Power Analysis Attacks on the Right-to-Left Square-Always Exponentiation Algorithm

Jaecheol Ha[1]*, Yongje Choi[2], Dooho Choi[2], and Hoonjae Lee[3]
[1]Hoseo University, Asan, ChungNam, Korea
jcha@hoseo.edu
[2]ETRI, Daejeon, Korea
{choiyj, dhchoi}@etri.re.kr
[3]Dongseo University, Busan, Korea
hjlee@dongseo.ac.kr

Abstract

The naive implementation of an exponentiation used in public key cryptography may reveal a secret key to the attacker by several side-channel attacks. Recently, a novel square-always exponentiation algorithm based on trading multiplications for squarings is proposed. This algorithm for RSA implementation is faster than existing regular countermeasures against side-channel attacks. This paper suggests that the right-to-left square-always exponentiation algorithm is vulnerable to some side-channel attacks: collision distance-based doubling, chosen-message SPA, and CPA-based combined attacks. The chosen-message SPA attack can be intactly applied to this algorithm. The other two attacks are variants of the doubling attack and SPA-based combined attack, respectively. In addition, the paper presents an improved right-to-left square-always algorithm resistant to existing and proposed power analysis attacks by using the additive message blinding method and the message update technique before the main iterative operation.

Keywords: RSA Cryptosystem, Side-Channel Attack, Simple Power Analysis, Square-Always Exponentiation

1 Introduction

To provide security using embedded devices, several cryptographic algorithms are typically implemented on general-purpose processors, co-processors, smart card chips, and so on. Although a cryptographic algorithm is securely considered against traditional cryptanalysis, it can be vulnerable to a new type of cryptanalysis called the side-channel attack. Based on the side-channel attack, an adversary can recover the secret key by exploiting the leakage of side-channel information (e.g., power consumption, electromagnetic emanation, and the operating time), which is produced during the execution of a cryptographic algorithm embedded in a physical device.

Embedded security devices in which the secret key is usually stored operate in hostile environments where the key can be stolen or manipulated by malicious users. In particular, if the attacker already knows implementation information such as the cryptographic algorithm, the chip design technology, and device specifications, then the side-channel attack can be more easily attempted. An unreliable insider with knowledge of the internal state and implementation conditions can easily manipulate acquired cryptographic devices. From this perspective concerning the side-channel attack, a malicious insider is a more serious threat than an outsider.

Journal of Internet Services and Information Security (JISIS), volume: 4, number: 4 (November 2014), pp. 38-51
*Corresponding author: Dept. of Information Security, Hoseo University, 20, Hoseo-ro 79-Gil, Babang-Eup, Asan-Si, Chungnam, 336,795, Korea, Tel: +82-(0)41-540-5991, Web: http://islab.hoseo.ac.kr/jcha

RSA (Rivest, Sharmir, and Adelman) [15] is well known to be the most widely used public key cryptosystem in embedded security devices. More precisely, the computation for an RSA signature or encryption is based on a modular exponentiation composed of hundreds of squarings and multiplications. The power analysis attack is an important category of side-channel attacks proposed by Kocher [13], who introduce both SPA (Simple Power Analysis) and DPA (Differential Power Analysis) attacks on the exponentiation algorithm. Since then, many studies have introduced new attack techniques for different cryptographic algorithms and attempted to improve the processing of power signals to extract the secret key by using fewer traces than existing attacks [8, 14, 3].

Indeed, an attacker measures the power dissipation of a single trace in an SPA attack and extracts the secret key by using this physical information. To protect against an SPA attack, many researchers have proposed various exponentiation algorithms as countermeasures, including the square-and-multiply-always algorithm [8], the Montgomery ladder [12], the Joye ladder [11], and the so-called atomic exponentiation algorithm[4]. These methods for countering SPA attacks basically allow computational operations independently from the value of secret key bits.

Recently at Indocrypt 2011, Clavier et al. [6] present a novel exponentiation method based on trading multiplications for squarings. Their exponentiation uses only squarings to avoid exploiting any side-channel leakage to an attacker by distinguishing between multiplications and squarings during the execution of the classical exponentiation [1]. They also present two types of exponentiation algorithms: left-to-right and right-to-left methods. Indeed, the right-to-left square-always exponentiation algorithm is recommended for practical implementation because the left-to-right algorithm is vulnerable to the doubling attack [10], the chosen-message SPA attack [20], and combined attack [2].

This paper suggests that the right-to-left square-always algorithm is also vulnerable to three side-channel attacks: the so-called collision distance-based doubling attack, the chosen-message SPA attack, and CPA-based combined attack. Among these, the chosen-message SPA attack can be intactly applied to this algorithm. The other two attacks are variants of the doubling attack and SPA-based combined attack, respectively. In addition, the paper proposes an improved exponentiation algorithm resistant to existing and proposed side-channel attacks.

The rest of this paper is organized as follows: Section 2 briefly recalls the exponentiation algorithm for RSA implementation and several side-channel attacks on it. Section 3 presents some side-channel attacks on the right-to-left square-always exponentiation algorithm. Section 4 presents the improved exponentiation algorithm resistant to existing and proposed side-channel attacks, and Section 5 concludes.

2 Side-Channel Attacks on the RSA Exponentiation

This section recalls the exponentiation algorithm for implementing the RSA cryptosystem, introduces side-channel attacks on existing exponentiation methods, and describes two atomicity-based countermeasures and their principles to resist a dedicated SPA attack.

2.1 RSA and Exponentiation

RSA is a popular public key cryptosystem for encrypting or signing a given message. In the RSA system, two prime integers p and q are generated, and the public modulus n is their product. Here let e be the public key and d be the corresponding secret key such that $e \cdot d = 1 \bmod (p-1)(q-1)$. The classical signing with RSA for a message m consists of computing the value $s = m^d \bmod n$. Here, the signer uses the secret key d as an exponent of this exponentiation. After s and m are sent to the verifier, the signature s is verified by checking that $s^e \bmod n$ is equal to m.

An exponentiation is basically a sequence of multiplications and squarings. Two types of exponentiation algorithms are usually considered according to the order of exponent scans. The first type starts from the most significant bit and works downward. This method is called the left-to-right algorithm. By contrast, the second one starts from the least significant bit and works upward and is known as the right-to-left algorithm. As already highlighted in [6] and [10], the right-to-left implementation is more resistant to many side-channel attacks than the left-to-right one. Therefore, this paper focuses on the strong implementation of the right-to-left exponentiation algorithm.

2.2 SPA Attack and its Variants

The final goal of an attacker breaking the RSA cryptosystem is to recover the secret key d from a side-channel leakage. This secret key is used as an exponent of the exponentiation for computing an RSA signature or decryption for message m.

SPA Attack In an RSA exponentiation, an SPA attack consists of observing that if the multiplication operation has a different power consumption pattern from the one for the squaring operation, then the secret key can be recovered from a single power trace. The so-called square-and-multiply algorithm is typically used to perform an RSA exponentiation. However bits of the secret key are directly retrieved from the observed trace if the square-and-multiply algorithm is naively implemented. That is, this exponentiation algorithm is vulnerable to an SPA attack [1].

The classical countermeasures consist of using the so-called *regular* algorithm such as the square-and-multiply-always and Montgomery ladder algorithms or applying the *atomicity* principle, as described in detail in the next section. In addition, the square-and-multiply-always algorithm is vulnerable to the C-Safe error attack [18], an active attack to extract the secret key by injecting some faults during the exponentiation operation.

Doubling Attack In a different type of SPA attack called the doubling attack [10], the adversary requires the execution of at least two exponentiations with the same secret key d. The attack defeats the left-to-right square-and-multiply-always algorithm by using two related input messages m and m^2. The basic idea of this attack is that the attacker may find some collision pairs of intermediate values between two power traces measured after inputting each message. In addition, the Montgomery ladder algorithm is susceptible to the relative doubling attack [19], a special case of the original doubling attack.

Chosen-Message SPA attack Yen *et al.* introduce a new type of SPA attack based on the input of a specially chosen message [20]. This attack can defeat an SPA countermeasure such as the square-and-multiply algorithm by using a single power trace. In their paper, an attacker uses an input message $m = (n-1)$ for a modular exponentiation. In this case, only two values involved in the modular exponentiation are 1 and $(n-1)$. Because these two values have a distinct Hamming weight, three operations 1×1, $1 \times (n-1)$, and $(n-1) \times (n-1)$ generated during the exponentiation have different and distinguishable power traces. Therefore, it is easy to explore the sequence of squarings and multiplications and retrieve the secret key.

Horizontal CPA Attack Recently, Clavier *et al.* present another class of side-channel attacks [7], the horizontal CPA (Collision Power Analysis) attack, which is inspired by the Big Mac attack in Walter [16]. This attack consists of computing a classic statistical treatment method such as correlation factors on several time segments extracted from a single execution power trace of a known message. Here a segment refers to one modular operation such as a multiplication or squaring.

The horizontal CPA attack in [17] can defeat the square-and-multiply-always algorithm with a message blinding countermeasure. Indeed, an attacker uses the observation that M-d-S (Multiplication-discard-Squaring) and M-S (Multiplication-Squaring) in the exponentiation algorithm can be distinguished according to their sharing an operand. Because two operations in M-d-S, namely multiplications and squaring, share an operand, their power leakage is more strongly correlated than those in M-S.

	Alg. 1 : Left-to-Right Multiply-Always Exponentiation
	Input: m, n, $d = (d_k - 1, d_k - 2, \cdot\, d_0)_2$
	Output: $m^d \bmod n$
1.	$R_0 = 1$
2.	$R_1 = m$
3.	$i = k - 1$; $t = 0$
4.	while $i \geq 0$ do
5.	$R_0 = R_0 \times R_t \bmod n$
6.	$t = t \oplus d_i$; $i = i - 1 + t$
7.	return (R_0)

Figure 1: Left-to-right multiply-always exponentiation.

Actually, two operations in M-S do not share an operand. An attacker computes the correlation factor between power measurements for two consecutive segment operations and uses it to recover the secret key. Because this attack requires only one execution trace of the exponentiation, we can defeat it using the message blinding countermeasure or operand randomization in LIM (Long Integer Multiplication) [7].

SPA-based Combined Attack Amiel *et al.* introduce a combined attack on SPA-resistant RSA implementation by combing fault and SPA attacks [2]. Here they target the atomicity-based square-and-multiply algorithm shown in Figure 1, which is the classical SPA-protected implementation. To extract the secret key, the attacker inserts a fault to bypass step 1 in algorithm 1, and then R_0 retains the initial value (typically $R_0 = 0$). As a result of this fault injection, two different power signal patterns appear in step 5 during the main iteration of the exponentiation. The first pattern is the squaring $R_0 = R_0 \times R_0 \bmod n = 0 \times 0 = 0$, which is performed in the case of $t = 0$. The second one is the multiplication $R_0 = R_0 \times R_1 \bmod n = 0 \times m = 0$ in the case of $t = 1$. Because the two power trace patterns are different according to each exponent bit, the attacker can simply recover the secret key by just analyzing the simple power trace.

2.3 Atomic Algorithm

As discussed earlier, the classical countermeasures against the SPA attack can be implemented using a regular algorithm or applying the atomicity principle. The regular exponentiation method includes the square-and-multiply-always, Montgomery ladder, and Joye ladder algorithms. Such regular algorithms perform one multiplication and one squaring during each loop iteration and thus require $1M + 1S$ per secret bit. Here S refers to the cost of a modular squaring, and M, to that of a modular multiplication. In general, $S \approx 0.8M$ in the literature and some experiments on cryptographic processors.

The atomicity principle is presented to protect the square-and-multiply algorithm from the SPA attack. Because the squaring in an exponentiation is performed using a multiplication routine, we call atomic-based exponentiation to multiply-always algorithm. The advantage of the multiply-always algorithm is its better performance in comparison to regular ones. The algorithms in Figures 1 and 2 are two variants of the multiply-always algorithm, which requires $1.5M$ per secret bit on average. Nevertheless, these multiply-always algorithms basically perform a sequence of square-and-multiply algorithms without dummy operations.

Amiel *et al.* show that the Hamming weight of the output of the real multiplication $(x \times y)$ used in the multiply-always algorithm is different from that for the squaring $(x \times x)$ [1]. Although the squaring in the multiply-always algorithm is actually performed by the multiplication routine, an attacker can distinguish that this multiplication performs a squaring $(x \times x)$ or a real multiplication $(x \times y)$. As a

	Alg. 2 : Right-to-Left Multiply-Always Exponentiation
	Input: m, n, $d = (d_k - 1, d_k - 2, \cdot\, d_0)_2$
	Output: $m^d \bmod n$
1.	$R_0 = m$
2.	$R_1 = 1$
3.	$i = 0$; $k = 0$
4.	while $i \leq k - 1$ do
5.	$k = k \oplus d_i$
6.	$R_k = R_k \times R_0 \bmod n$
7.	$i = i + (\neg k)$
8.	return (R_1)

Figure 2: Right-to-left multiply-always exponentiation.

result, this analysis points out that the multiply-always algorithm includes an intrinsic flaw in the square-and-multiply algorithm.

3 Side-Channel Attack on the Square-Always Exponentiation Algorithm

3.1 Right-to-Left Square-Always Exponentiation Algorithm

Clavier *et al.* present in Indocrypt 2011 a novel exponentiation algorithm that has not only the efficiency of the atomicity principle but also immunity against the SPA attack on the multiply-always algorithm [6]. Their countermeasure exponentiation consists of using equation (1) or (2) to compute a multiplication.

$$x \times y = \frac{(x+y)^2 - x^2 - y^2}{2} \tag{1}$$

$$x \times y = \left(\frac{x+y}{2}\right)^2 - \left(\frac{x-y}{2}\right)^2 \tag{2}$$

Because one multiplication can replace two squarings, they combine equation (1) with the left-to-right multiply-always algorithm and apply equation (2) to implement the right-to-left multiply-always algorithm 3. Algorithm 3 in Figure 3 is a right-to-left square-always exponentiation algorithm. The step 6 of algorithm 2 shown in Figure 2 is mainly changed to the loop computations of algorithm 3, from step 7 to 10. Here they use a matrix for more readable and efficient implementation:

$$M = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 2 & 1 \\ 2 & 1 & 2 & 2 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 1 \end{pmatrix} \tag{3}$$

The main loop of algorithm 3 can be described as a four-state machine. The four sequences shown in Figure 4 are operations corresponding to each state. Here the first state $j = 0$ corresponds to the processing of a 0 exponent bit, and $j = 1,\ 2$, and 3 is a 1 bit, as detailed in Figure 4. The main operation of each state is the squaring ${R_{M_{j,3}}}^2 \bmod n$. Dummy operations are identified by the symbol $\star$. For further details, see [6].

	Alg. 3 : Right-to-Left Square-Always Exponentiation
	Input: m, n, $d = (d_k - 1, d_k - 2, \cdots d_0)_2$
	Output: $m^d \bmod n$
1.	$R_0 = m$
2.	$R_1 = 1$
3.	$R_2 = 1$
4.	$i = 0; \quad j = 0$
5.	while $i \leq k-1$ do
6.	$\quad j = d_i(1 + (j \bmod 3))$
7.	$\quad R_{M_{j,0}} = R_{M_{j,1}} + R_0 \bmod n$
8.	$\quad R_{M_{j,2}} = R_{M_{j,3}}/2 \bmod n$
9.	$\quad R_{M_{j,4}} = R_{M_{j,5}} - R_{M_{j,6}} \bmod n$
10.	$\quad R_{M_{j,3}} = {R_{M_{j,3}}}^2 \bmod n$
11.	$\quad i = i + M_{j,7}$
12.	return (R_1)

Figure 3: Right-to-left square-always exponentiation.

$j = 0$ ($d_i = 0$)		$j = 2$ ($d_i = 1$)	
$j = 0$ [$\star$ if j was 0]		$j = 2$	
$R_0 = R_0 + R_0 \bmod n$	$\star$	$R_0 = R_2 + R_0 \bmod n$	$\star$
$R_2 = R_0/2 \bmod n$	$\star$	$R_1 = R_1/2 \bmod n$	
$R_0 = R_0 - R_2 \bmod n$	$\star$	$R_0 = R_0 - R_2 \bmod n$	$\star$
$R_0 = R_0^2 \bmod n$		$R_1 = R_1^2 \bmod n$	
$i = i + 1$		$i = i$	$\star$

$j = 1$ ($d_i = 1$)		$j = 3$ ($d_i = 1$)	
$j = 1$		$j = 3$	
$R_2 = R_1 + R_0 \bmod n$		$R_0 = R_0 + R_0 \bmod n$	$\star$
$R_2 = R_2/2 \bmod n$		$R_0 = R_0/2 \bmod n$	$\star$
$R_1 = R_0 - R_1 \bmod n$		$R_1 = R_2 - R_1 \bmod n$	
$R_2 = R_2^2 \bmod n$		$R_0 = R_0^2 \bmod n$	
$i = i$	$\star$	$i = i + 1$	

Figure 4: A four-state machine of main loop in Algorithm 3

The square-and-multiply algorithm requires $2S$ per exponent bit on average, which theoretically implies an 11.1% speed-up over regular algorithms under the assumption that $S/M = 0.8$. The two algorithms are protected against the SPA attack because of implementation based on the atomicity principle. Because left-to-right algorithms are highly vulnerable to the doubling attack [10] and the chosen-message SPA attack [20] and more likely to be subject to combined attacks [2]], the right-to-left square-always algorithm is generally preferred to the left-to-right version. In addition, according to this paper's additive security analysis, the left-to-right square-always algorithm is susceptible to a horizontal CPA attack.

Table 1: Collision distance-based doubling attack.

i	d_i	Input : m				Input : m^2				Remarks
		$j=0$ $(d_i=0)$	$j=1$ $(d_i=1)$	$j=2$ $(d_i=1)$	$j=3$ $(d_i=1)$	$j=0$ $(d_i=0)$	$j=1$ $(d_i=1)$	$j=2$ $(d_i=1)$	$j=3$ $(d_i=1)$	
0	1		$R_2=R_2^2$	$R_1=R_1^2$	$R_1=m$ $R_0=m^2$		$R_2=R_2^2$	$R_1=R_1^2$	$R_1=m^2$ $R_0=m^4$ ①	Known as odd
1	0	$R_0=m^4$ ①				$R_0=m^8$ ②				Adjacent collision
2	0	$R_0=m^8$ ②				$R_0=m^{16}$				Adjacent collision
3	1		$R_2=R_2^2$	$R_1=R_1^2$	$R_1=m^9$ $R_0=m^{16}$		$R_2=R_2^2$	$R_1=R_1^2$	$R_1=m^{18}$ $R_0=m^{32}$	faraway collision
4	1		$R_2=R_2^2$	$R_1=R_1^2$	$R_1=m^{25}$ $R_0=m^{32}$		$R_2=R_2^2$	$R_1=R_1^2$	$R_1=m^{50}$ $R_0=m^{64}$ ③	faraway collision
5	0	$R_0=m^{64}$ ③				$R_0=m^{128}$				Adjacent collision
6	1		$R_2=R_2^2$	$R_1=R_1^2$	$R_1=m^{89}$ $R_0=m^{128}$		$R_2=R_2^2$	$R_1=R_1^2$	$R_1=m^{178}$ $R_0=m^{256}$	faraway collision
return		$R_1=m^{89}$				$R_1=m^{178}$				

3.2 Vulnerability of the Right-to-Left Square-Always Algorithm

This section points out that the right-to-left square-always algorithm is vulnerable to some side-channel attacks, including other type of doubling attack, the chosen-message SPA attack, and the so-called CPA-based combined attack.

Collision Distance-based Doubling Attack The doubling attack is a special case of the SPA attack with a chosen-message assumption and is useful for thwarting the SPA-protected left-to-right square-and-multiply-always algorithm. This attack has been known to work only when the left-to-right routine exponentiation is used. However this paper demonstrates that the right-to-left version can be damaged by a variant of the doubling attack. Here call this attack a collision distance-based doubling attack.

The assumption about the input condition of this attack is the same as that for the original doubling attack. That is, the two attacks use related inputs m and m^2 as the chosen input message. However, analysis methods for retrieving the secret bit are different. The goal of the doubling attack is to find the state of collision pairs. On the other hand, one of this paper's proposed attacks is to check the distance of collision pairs. The example in Table 1 details the collision distance-based doubling attack. Let the secret key d be $89=(1,0,1,1,0,0,1)_2$. Here the sequence of operations can be compared when the right-to-left square-always algorithm in Figure 3 is used to compute m^d and $(m^2)^d$.

A collision between intermediate values occurs only during the computation of the sequential squaring of m, that is, m^2, m^4, m^8 and so on. More precisely, it is observed that the squaring operation at rank i in the computation of m^d is the same as the adjacent squaring operation at rank $(i-1)$ in the computation of $(m^2)^d$ if and only if $d_i=0$. As a result, if collision pairs are adjacent in the next state, then, according to this iterative operation, the secret bit d_i is 0. By contrast, if two nonadjacent collisions are away at a distance of about 3 states, then the secret bit d_i is 1. In Table 1, ①, ②, and ③ refer to adjacent collision pairs(nonadjacent pairs are not marked).

The original doubling attack against the left-to-right square-and-multiply-always algorithm focuses on deriving an exponent bit by determining whether collisions exist. The proposed attack is applied to the right-to-left exponentiation algorithm by checking the time distance of collision pairs. As a result, the attacker can retrieve the secret key d by observing the power collision distance of two exponentiations by using inputs m and m^2.

Chosen-Message SPA Attack Consider the security of the right-to-left square-always exponenti-

Table 2: Computations of chosen-message SPA attack.

i	d_i	Input : $(n-1)$				Remarks
		$j=0$ $(d_i=0)$	$j=1$ $(d_i=1)$	$j=2$ $(d_i=1)$	$j=3$ $(d_i=1)$	
0	1		$R_2=0^2$	$R_1=\left(\frac{n-2}{2}\right)^2=1\bigstar$	$R_1=(n-1)$ $R_0=(n-1)^2=1\bigstar$	High power
1	0	$R_0=1^2$				Low power
2	0	$R_0=1^2$				Low power
3	1		$R_2=0^2$	$R_1=\left(\frac{2-n}{2}\right)^2=1\bigstar$	$R_1=(n-1)$ $R_0=1^2$	High power
4	1		$R_2=0^2$	$R_1=\left(\frac{2-n}{2}\right)^2=1\bigstar$	$R_1=(n-1)$ $R_0=1^2$	High power
5	0	$R_0=1^2$				Low power
6	1		$R_2=0^2$	$R_1=\left(\frac{2-n}{2}\right)^2=1\bigstar$	$R_1=(n-1)$ $R_0=1^2$	High power
return		$R_1=(n-1)$				

ation algorithm for the case in which the chosen message $m=(n-1)$ is used as the input, that is, $R_0=(n-1)$. Here this chosen-message SPA attack is generally referred to as an $(n-1)$ attack. Our chosen-message SPA attack on the right-to-left square-always exponentiation algorithm can be applied only if the exponent is an odd number. That is, this attack is not useful for an even number. Table 2 details the $(n-1)$ attack on the algorithm in Figure 3. For example, the secret key d is considered to be $89=(1,1,0,1,0,0,1)_2$.

In the case of $d_i=0(j=0)$ and $d_i=1(j=3)$, the squaring result R_0 is always 1 because $(n-1)^{2^i}$ is 1, $i=1,2,3,....n-1$. In the case of $d_i=1(j=1)$, the squaring result is always 0 because $R_2=((R_1+R_0)/2)^2=((1+(n-1))/2)^2 \bmod n=0^2$ or $R_2=((R_1+R_0)/2)^2=(((n-1)+1)/2)^2 \bmod n=0^2$. In addition, if $d_i=1(j=2)$, then the squaring result is always 1 because $R_1=((n-2)/2)^2 \bmod n$ or $R_1=((2-n)/2)^2 \bmod n$. Therefore, it can be seen that the Hamming weight of the operational input $(n-2)/2$ or $(2-n)/2$ is high only if $d_i=1(j=2)$. In Table 2, the symbol $\bigstar$ refers to the distinct segment of squarings in which power consumption is high. As a result, an attacker can recover the secret key d by observing only one simple power trace measured after inputting a special input $(n-1)$.

CPA-based Combined Attack This paper shows that the right-to-left square-always algorithm is vulnerable to a new combined attack under the same fault injection model applied to the atomicity-based square-and-multiply algorithm in [4]. However, power signal analysis methods are different. Existing methods are based on the simple power analysis, that is, an SPA-based combined attack, whereas the proposed attack is based on finding the collision power of two segment operations. Therefore, this attack is called a CPA-based combined attack.

Indeed, during the computation of the right-to-left square-always algorithm, the attacker tries to skip computations in step 2 of algorithm 3 by a fault injection. If step 2 is bypassed because of a fault injection, then the value of R_1 is kept as the initial value in memory, typically $R_1=0$. By such a fault effect, the collision of two adjacent squarings can be found in step 10. As an example, the computational sequence of exponentiations after initializing R_1 with 0 by a fault injection is shown in Table 3.

The final values of squarings at $j=1$ are the same as those at $j=2$. Indeed, a collision between

Table 3: Combined attack based on fault and CPA attacks.

i	d_i	$R_1 = 0$ by fault injection				Remarks
		$j = 0$ ($d_i = 0$)	$j = 1$ ($d_i = 1$)	$j = 2$ ($d_i = 1$)	$j = 3$ ($d_i = 1$)	
0	1		$R_2 = \left(\frac{m}{2}\right)^2$ ①	$R_1 = \left(\frac{m}{2}\right)^2$ ①	$R_1 = 0$ $R_0 = m^2$	Adjacent collision
1	0	$R_0 = m^4$				No collision
2	0	$R_0 = m^8$				No collision
3	1		$R_2 = \left(\frac{m^8}{2}\right)^2$ ②	$R_1 = \left(\frac{m^8}{2}\right)^2$ ②	$R_1 = 0$ $R_0 = m^{16}$	Adjacent collision
4	1		$R_2 = \left(\frac{m^{16}}{2}\right)^2$ ③	$R_1 = \left(\frac{m^{16}}{2}\right)^2$ ③	$R_1 = 0$ $R_0 = m^{32}$	Adjacent collision
5	0	$R_0 = m^{64}$				No collision
6	1		$R_2 = \left(\frac{m^{64}}{2}\right)^2$ ④	$R_1 = \left(\frac{m^{64}}{2}\right)^2$ ④	$R_1 = 0$ $R_0 = m^{128}$	Adjacent collision
return		$R_1 = 0$				

two adjacent squarings occurs only if the secret bit d_i is 1. In Table 3, the marks ①, ②, ③, and ④ refer to adjacent collision pairs. If an attacker finds adjacent collisions by using only a single power trace measured after injecting a fault in step 2, then the attacker can easily know that the exponent bit d_i is 1. As a result, the right-to-left square-always algorithm reveals the secret key d for a combined attack with fault and CPA attacks. The weakness of the right-to-left square-always algorithm with respect to a new combined attack originates from the fact that one multiplication in the case of $d_i = 1$ composed of three squarings differs from the original squaring in the case of $d_i = 0$.

4 Improvements and Comparisons

To prevent collision-distance based doubling and chosen-message SPA attacks simultaneously, input message blinding can be a clever countermeasure. Because an attacker cannot use an intentional message as an input because of the application of the message blinding technique, the exponentiation algorithm can easily prevent such chosen-message type attacks. Some message blinding methods include multiplicative blinding, additive blinding, and modulus blinding. However the exponent blinding is not efficient countermeasure against the chosen-message SPA attack.

To defeat a CPA-based combined attack, the register value R_1, which is assumed to be 0 by a fault injection, should be automatically filled with an unknown random value. As a result, although the faulty register becomes 0 by an attacker, it should not leak any side information during the exponentiation operation.

	Alg. 4 : Improved Right-to-Left Square-Always Exponentiation
	Input: m, n, $d = (d_k - 1, d_k - 2, \cdots d_0)_2$
	Output: $m^d \bmod n$
1.	Generate random numbers, r_1 and r_2 such that $r_2 > r_1$
2.	$n_1 = r_1 \cdot n$; $n_2 = r_2 \cdot n$
3.	$R_0 = (m + n_1) \bmod n_2$
4.	$R_1 = (1 + n_1) \bmod n_2$
5.	$R_2 = (1 + n_1) \bmod n_2$
6.	$i = 0$; $j = 0$
7.	while $i \leq k - 1$ do
8.	$j = d_i(1 + (j \bmod 3))$
9.	$R_{M_{j,0}} = R_{M_{j,1}} + R_0 \bmod n_2$
10.	$R_{M_{j,2}} = R_{M_{j,3}}/2 \bmod n_2$
11.	$R_{M_{j,4}} = R_{M_{j,6}} - R_{M_{j,6}} \bmod n_2$
12.	$R_{M_{j,3}} = (R_{M_{j,3}} + n_1) \bmod n_2$
13.	$R_{M_{j,3}} = R_{M_{j,3}}^2 \bmod n_2$
14.	$i = i + M_{j,7}$
15.	return ($R_1 \bmod n$)

Figure 5: Improved right-to-left square-always exponentiation.

4.1 Improvements of the Square-Always Algorithm

Based on the aforementioned analysis of side-channel attacks and existing countermeasures, this paper presents an improved exponentiation algorithm (Figure 5). Indeed, the following countermeasures are added to the right-to-left square-always algorithm.

Additive Messages and Modulus Blinding A message can be randomized additively by the classical countermeasure $m' = m + r_1 \cdot n \bmod r_2 \cdot n$ with two random values r_1 and r_2 [5, 9]. Here r_i should use a 32-bit or larger random number to preserve the level of security. The initial operations from steps 1 to 5 in Figure 4 are applied to prevent the collision distance-based doubling and chosen-message SPA attacks. The core objective of this countermeasure is to prevent any attacker from controlling the input message. In addition, because an attacker cannot guess intermediate values computed during the execution of the algorithm because of the randomness of the modulus, the proposed algorithm is resistant to the DPA attack.

Input Message Update of Squaring If the register R_1 is kept at 0 by inserting a fault to bypass step 2 in Figure 3, then the Hamming weight of it is also 0. This condition affects the level of power consumption during squaring operations, and there is some information leakage. To prevent a CPA-based combined attack, the intermediate value should be automatically updated to another random one. Nevertheless, this update technique does not affect the final output in the case of a normal exponentiation execution.

As shown in step 12 of Figur 4, the input value of each squaring should be randomized, such as $R_{M_{j,3}} = (R_{M_{j,3}} + n_1) \bmod n_2$. Actually, only the randomization of R_1 in step 12 is enough to resist a CPA-based combined attack. Nevertheless, all register values used as the squaring input must be updated to maintain the regularity and efficiency of the algorithm. In addition, this message update technique before the main squaring operation can be applied to the left-to-right square-always algorithm to prevent a horizontal CPA attack.

Table 4: Security comparison of exponentiation algorithms

Algorithm & Attack	Left-to-Right			Right-to-Left		
	Multiply -always [4]	Square -always [6]	Blinded method [5]	Multiply -always [4]	Square -always [6]	Improved
SPA [13]	X	O	O	X	O	O
Doubling [10]	X	X	O	X	-	-
Collision distance-based doubling	-	-	-	-	X	O
Chosen-message SPA [20]	X	X	O	X	X	O
SPA-based combined [2]	X	X	X	X	-	-
CPA-based combined	-	-	-	-	X	O

Note : O : Secure, X : Weak

4.2 Comparisons of Exponentiations

Table 4 provides a security comparison based on existing countermeasures against several side-channel attacks. As mentioned earlier, atomicity-based multiply-always algorithms are proposed mainly to prevent SPA attacks [4]. However, these algorithms are no longer secure because an attacker can distinguish multiplications from squarings [1].

Therefore, square-always exponentiation algorithms [6] are presented to replace one multiplication with two squarings in multiply-always versions. As pointed out in previous research, the left-to-right square-always exponentiation algorithm can be insecure against doubling, chosen-message SPA, and SPA-based combined attacks. If the message blinding technique is applied to this algorithm, then it can defeat doubling and chosen-message SPA attacks. To resist a combined attack, some additive revision is required for the left-to-right square-always algorithm. One solution to this problem is the intermediate message update method, which is applied before all squaring operations of the improved right-to-left square-always algorithm (Figure 4).

This paper points out that the right-to-left square algorithm is vulnerable to three new types of side-channel power attacks: collision distance-based doubling attack, chosen-message SPA attack, and CPA-based combined attack. Two of these attacks are variants of the original doubling attack and the SPA-based combined attack. For high security, an additive message randomization method and a message update technique before the main squaring operation are adopted. The improved version of the right-to-left square-always algorithm is designed to thwart almost all side-channel attacks, including the DPA attack.

In terms of computational efficiency, the square-always algorithms require an average of $2S$ per exponent bit, which indicates an 11.1% theoretical speed-up over regular exponentiation methods such as the Montgomery ladder and square-and-multiply-always algorithms. The main computation in the improved algorithm (Figure 4) is the modular squaring in step 13. Nevertheless, the computational load for computing the additive message update in step 12 is negligible in comparison to the main operation.

The number of modular squaring operations in step 13 is the same as that for the original right-to-left square-always algorithm. However, squarings using the modulo number n_2 in the improved algorithm instead of n in the original right-to-left square-always algorithm can increase the level of complexity. The additive message blinding method based on extending the modulo number to n_2 is usually adopted to prevent a DPA attack. Because the modular squaring operation for two n-bit integers takes $O(n^2)$-bit operations, the improved algorithm using the extended modulus requires more time (about $((1024+32)/1024)^2 \approx 6.3\%$) than the original right-to-left square-always algorithm in the case of $|k| = 1024$.

5 Conclusions

Since the introduction of side-channel attacks, there has been an increase in the number of malicious attackers, including insiders with detailed knowledge of design information on the implementation of embedded devices. The square-always exponentiation algorithm is presented as a countermeasure against side-channel attacks aimed at distinguishing squarings from multiplications. This exponentiation algorithm is faster than existing countermeasures such as the Montgomery ladder and square-and-multiply-always algorithms.

This paper analyzes the vulnerability of the right-to-left square-always exponentiation algorithm with respect to power analysis attacks. According to the results, this algorithm is vulnerable to the collision distance-based doubling, the chosen-message SPA, and CPA-based combined attacks. In this regard, the paper presents an improved version of the right-to-left square-always exponentiation algorithm based on additive message blinding and an intermediate massage update technique. The improved exponentiation algorithm is well suited for the implementation of a secure RSA cryptosystem in low-resource devices.

Acknowledgments

This work was supported by the KLA-SCARF project, the ICT R&D program of ETRI (Research on Key Leakage Analysis and Response Technologies)

References

[1] F. Amiel, B. Feix, M. Tunstall, C. Whelen, and W. Marnane. Distinguishing Multiplications from Squaring Operations. In *Proc. of the 15th International Workshop on Selected Areas in Cryptography (SAC'08), Sackville, New Brunswick, Canada, LNCS*, volume 5381, pages 346–360. Springer-Verlag, August 2009.

[2] F. Amiel, K. Villegas, B. Feix, and L. Mercel. Passive and Active Combined Attacks: Combining fault attacks and side channel analysis. In *Proc. of the 4th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'07), Vienna, Austria*, pages 92–102. IEEE, September 2007.

[3] E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage method. In *Proc. of the 6th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'04), MA, USA, LNCS*, volume 3156, pages 16–29. Springer-Verlag, August 2004.

[4] B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. *IEEE Transactions on Computers*, 53(6):760–768, June 2004.

[5] C. Clavier and B. Feix. Updated recommendations for blinded exponentiation vs. Single trace analysis. In *Proc. of the 4th International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE'13), Paris, France, LNCS*, volume 7864, pages 80–98. Springer-Verlag, March 2013.

[6] C. Clavier, B. Feix, G. Gagnerot, and M. Roussellet. Square always exponentiation. In *Proc. of the 12th International Conference on Cryptology in India (INDOCRYPT'11), Chennai, India, LNCS*, volume 7107, pages 40–57. Springer-Verlag, December 2011.

[7] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. Horizontal correlation analysis on exponentiation. In *Proc. of the 12th International Conference on Information and Communications Security (ICICS'10), Barcelona, Spain, LNCS*, volume 6476, pages 46–61. Springer-Verlag, December 2010.

[8] J. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Proc. of the 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES'99), Worcester, MA, USA, LNCS*, volume 1717, pages 292–302. Springer-Verlag, August 1999.

[9] J. Courrege, B. Feix, and M. Rousellet. Simple power analysis on exponentiation revisited. In *Proc. of the 9th International Conference on Smart Card Research and Advanced Application(CARDIS'10), Passau, Germany, LNCS*, volume 6035, pages 65–79. Springer-Verlag, April 2010.

[10] P. Fouque and F. Valette. The doubling attack- why upwards is better than downwards. In *Proc. of the 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'03), Cologne, Germany, LNCS*, volume 2779, pages 269–280. Springer-Verlag, September 2003.

[11] M. Joye. Highly regular m-ary power ladders. In *Proc. of the 16th Annual International Workshop on Selected Areas in Cryptography (SAC'09), Calgary, Alberta, Canada, LNCS*, volume 5867, pages 350–363. Springer-Verlag, August 2009.

[12] M. Joye and S. Yen. The Montgomery powering ladder. In *Proc. of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'02), Redwood Shores, CA, USA, LNCS*, volume 2523, pages 291–302. Springer-Verlag, August 2002.

[13] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Proc. of the 19th International Cryptology Conference (CRYPTO'99), Santa Barbara, California, USA, LNCS*, volume 1666, pages 388–397. Springer-Verlag, August 1999.

[14] T. Messerges, E. Dabbish, and R. Sloan. Power analysis attacks on modular exponentiation in smart cards. In *Proc. of the 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES'99), Worcester, MA, USA, LNCS*, volume 1717, pages 144–157. Springer-Verlag, August 1999.

[15] R. Rivest, A. Shamir, and L. Adelman. A method for obtaining digital signature and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[16] C. Walter. Sliding window succumbs to Big Mac attack. In *Proc. of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES'01), Paris, France, LNCS*, volume 2162, pages 286–299. Springer-Verlag, May 2001.

[17] M. Witteman, J. van Woudenberg, and F. Menarini. Defeating RSA multiply-always and message blinding countermeasures. In *Proc. of the Cryptographers' Track at the RSA Conference (CT-RSA'11), San Francisco, CA, USA, LNCS*, volume 6558, pages 77–88. Springer-Verlag, February 2011.

[18] S. Yen, S. Kim, S. Lim, and S. Moon. A countermeasure against one physical cryptanalysis may benefit another attack. In *Proc. of the 4th International Conference on Information Security and Cryptology (ICISC'01), Seoul, Korea, LNCS*, volume 2288, pages 414–427. Springer-Verlag, December 2002.

[19] S. Yen, L. Ko, S. Moon, and J. Ha. Relative doubling attack against Montgomery ladder. In *Proc. of the 8th International Conference on Information Security and Cryptology (ICISC'05), Seoul, Korea, LNCS*, volume 3935, pages 117–128. Springer-Verlag, December 2005.

[20] S. Yen, W. Lien, S. Moon, and J. Ha. Power analysis by exploiting chosen message and internal collision-Vulnerability of checking mechanism for RSA decryption. In *Proc. of the 1st International Conference on Cryptology in Malaysia (Mycrypt'05), Kuala Lumpur, Malaysia, LNCS*, volume 3715, pages 183–195. Springer-Verlag, September 2005.

Author Biography

Jaecheol Ha received the BE, ME, and PhD in electronics engineering from Kyungpook National University, Rep. of Korea, in 1989, 1993, and 1998, respectively. He is currently a full professor of the department of information and security at Hoseo University, Asan, Rep. of Korea. During 1998 to 2006, he also worked as a professor in the department of information and communication at Korea Nazarene University, Cheonan, Korea. In 2006, he was a visiting researcher at the Information Security Institute of Queensland University of Technology, Australia. His research interests include network security, smart card security, crypto chip design, and side-channel attacks.

Yongje Choi received his BSEE and MS from Cheonnam National University, Kwangju, Rep. of Korea in 1996 and 1999, respectively. He is a senior member of technical staff at ETRI, Deajeon, Rep. of Korea. His research interests include side channel VLSI design, crypto processor design, side channel analysis, and information security.

Dooho Choi received the BS in mathematics from Sungkyunkwan University, Seoul, Korea, in 1994, and the MS and PhD in mathematics from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1996, 2002, respectively. He has been a principal researcher at ETRI, Daejeon, Korea, since January 2002. His current research interests are side-channel analysis and its resistant crypto design, security technologies of RFID, and lightweight cryptographic protocol/module design for IoT.

Hoonjae Lee received the B.S., M.S. and Ph.D. degree in Electrical Engineering from Kyungpook national university in 1985, 1987 and 1998, respectively. He had been engaged in the research on cryptography and network security at Agency for Defense Development from 1987 to 1998. Since 2002 he has been working for Department of Computer Engineering of Dongseo University as an associate professor, and now he is a full professor. His current research interests are in security communication system, side-channel attack, USN & RFID security. He is a member of the Korea institute of information security and cryptology, IEEE Computer Society, IEEE Information Theory Society and etc