threaten the DPA countermeasure of the block cipher using the additive masking [17]; however, it is doubtful that the message blinding method corresponding multiplicative masking has a weakness against the analysis of [16] because this analysis uses a characteristic to be only processed in the additive masking. The typical SODPA against the message blinding method is the Okeya-Sakurai style SODPA. The Okeya-Sakurai style SODPA on the BRIP countermeasure is performed as follows. In step 6.2 of algorithm 3, $U$=Mont($U$, $U_{d_i}$) is calculated with fixed precomputed values of $U_0$ and $U_1$. The power consumptions when $U_{d_i}$ is being loaded or used are similar if $d_j = d_i$, where $j \neq i$. Therefore, an attacker can find the secret key because the difference of power consumptions corresponding to the loading or manipulating time of the operand for the same bits $d_i = d_j$ ($i \neq j$) is smaller than those for different bits.

However, the practical difficulty of the Okeya-Sakurai style SODPA is how to detect the time when the operand $U_{d_i}$ is loaded or manipulated. This problem is a fundamental and critical issue in SODPA. If this problem is not solved, most SODPA attacks are still not practical and realizable. Another problem of this attack happens when additional countermeasures such as inserting noise, random time delay, and random clocks are applied. Most of state-of-the-art smart cards additionally use such countermeasures to enhance the security of the device. When we try to perform SODPA in these devices in practice, it fails to recover the secret key because of the effect of noisy signal, misalignment, or desynchronization of signals.

## III. Practical Second-Order CPA on the BRIP Algorithm

The SODPA on the BRIP algorithm requires the following assumptions:

i) **Assumption of SODPA (AOS):** By the profiling stage, an attacker must accurately know the time when an operand is loaded or manipulated in the process of multiplication computation.

ii) **Assumption of weak target device (AOWT):** When attackers can collect only a limited number of power signals, the power signals must not be affected by additional countermeasures, such as insertion noise, random time delay, or random clocks.

For the success of the SODPA like the Okeya-Sakurai style attack, an attacker must know some information needed in analysis. While it may be easy for developers of a target device to acquire this information, attackers without any knowledge of the device must use a lot of time to carry out various tests, such as power fluctuation tests according to input values. This is the first step to achieving AOS. In addition, the SODPA must be conducted to recover a secret key under AOWT because that is executed with the differential power trace only.

The practical SOCPA on the BRIP algorithm which we propose in this paper can also be applied to circumstances that do not even satisfy AOS and AOWT.

### 1. Attack Scenario

The BRIP algorithm for RSA is composed of two exponentiations according to step 3 and one main exponentiation according to step 6 in algorithm 3. The target position of the proposed attack is the main exponentiation related to the secret key. That is, we find the secret exponent $d$ in non-CRT RSA and $d_p$ or $d_q$ in CRT RSA. Finding $d_p$ or $d_q$ means that the secret primes $p$ or $q$ are exposed by $gcd(((r^{dp})^e \bmod n) - r, n)$, where $r$ is the random number. The scenario of our attack progresses as described in the following subsections.

#### *A. Collecting Power Traces*

At first, an attacker acquires $N$ power traces by carrying out the BRIP algorithm for RSA. At this time, we never consider plaintext or ciphertext. That is, our attack does not require the attacker to know any plaintext or ciphertext.

#### *B. Preprocessing*

In algorithm 3, one operand of multiplication is decided by a processed bit of the secret key. If the bit is 0, $U_0$ is manipulated; otherwise, $U_1$ is manipulated in step 6.2. This weakness can be applied to most left-to-right-type exponentiation algorithms.

Since our attack is only concerned about multiplication while ignoring squaring, an attacker reconstructs new power traces by extracting and concatenating the signals of multiplication related to step 6.2 from power traces produced during the main exponentiation of algorithm 3. We define the newly reconstructed $i$-th power trace as $C_i$ $(1 \leq i \leq N)$. In addition, we define points of $C_i = M_{i,0} || M_{i,1} || \cdots || M_{i,n-1}$ as $C_i = (c_{i,0} \cdots c_{i,1*L-1}) || (c_{i,1*L} \cdots c_{i,2*L-1}) || \cdots || (c_{i,(n-1)*L} \cdots c_{i,n*L-1})$, where $M_{i,k}$ $(0 \leq k \leq n-1)$ is the multiplication signal corresponding to the ($k$+1)th action of step 6.2 in the $i$-th power trace, $L$ is the length of power signal for multiplication, and $n$ is the bit length of the secret key.

#### *C. Acquisition of the Second-Order Correlation Traces*

To extract the secret key, we can compute the second-order correlation power traces from reconstructed power traces $C_i$ $(1 \leq i \leq N)$. The correlation trace $CT_i$ $(2 \leq i \leq n)$ implies the relation between the multiplication corresponding to the MSB of the secret key and the other multiplication corresponding to the $i$-th bit. Algorithm 4 describes steps to acquire correlation traces $CT_2, CT_3, \cdots, CT_n$.

#### *D. Exposure of the Secret Key*

In algorithm 4, we can instinctively anticipate that correlation traces will show high-correlation coefficients for the bits that have the same value as the MSB of the secret key. Thus, when loading or manipulating an operand of the multiplication, a high correlation coefficient will be shown. We refer to these times as points of interest. However, it is impossible to distinguish points of interest visually in some cases that do not

```
Algorithm 4. Acquisition of the second-order correlation traces
Input: $C_1, C_2, \cdots, C_N$ where $C_i=(c_{i,0}\cdots c_{i,n*L-1})$.
Output: Second-order correlation traces $CT_2, CT_3, \cdots, CT_n$ where
        $CT_i=(\xi_{i,0}, \xi_{i,1}, \cdots, \xi_{i,L-1})$.
  1. For $j$=0 to $L-1$ do
    1.1. Set the set $XT$ as $\{c_{1,j}, c_{2,j}, \cdots, c_{N,j}\}$.
    1.2. For $i$=2 to $n$ do
      1.2.1. Set the set $YT$ as $\{c_{1,(i-1)*L+j}, c_{2,(i-1)*L+j}, \cdots, c_{N,(i-1)*L+j}\}$.
      1.2.2. Generate $\xi_{i,j}$ by computing the correlation coefficient
             of $XT$ and $YT$.
  2. Return $CT_i=(\xi_{i,0}, \xi_{i,1}, \cdots, \xi_{i,L-1})$ $(2\leq i\leq n)$.
```

satisfy AOWT. The reason is that the points of interest are spread over several consecutive cycles rather than being centralized in a specific time.

To detect the interval of interest if AOWT is not satisfied, we use the variance between correlation traces. The basic idea is that the correlation values for the interval of interest are divided into two groups, even if they cannot be distinguished by direct observation. One group is composed of relatively high correlation coefficients, and the other group consists of relatively low correlation coefficients. However, in intervals outside the interval of interest, correlation coefficient values are almost the same. That is, the variance value between correlation traces has a bigger value in the interval of interest. A way to find the points of interest is shown in algorithm 5.

```
Algorithm 5.  Finding points of interest
Input: Second-order correlation traces $CT_k, CT_{k+1}, \cdots, CT_n$.
Output: $s$ points of interest.
1. Compute the variance trace $V(CN)$ of $CT_k, CT_{k+1}, \cdots, CT_n$ where
   $V(CN)=(v_0 v_1 \cdots v_{L-1})$ and $v_i$ is the variance of $\{\xi_{k,i}, \xi_{k+1,i}, \cdots, \xi_{n,i}\}$.
2. Select points of interest corresponding to max $s$ values among
   $\{v_0, v_1, \cdots, v_{L-1}\}$.
3. Return the points selected in step 2.
```

After finding the points of interest, the remaining step is straightforward. Finally, to find the secret key, we sum up correlation coefficients corresponding to visually distinguished points or to the points of interest acquired by algorithm 5.

## 2. Experimental Results

### A. Experimental Environments

We carried out experiments on a smartcard having a dedicated co-processor equipped with the radix-4 Montgomery multiplier. The Montgomery multiplier implemented in our target device does not require any reduction operation in step 3 of algorithm 1. Therefore, the target device with this Montgomery multiplier is resistant against attacks such as the Schindler-style timing attack algorithm [18].

In step 6.2 of algorithm 3, Mont($U$,$U_{di}$) could be designed as Mont($U_{di}$,$U$) by the intention of the designers. To be precise, two operands of the Montgomery multiplier carry out different functions. In Mont($A$,$B$), the word value $b_i$ of the second operand $B$ decides the multiple $b_iA$ of the first operand $A$. That is, at each iteration of algorithm 1, $b_i$ of $B$ is scanned and then the value of $b_iA$ is added to the accumulator. This means that the result of our second-order CPA can differ according to the position of operands in step 6.2 of algorithm 3. Therefore, we proceed with the experiment as two cases, namely, Mont($U$,$U_{di}$) and Mont($U_{di}$,$U$). We also carry out experiments by applying a random clock as an additional countermeasure.

### B. Experimental Results for the Fixed Values in the Second Operand

As mentioned in the previous section, the first three steps of our attack–collecting power signals, pre-processing, and acquisition of the second-order correlation traces–should be performed whether AOWT is satisfied or not. However, in the final step (exposure of the secret key) if we can discern the visible high correlation peaks, then we can directly expose the secret key by observing second-order correlation traces. Otherwise, we have to execute algorithm 5.

First, when $U$=Mont($U$, $U_{di}$) is equipped with fixed clocks, Fig. 1 shows the second-order correlation traces for the most significant 16 bits of the secret key, that is, 0x3ABA.

By considering the correlation trace $CT_2$ of Fig. 1, we find that the second MSB of the secret key is the same as the first bit. On the other hand, we can know that the third MSB is different from the first MSB. Therefore, we can decide the 16 most significant bits of the secret key as either 0x3ABA or 0xC545. To find the 16 most significant bits, we used 200 power traces, and the high correlation peaks could be visually discerned by direct observation. To test the minimum number of power

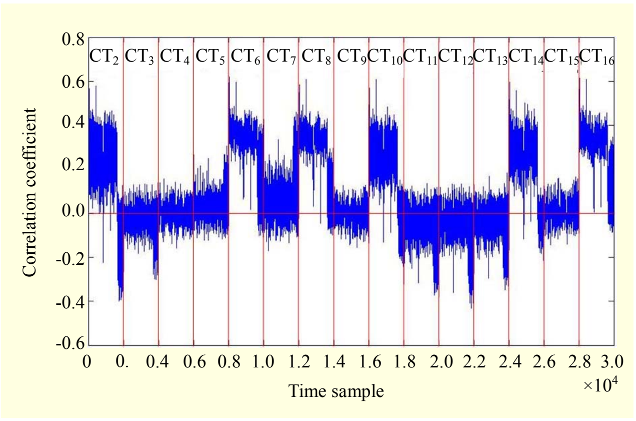

Fig. 1. Second-order correlation traces using 200 power traces for $U$=Mont($U$, $U_{di}$) equipped with fixed clocks.

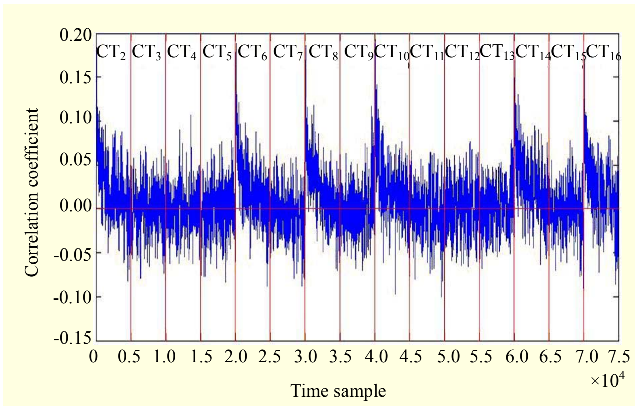

Fig. 2. Second-order correlation traces using 1,000 power traces for $U$=Mont($U$, $U_{d_i}$) equipped with internal random clocks.

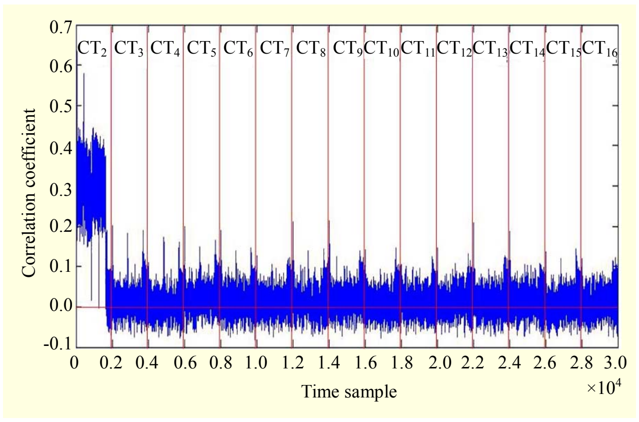

Fig. 3. Second-order correlation traces using 200 power traces for $U$=Mont($U_{d_i}$, $U$) equipped with fixed clocks.

traces required for a successful attack, we only used 20 power traces by summing up all the correlation coefficients corresponding to each multiplication signal.

Second, we consider what happens when an internal random clock is applied. Figure 2 shows the correlation traces. Similarly, we can classify the same bits and the different bits compared to the first bit. In the case of inserting a random clock during computation of exponentiation, we used 1,000 power traces to find the entire 1,024 bit key.

If a random clock is used, the minimum number of traces needed for recovering the secret key was only 70 when we sum up all the correlation coefficients corresponding to each multiplication signal. These results mean that our attack is very practical and can find the secret key with only a few power traces.

### *C. Experimental Results for the Fixed Values in the First Operand*

Considering the computation of $U$=Mont($U_{d_i}$, $U$) equipped with fixed clocks, the second-order correlation traces for the 16 most significant bits of the secret key are shown in Fig. 3.

In Fig. 3, only $CT_2$ has relatively high correlation coefficients, but the others (from $CT_3$ to $CT_{16}$) have low correlation coefficients. This means that the first MSB of the secret key is 0, and the second MSB is 1. In algorithm 3, the first action of step 6.2 carries out the computation of $R^{v+1}$=Mont($U_0$, $R^{2v+1}$) when the first MSB is 0. On the other hand, the second action carries out the computation of $XR^{v+1}$=Mont($U_1$, $R^{2v+1}$) when the second MSB is 1. Based on these facts, $CT_2$ has high correlation coefficients because the second operands of both multiplications are identical to each other. If the second MSB is 0, the second action of step 6.2 is $R^{v+1}$=Mont($U_0$, $R^{2v+1}$), and the third action is $R^{v+1}$=Mont($U_0$, $R^{2v+1}$) or $XR^{v+1}$=Mont($U_1$, $R^{2v+1}$). In this case, $CT_3$ also has relatively high correlation coefficients.

We can know the most significant consecutive 0 bits of the secret key by checking continuous sections that have high correlation coefficients. However, it is difficult to decide the next remaining bits by direct observation. Therefore, we have to apply algorithm 5 to $CT_3$, $CT_4$,⋯, $CT_{1,024}$ in order to find points of interest. Figure 4 shows the variance trace $V(CN)$.

In Fig. 4, we selected from 100 to 200 points and from 300 to 400 points having the highest variance values as the points of interest. To find the secret key, we summed up correlation coefficients within two intervals from 100 to 200 and from 300 to 400 in each correlation trace, namely from $CT_2$ to $CT_{1,024}$. Figure 5 shows the partial result when summing up correlation coefficients in two intervals in each correlation trace from $CT_2$ to $CT_{32}$.

In Fig. 5, the $i$-th bar is the sum of the correlation coefficients $CT_{i+1}$ of interesting points. If a higher value than the specific threshold is observed, this bit and the first bit have the same value. Otherwise, this bit is different from the first bit. Because the first MSB is 0, the third MSB, indicated by the second bar in Fig. 5, is 0, and the fourth MSB, indicated by the third bar, is

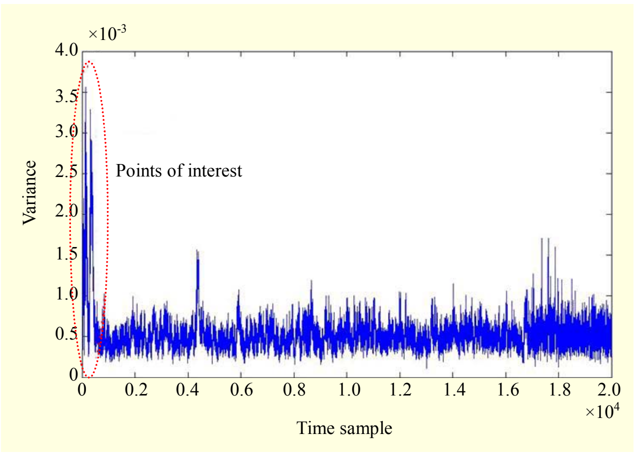

Fig. 4. Finding points of interest from the correlation trace of Fig. 3.

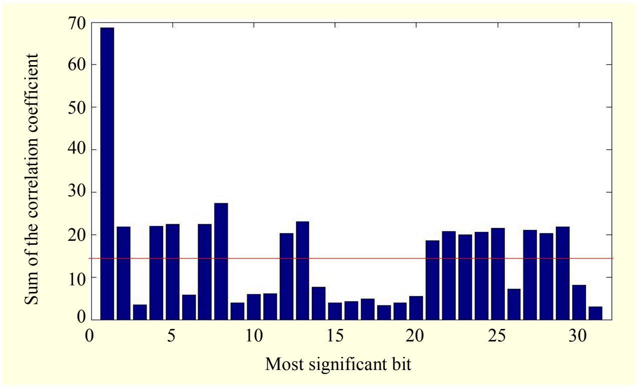

Fig. 5. Finding the most significant 32 bits (0x5273f823) of the private key by summing correlation coefficients of interesting points.

1. In this case, we were able to find the 1,024-bit secret key with about 800 power traces.

Finally, we consider the case in which a random clock is used. Similar to the case in which a fixed clock is used, we first find points of interest and then compute the summation of correlation coefficients of interesting points. After finishing this work, we can classify the same bits and the different bits from the first MSB. However, in this case, we need many more traces than when a fixed clock is used. Therefore, we found only the first 128 bits of the secret key using about 11,000 power traces. Since our attack requires no knowledge of plaintext or ciphertext and uses only the relation of whether two consecutive bits are the same or not, we can find the entire 1,024-bit key by executing the same attack that found a 128-bit key 8 more times.

We assumed that the points of interest were found and then tried the Okeya-Sakurai style SODPA in order to compare it with our SOCPA. We needed more than 50 traces and 500 traces in case of Mont($U$, $U_{d_i}$), using a fixed clock and a random clock, respectively. However, we could find only a few bits of the secret key using over 15,000 traces in the case of Mont($U_{d_i}$, $U$). That is, we failed to recover the entire 1,024-bit key by using Okeya-Sakurai style SODPA. This result means that our attack method is far more efficient and practical than the previously proposed method. Furthermore, it makes the process of finding points of interest more automated.

Based on Table 1, we can conclude that the correlation when the second operand is the same is higher than that when the first operand is the same.

### 3. Extension to Montgomery Ladder Method

In the previous subsection, we described the SOCPA against the BRIP algorithm. Now, we extend the SOCPA against the ML method, which is secure against SPA combined with the

Table 1. Number of traces needed to recover a 1,024-bit key with Okeya-Sakurai style SODPA and proposed SOCPA.

| | Mont($U$, $U_{d_i}$) | | Mont($U_{d_i}$, $U$) | |
|---|---|---|---|---|
| | Fixed clock | Random clock | Fixed clock | Random clock |
| Okeya-Sakurai style SODPA | 50 | 500 | n/a | n/a |
| Our SOCPA | 20 | 70 | 800 | 88,000 |

message blinding method. The algorithm that is secure against SPA introduced in [5] is shown here as algorithm 6. In this algorithm, we omit the method which makes the message blind because blinding methods can be applied variously. However, our method can always be successful independent of blinding methods.

```
Algorithm 6. Montgomery ladder method
Input: $X$, $m$, and $d = (d_{n-1}d_{n-2}\cdots d_1d_0)_2$, where $n$ is the size of modulus
    $m$.
Output: $X^d$ mod $m$.
  1. Compute $U_0$=Mont($X$, $R^2$) and $U_1$=Mont($U_0$, $U_0$).
  2. For $i=n-2$ down to 0 do
    2.1. $U_{1-d_i}$=Mont($U_0$, $U_1$).
    2.2. $U_{d_i}$=Mont($U_{d_i}$, $U_{d_i}$).
  3. Return Mont($U_0$,1).
```

Steps 2.1 and 2.2 in the main loop compute Mont($U_0$, $U_0$) and Mont($U_0$, $U_1$) for $d_i$=0 and Mont($U_1$, $U_1$) and Mont($U_0$, $U_1$) for $d_i$=1. That is, the second operands are different for $d_i$=0 and equal for $d_i$=1. As mentioned before, when the second operand is the same, the correlation is higher than for the opposite case. Therefore, we can check whether a bit is zero or one by computing the correlation coefficient between multiplications in step 2.1 and 2.2 at each loop. Correlation traces for $d_i$=0 (upper) and $d_i$=1 (lower) are shown in Fig. 6.

## IV. New Countermeasure

Our attack can break the BRIP algorithm proposed to prevent SPA, DPA, and RPA because the BRIP algorithm manipulates the same operand when each bit of the secret key is equal. Most of the left-to-right exponentiation algorithms have this kind of weakness. Therefore, we propose a secure and efficient countermeasure based on the ML technique which aims at removing the dependency between the key bit and processed operand. The proposed exponentiation algorithm is shown as algorithm 7.

Algorithm 7 is based on the ML method. However, directly

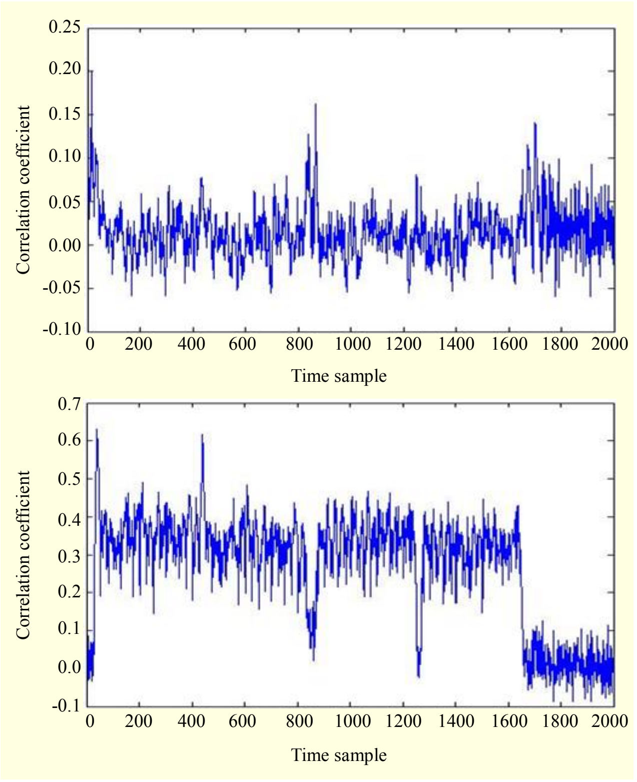

Fig. 6. Correlation traces in the Montgomery ladder, where the upper graph is the correlation coefficient for $d_i$=0, and the lower graph is the correlation coefficient for $d_i$=1.

```
Algorithm 7. Proposed countermeasure against SODPA
Input: $X$, $m$, $d = (d_{n-1}d_{n-2}\cdots d_1d_0)_2$, $z$ and $z^{-2^n}$, where $n$ is the size of modulus $m$.
Output: $X^d \bmod m$.
1. If $X = 1$ then return 1. Else if $X = -1$ then return $1 - 2d_0$.
2. Generate $a$ random value $v$.
3. Compute $U_0 = z^v \bmod m$ and $U = (z^{-2^n})^v \bmod m$ using SPA resistant exponentiation.
4. Compute $U_0$=Mont($U_0$, $R^2$),$U_1$=Mont($U_0$, Mont($X$, $R^2$) ).
5. For $i=n-1$ down to 0 do
    5.1. Generate the random bit $r$.
    5.2. $U_2$=Mont($U_{d_i}$, $U_r$).
    5.3. $U_{1-r}$=Mont($U_{1-r}$, $U_{d_i}$).
    5.4. $U_r$=$U_2$.
6. Return Mont($U_0$, $U$).
```

applying the ML method to the message blinding method is also insecure against our attack. Thus, we have to modify this algorithm. In algorithm 6, two multiplications are composed of $U_{1\text{-}d_i}$=Mont($U_0$, $U_1$) and $U_{d_i}$=Mont($U_{d_i}$, $U_{d_i}$). We can find the secret information from the correlation coefficient trace between two multiplications because the location of two operands $U_0$ and $U_1$ are fixed during the computation of $U_{1\text{-}d_i}$=Mont($U_0$, $U_1$).

The first feature of the proposed algorithm is to change the location of two operands $U_0$ and $U_1$ of $U_{1\text{-}d_i}$=Mont($U_0$, $U_1$) randomly. This is enough to defend against our attack. However, we try to eliminate any dependency between computed values in the previous bit and loaded values in the present bit. The second feature of our countermeasure is to change the order of two operations $U_{1\text{-}d_i}$=Mont($U_0$, $U_1$) and $U_{d_i}$=Mont($U_{d_i}$, $U_{d_i}$) randomly.

Table 2. Flow of computation in our algorithm and the randomness of the operation order including the location of operands.

| $d_i$ | $r$ | Montgomery ladder | Algorithm 7 |
|---|---|---|---|
| 0 | 0 | $U_1$=Mont($U_0$, $U_1$)<br>$U_0$=Mont($U_0$, $U_0$) | $U_2$=Mont($U_0$, $U_0$)<br>$U_1$=Mont($U_1$, $U_0$)<br>$U_0$=$U_2$ |
| | 1 | | $U_2$=Mont($U_0$, $U_1$)<br>$U_0$=Mont($U_0$, $U_0$)<br>$U_1$=$U_2$ |
| 1 | 0 | $U_0$=Mont($U_0$, $U_1$)<br>$U_1$=Mont($U_1$, $U_1$) | $U_2$=Mont($U_1$, $U_0$)<br>$U_1$=Mont($U_1$, $U_1$)<br>$U_0$=$U_2$ |
| | 1 | | $U_2$=Mont($U_1$, $U_1$)<br>$U_0$=Mont($U_0$, $U_1$)<br>$U_1$=$U_2$ |

Table 2 presents the correctness of our proposed countermeasure and the randomness of the order of operations including the location of operands.

In our algorithm, the input message $X$ is blinded by a random value $z^v$, where $z$ is the random value stored in the ROM. We can know the output value after exponentiation because our algorithm always computes two multiplications, $U_{1\text{-}d_i}$= Mont($U_0$, $U_1$) and $U_{d_i}$=Mont($U_{d_i}$, $U_{d_i}$) in step 5. After finishing exponentiation, the blinding factor becomes $(z^v)^{2^n}$; therefore, we have to remove the random blinding factor to obtain a correct value. To remove this blinding factor, the random value $z^{-2^n}$ must be stored inside a smart card, and $z^v$ and $(z^{-2^n})^v$ are initially computed at each new execution by computing exponentiation using a 48-bit random exponent $v$.

## 1. Security Analysis

As mentioned before, our algorithm is basically designed to combine the ML method with the message blinding method. Therefore, our algorithm is secure against SPA, DPA, and RPA.

Next, we consider the security against SODPA or SOCPA. In our algorithm, when $d_i$=0, $U_2$=Mont($U_0$, $U_0$) and $U_1$=Mont($U_1$, $U_0$) for $r$ = 0, and $U_2$=Mont($U_0$, $U_1$) and $U_0$=Mont($U_0$, $U_0$) for $r$ = 1. The correlation coefficient for $r$=0 is higher than that for $r$=1, because the second operand is the same when $r$=0. Therefore, if we compute correlation coefficients between the first multiplication and the second one, the peak of the correlation coefficient is reduced by half compared with the case of $r$=0 because of the random decision of $r$. In the other case, when $d_i$=1, $U_2$=Mont($U_1$, $U_0$) and $U_1$=Mont($U_1$, $U_1$) for $r$ = 0, and $U_2$=Mont($U_1$, $U_1$) and $U_0$=Mont($U_0$, $U_1$) for $r$ = 1. For the same reason, when $d_i$=0 the peak of the correlation coefficient is reduced by half compared with the case when $r$=1. In conclusion, the peaks are always the same, independent of the secret key bit. Figure 7 shows correlation traces for $d_i$=0 (upper) and $d_i$=1 (lower). Therefore, our countermeasure is secure against the proposed SOCPA as well as SPA, DPA, RPA, and SODPA.

## 2. Efficiency Comparison

Our algorithm has almost the same computational cost as the BRIP algorithm. Although our algorithm requires additional ROM for $z$ and $z^{-2^n}$, ROM is comparatively sufficient in crypto devices. Both algorithms require two exponentiations by $v$ and one main exponentiation using the secret key $d$. Let $t$ and $n$ be the bit sizes of $v$ and $d$. Then, the computational cost of the BRIP method is $3tM + 2nM + 5M$, where we assume that the exponentiation by $v$ is computed with a binary method secure against SPA, and $M$ denotes the computational time of a multiplication. The computational cost of our method is $3tM + 2nM + 4M$.

The exponent splitting (ES) $X^r X^{d-r}$ mod $m$ requires two

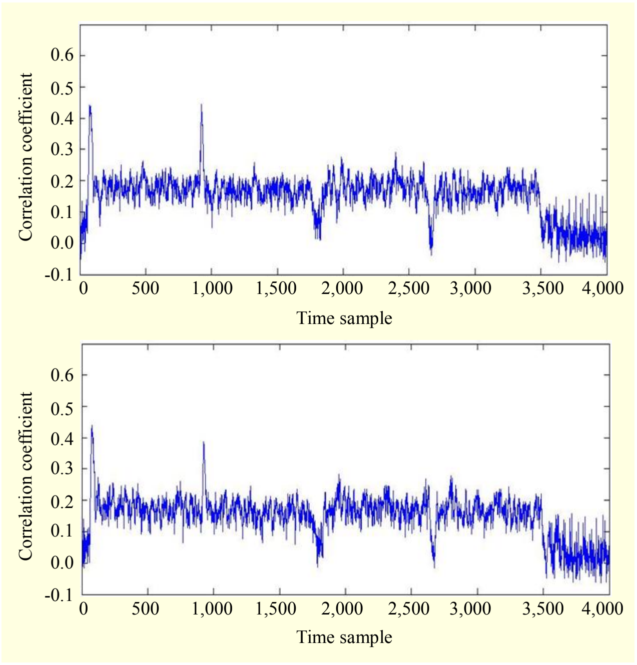

Fig. 7. Correlation traces in our algorithm, where the upper trace is the correlation coefficient for $d_i$=0 and the lower trace is the correlation coefficient for $d_i$=1.

Table 3. Performance comparison of the proposed algorithm with other algorithms for a 1024-bit key.

| | BRIP | ES | IES | Ours |
|---|---|---|---|---|
| Computational cost | 2,197M | 3,072M | 1,664M | 2,196M |
| Inverse requirement | No | No | Yes | No |
| Additional ROM | No | No | No | 2×1,024 bits |

exponentiations. If it uses the binary method using atomicity [3], then the computational cost is $3nM$. In the case of improved exponent splitting (IES) $(X^r)^{[d/r]}X^{(d \bmod r)}$ mod $m$, if it uses the simultaneous method for two exponentiations $(X^r)^{[d/r]}$ mod $m$ and $X^{(d \bmod r)}$ mod $m$, and the size of a random number $r$ is $n/2$, then the computation cost is $13/8nM$. However, it requires the inverse of $r$ which takes a long computation time and large memory.

Table 3 shows a performance comparison of our algorithm with other countermeasures. In this table, we suppose that the length of the random number $v$ in both the BRIP algorithm and our algorithm is 48 bits. Our algorithm requires more multiplications than the IES method; however, since the IES method requires the computation of inverse values, our algorithm is more suitable for RSA cryptosystems.

## V. Conclusion

In this paper, we have proposed a practical SOCPA. The first advantage of our attack is that it is able to find points of interest easily, which is the most fundamental issue in mounting a high-order DPA. Namely, our attack method can work without AOS because no profiling stage is required. The second advantage of our attack is that it can find the secret key from a smaller number of power traces, compared with the previous SODPA. In addition, our attack does not require any knowledge about plaintext or ciphertext. Therefore, our attack can be easily applied to CRT RSA and various algorithms such as the basic ML method.

We have also proposed an efficient countermeasure against the proposed SOCPA as well as SPA, DPA, RPA, and SODPA. We have constituted the ML type method equipped with the message blinding method, which does not require any additional computation, including inverse computation.

## References

[1] P. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis,” *CRYPTO*, LNCS 1666, 1999, pp. 388-397.
[2] T. Messerges, E. Dabbish, and R. Sloan, “Power Analysis Attacks

of Modula Exponentiation in Smartcards," *CHES*, LNCS 1717, 1999, pp. 144-157.

[3] 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.

[4] J.S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," *CHES*, LNCS 1717, 1999, pp. 292-302.

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

[6] F. Amiel, B. Feix, and K. Villegas, "Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms," *SAC*, LNCS 4876, 2007, pp. 110-125.

[7] K. Okeya and K. Sakurai, "Power Analysis Breaks Elliptic Curve Cryptosystems Even Secure against the Timing Attack," *INDOCRYPT*, LNCS 1977, 2000, pp. 178-190.

[8] C. Clavier and M. Joye, "Universal Exponentiation Algorithm: A First Step towards Provable SPA-Resistance," *CHES*, LNCS 2162, 2001, pp. 300-308.

[9] M. Ciet and M. Joye, "(Virtually) Free Randomization Technique for Elliptic Curve Cryptography," *ICICS*, LNCS 2836, 2003, pp. 348-359.

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

[11] K. Itoh, T. Izu, and M. Takenaka, "Improving the Randomized Initial Point Countermeasure against DPA," *ACNS*, LNCS 3989, 2006, pp. 459-469.

[12] F. Amiel and B. Feix, "On the BRIP Algorithms Security for RSA," *WISTP*, LNCS 5019, 2008, pp. 136-149.

[13] K. Okeya and K. Sakurai, "A Second-Order DPA Attack Breaks a Window Method Based Countermeasure against Side Channel Attacks," *ISC*, LNCS 2433, 2002, pp. 389-401.

[14] P.A. Fouque and F. Vallette, "The Doubling Attack: Why Upwards Is Better than Downwards," *CHES*, LNCS 2779, 2003, pp. 269-280.

[15] A.C. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1996.

[16] E. Oswald et al., "Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Cipher," *CT-RSA 2006*, LNCS 3860, 2006, pp. 192-207.

[17] E. Oswald and K. Schramm, "An Efficient Masking Scheme for AES Software Implementation," *WISA 2005*, LNCS 3786, 2006, pp. 292-305.

[18] W. Schindler, "A Timing Attack against RSA with the Chinese Remainder Theorem," *CHES*, vol. 1965, 2000, pp. 109-124.

**HeeSeok Kim** received his BS degree in mathematics from Yonsei University, Seoul, Rep. of Korea, in 2006, and the MS degree in engineering in information security from Korea University, Seoul, Rep. of Korea, in 2008. He is currently a PhD student with Korea University. He is also a researcher with the Center for Information Security Technologies (CIST). His research interests include side channel attacks and public-key and symmetric-key cryptosystems.

**Tae Hyun Kim** received the BS degree in mathematics from the University of Seoul, Seoul, Rep. of Korea, in 2002, and the MS degree in engineering in information security from Korea University, Seoul, Rep. of Korea, in 2004. He was a visiting researcher with the School of Systems Information Science, Future University, Hakodate, Japan, from April to September 2006. He received his PhD in engineering in information security from Korea University, Seoul, Rep. of Korea, in 2009. Since August 2009, he has been a researcher with the Attached Institute of Electronics and Telecommunications Research Institute (ETRI), Daejeon, Rep. of Korea.

**Joong Chul Yoon** received the BS, MS, and PhD degrees in mathematics from Korea University in 1993, 1995, and 1999, respectively. He worked for SECURITY Technologies Inc. from 1999 to 2002. From 2002 to 2003, he worked as a postdoctoral researcher with the Center for Information Security Technologies (CIST). Since 2003, he has been working for the System LSI Division of Samsung Electronics. Currently, his main research interests are side channel attacks, public key cryptosystems, and related security issues.

**Seokhie Hong** received his Master's and PhD degrees in mathematics from Korea University in 1997 and 2001, respectively. He worked for SECURITY Technologies Inc. from 2000 to 2004. From 2004 to 2005, he worked as a postdoctoral researcher with COSIC at KU Leuven, Belgium. Since 2005, he has been with Korea University, where he is now working in the Graduate School of Information Management and Security. His specialty lies in the area of information security, and his research interests include the design and analysis of symmetric-key cryptosystems, public-key cryptosystems, and forensic systems.