Relative Doubling Attack Against Montgomery Ladder*

Sung-Ming Yen[1], Lee-Chun Ko[1], SangJae Moon[2], and JaeCheol Ha[3]

[1] Laboratory of Cryptography and Information Security (LCIS)
Dept of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan 320, R.O.C.
E-mail: {yensm;cs222085}@csie.ncu.edu.tw
http://www.csie.ncu.edu.tw/~yensm/

[2] School of Electronic and Electrical Engineering
Kyungpook National University, Taegu, Korea 702-701
E-mail: sjmoon@ee.knu.ac.kr

[3] Dept of Computer and Information
Korea Nazarene University, Choong Nam, Korea 330-718
E-mail: jcha@kornu.ac.kr

Abstract. Highly regular execution and the cleverly included redundant computation make the square-multiply-always exponentiation algorithm well known as a good countermeasure against the conventional simple power analysis (SPA). However, the doubling attack threatens the square-multiply-always exponentiation by fully exploiting the existence of such redundant computation. The Montgomery ladder is also recognized as a good countermeasure against the conventional SPA due to its highly regular execution. Most importantly, no redundant computation is introduced into the Montgomery ladder. In this paper, immunity of the Montgomery ladder against the doubling attack is investigated. One straightforward result is that the Montgomery ladder can be free from the original doubling attack. However, a non-trivial result obtained in this research is that a relative doubling attack proposed in this paper threatens the Montgomery ladder. The proposed relative doubling attack uses a totally different approach to derive the private key in which the relationship between two adjacent private key bits can be obtained as either $d_i = d_{i-1}$ or $d_i \neq d_{i-1}$. Finally, a remark is given to the problem of whether the upward (right-to-left) regular exponentiation algorithm is necessary against the original doubling attack and the proposed relative doubling attack.

Keywords: Chosen-message attack, Cryptography, Doubling attack, Exponentiation, Scalar multiplication, Side-channel attack, Simple power analysis (SPA), Smart card.

* This work was supported by University IT Research Center Project.

1 Introduction

Cryptographic hardware devices like smart cards are widely used nowadays. During the past few years many research results have been published on considering smart card side-channel attacks because of the popular usage of smart cards on implementing cryptosystems. This new branch of cryptanalysis is usually called the side-channel attack. The power analysis attack is an important category of side-channel attack originally pointed out by Kocher [1] in which both simple power analysis (SPA) and differential power analysis (DPA) were considered.

In a SPA, the attacker observes on one or a few collected power traces of the smart card executing an algorithm and tries to identify the occurrence of an instruction execution or a specific operand/data access which are driven by a part of the private key. Through the above observation, if precise enough, the private key can be derived. In a DPA, the attacker tries to verify his guess on a part of the private key by analyzing on only some specific bits of the result of a specific intermediate step of an algorithm which is a function of the private key. In order to largely enhance the signal to noise ratio to mount a successful DPA, it usually collects much more[1] power traces than in a SPA and partitions the power traces into some groups according to the guessed key bits and a underlying attack design. Difference of the above power traces of different groups is therefore used to verify the guess on key bits. Usually, a DPA is mounted by analyzing on many executions of the same algorithm with different random inputs, and theoretically those inputs will be better if statistically unrelated.

Exponentiation and its analogy, point scalar multiplication on elliptic curve, are of central importance in modern cryptosystems implementation as they are of the basic operation of almost all modern public-key cryptosystems, e.g., the RSA system [2], the ElGamal system [3], and the elliptic curve cryptography [4, 5]. Therefore, many side-channel attacks and also the related countermeasures on implementing exponentiation and point scalar multiplication have been reported in the literature.

The square-multiply-always exponentiation (or point scalar multiplication) algorithm [6] is a well-known SPA countermeasure which exploits a simple and useful trick to design a regularly executing algorithm by introducing redundant computation into each loop iteration when necessary. Unfortunately, Fouque and Valette proposed the doubling attack [7] to threaten the square-multiply-always algorithm (more precisely, the left-to-right version of the algorithm) by exploiting the existence of redundant computation in a novel approach.

Joye and Yen proposed an enhanced SPA countermeasure based on the Montgomery ladder [8] which was demonstrated to be also regularly executed but based on a totally different idea from the original square-multiply-always exponentiation. The most special thing about the Montgomery ladder is that no redundant computation exists in the algorithm which is also helpful to be immune from some hardware fault attacks [9, 10].

[1] It usually collects a few thousands or more power traces in order to obtain a meaningful average power trace.

However, no research has been reported on whether the Montgomery ladder can be immune from the doubling attack or any doubling-like attack in the light of the fact of no redundant computation within the algorithm. The main contribution of this paper is that a totally different approach of doubling attack (called the relative doubling attack) is proposed which can successfully threaten the Montgomery ladder with the same attack assumption as the original doubling attack. The lesson learned is that redundant computation removing in a regular exponentiation algorithm may not be sufficient against doubling-like attack.

2 Exponentiation Algorithm and Simple Power Analysis

2.1 Exponentiation Algorithm

In this paper, we consider the problem of computing modular exponentiation. In the context of RSA private computation (e.g., decryption), we consider the computation of $S = M^d \bmod n$ where M, d, and n are the input datum, the private key, and the modulus integer, respectively.

Let $\sum_{i=0}^{m-1} d_i\, 2^i$ be the binary expansion of exponent d. The computation $S = M^d \bmod n$ needs efficient exponentiation algorithms to speedup its implementation. Although numerous exponentiation algorithms have been developed for computing $M^d \bmod n$ (see [11] for a survey), practical solutions for devices with constrained computation and storage capabilities (e.g., smart cards) are usually restricted to the basic square-multiply algorithms in Fig. 1 and some slightly modified ones. The left-to-right (MSB-to-LSB) version in Fig. 1 (a) is especially preferable to implementations in smart cards because this algorithm needs only one temporary memory R_0 if the input data M is also stored inside the smart cards.

	Input: $M, d = (d_{m-1} \cdots d_0)_2, n$ Output: $M^d \bmod n$
01	$R_0 \leftarrow 1$
02	`for` $i = m-1$ `downto` 0 `do`
03	$R_0 \leftarrow (R_0)^2 \bmod n$
04	`if` $(d_i = 1)$ `then`
	$R_0 \leftarrow R_0 \times M \bmod n$
05	`return` R_0

(a) Left-to-right binary algorithm.

	Input: $M, d = (d_{m-1} \cdots d_0)_2, n$ Output: $M^d \bmod n$
01	$R_0 \leftarrow 1$; $R_1 \leftarrow M$
02	`for` $i = 0$ `to` $m-1$ `do`
03	`if` $(d_i = 1)$ `then`
	$R_0 \leftarrow R_0 \times R_1 \bmod n$
04	$R_1 \leftarrow (R_1)^2 \bmod n$
05	`return` R_0

(b) Right-to-left binary algorithm.

Fig. 1. Classical binary exponentiation algorithms.

2.2 Simple Power Analysis and Countermeasures

Side-channel attacks are developed based on the fact that in most real implementations some side-channel information (e.g., timing or power consumption) will

depend on the private key related instructions being executed and/or the data being manipulated. Therefore, the side-channel information may be exploited to mount a successful attack to retrieve the embedded private key, e.g., the private exponent d in $M^d \bmod n$.

The classical binary exponentiation algorithm in Fig. 1 (a) includes a conditional branch (i.e., the Step (04)) that is driven by the secret data d_i. If the two possible branches behave differently (or the branch decision operation itself behaves distinguishably), then some side-channel analysis (e.g., the simple power analysis–SPA) may be employed to retrieve the secret data d_i. So, further enhancement on the algorithms is necessary.

A novel idea of introducing redundant operations and eliminating secret data dependent statements was proposed previously to enhance the basic algorithms such that the improved versions behave more regularly. Some square-multiply-always (or its counterpart called the double-add-always for point scalar multiplication) based algorithms were already developed [6] by employing this observation. Two of these square-multiply-always algorithms are shown in Fig. 2.

	Input: $M, d = (d_{m-1} \cdots d_0)_2, n$ Output: $M^d \bmod n$
01	$R_0 \leftarrow 1$
02	for $i = m-1$ downto 0 do
03	$b \leftarrow \neg d_i$
04	$R_0 \leftarrow (R_0)^2 \bmod n$
05	$R_b \leftarrow R_0 \times M \bmod n$
06	return R_0

(a) SPA-protected left-to-right algorithm.

	Input: $M, d = (d_{m-1} \cdots d_0)_2, n$ Output: $M^d \bmod n$
01	$R_0 \leftarrow 1$; $R_2 \leftarrow M$
02	for $i = 0$ to $m-1$ do
03	$b \leftarrow \neg d_i$
04	$R_b \leftarrow R_0 \times R_2 \bmod n$
05	$R_2 \leftarrow (R_2)^2 \bmod n$
06	return R_0

(b) SPA-protected right-to-left algorithm.

Fig. 2. SPA-protected square-multiply-always countermeasures.

2.3 Doubling Attack

The doubling attack[2] [7] is a special category of SPA with chosen message assumption and it has been shown to be useful to thwart the well-known SPA-protected left-to-right (downward) square-multiply-always countermeasure (see Fig. 2 (a)). The main idea is to choose two strongly related inputs M and $M^2 \bmod n$ (so being a chosen-message attack) and to observe the collision of two computations for $M^{2(2x+d_i)} \bmod n$ and $M^{4x} \bmod n$ if $d_i = 0$. In the doubling attack, even if the attacker cannot decide whether a computation being performed is squaring or multiplication, the attacker can still detect collision of two operations (basically the squaring operation) within two related computations. More precisely, for two computations $A^2 \bmod n$ and $B^2 \bmod n$, even if the

[2] It can also be called the squaring attack for the scenario of exponentiation.

attacker cannot tell the values of A and/or B, however the attacker can detect the collision if $A = B$.

The following example given in Table 1 provides the details of the doubling attack. Let the private exponent d be $75 = (1, 0, 0, 1, 0, 1, 1)_2$ and the two related input data be M and M^2, respectively. The computational process of raising M^d and $(M^2)^d$ using the left-to-right square-multiply-always algorithm reveals the fact that if $d_i = 0$, then both the first computations (both are squarings) of iteration[3] $i - 1$ for M^d and iteration i for $(M^2)^d$ will be exactly the same. So, observing collisions (observation on the existence of same instruction with same operand) within computations of two collected power consumption traces enables the attacker to identify all private key bits of zero value except the LSB of d. In the scenario of RSA private computation, it is assumed that $d_0 = 1$.

The assumption made (was claimed in [7] to be correct by experiment) is very reasonable since the target computations usually take many machine clock cycles (thus more easy to measure and to observe) and depend greatly on the operands, so the collision is more easy to detect.

Table 1. Computations of M^d and $(M^2)^d$ in the square-multiply-always algorithm.

i	d_i	Process of M^d	Process of $(M^2)^d$
6	1	1^2 $1 \times M$	1^2 $1 \times M^2$
5	0	M^2 $M^2 \times M$	$(M^2)^2$ $M^4 \times M^2$
4	0	$(M^2)^2$ $M^4 \times M$	$(M^4)^2$ $M^8 \times M^2$
3	1	$(M^4)^2$ $M^8 \times M$	$(M^8)^2$ $M^{16} \times M^2$
2	0	$(M^9)^2$ $M^{18} \times M$	$(M^{18})^2$ $M^{36} \times M^2$
1	1	$(M^{18})^2$ $M^{36} \times M$	$(M^{36})^2$ $M^{72} \times M^2$
0	1	$(M^{37})^2$ $M^{74} \times M$	$(M^{74})^2$ $M^{148} \times M^2$
Return		M^{75}	M^{150}

2.4 Montgomery Ladder as Enhanced Countermeasure Against SPA

Montgomery ladder is originally due to Peter Montgomery [12] as a means to speed up scalar multiplication in the context of elliptic curves. It has been rediscovered in [13] in another context and applied to Lucas sequences.

In [8], an exponentiation algorithm based on Montgomery ladder was considered such that it can resist some side-channel attacks, e.g., SPA and timing

[3] Here, the iteration number is denoted decreasingly from $m - 1$ downward towards zero.

attack, and also the safe-error attacks [9, 10] (a category of hardware fault attack). The algorithm is given in Fig. 3. This algorithm is only SPA resistant and is used to simplify the description of the proposed attack. However, an enhanced version in [8] meant to be immune from the safe-error attacks with Step 04 replaced by ($R_b \leftarrow R_b \times R_{d_i} \bmod n$) is still vulnerable to the relative doubling attack proposed in this paper.

It is evident that the Montgomery ladder (and its enhanced version) behave regularly and most specially that there is no redundant computation within the algorithm.

	Input: $M, d = (d_{m-1} \cdots d_0)_2, n$ Output: $M^d \bmod n$
01	$R_0 \leftarrow 1$; $R_1 \leftarrow M$
02	for $i = m-1$ downto 0 do
03	$b \leftarrow \neg d_i$
04	$R_b \leftarrow R_0 \times R_1 \bmod n$
05	$R_{d_i} \leftarrow (R_{d_i})^2 \bmod n$
06	return R_0

Fig. 3. SPA-protected Montgomery ladder.

3 The Proposed Attack

3.1 Attack Assumption

The assumption made in this paper is basically the same as what considered in the doubling attack [7] and that in an attack reported in [14]. The assumption is that an adversary can distinguish collision of power trace segments (within a single or more power traces) when the smart card performs twice the same computation even if the adversary is not able to tell which exact computation is done. The collision instance to be distinguished in [7] and in our proposed attack is the modular squaring computation. An adversary is assumed to be able to detect the collision of $A^2 \bmod n$ and $B^2 \bmod n$ if $A = B$ even though A and B are unknown.

3.2 Relative Doubling Attack on Montgomery Ladder

Let $\sum_{j=0}^{m-1} d_j\, 2^j$ be the binary expansion of the exponent d. The Montgomery ladder (see Fig. 3) was designed based on the following observation [8]. Let $L_i = \sum_{j=i}^{m-1} d_j\, 2^{j-i}$ and $H_i = L_i + 1$, then we have

$$L_i = 2L_{i+1} + d_i = L_{i+1} + H_{i+1} - 1 + d_i,$$

$$H_i = L_{i+1} + H_{i+1} + d_i = 2H_{i+1} - 1 + d_i.$$

Based on the above observation, we obtain

$$(L_i, H_i) = \begin{cases} (2L_{i+1}, L_{i+1} + H_{i+1}) & \text{if } d_i = 0\,, \\ (L_{i+1} + H_{i+1}, 2H_{i+1}) & \text{if } d_i = 1\,. \end{cases} \quad (1)$$

In the algorithm (Fig. 3), the register R_0 is used to store the value of M^{L_i} and the register R_1 is used to store M^{H_i}. In order to develop an execution regular and SPA immune algorithm, the operations of Step 04 and Step 05 are designed to be as follows

$$(R_1 = M^{H_i}, R_0 = M^{L_i}) = \left(M^{L_{i+1}} \times M^{H_{i+1}}, (M^{L_{i+1}})^2\right) \quad \text{if } d_i = 0, \quad (2)$$

and

$$(R_0 = M^{L_i}, R_1 = M^{H_i}) = \left(M^{L_{i+1}} \times M^{H_{i+1}}, (M^{H_{i+1}})^2\right) \quad \text{if } d_i = 1. \quad (3)$$

The above statements clearly demonstrate that the Montgomery ladder executes highly regular and there is no redundant computation within the algorithm. Whatever the processed bit d_i, there is always a multiplication followed by a squaring. On the contrary, we want to emphasize that in the left-to-right square-multiply-always algorithm (see Fig. 2 (a)), redundant computation (i.e., Step 05: $R_b \leftarrow R_0 \times M \bmod n$) does exist when $d_i = 0$. The original doubling attack on the algorithm in Fig. 2 (a) exploits the existence of this redundant computation.

However, no research has been reported on whether the Montgomery ladder can be immune from the doubling attack or any doubling-like attack in the light of the fact of no redundant computation within the algorithm. A straightforward result can be obtained easily is that the original doubling attack does not apply to the Montgomery ladder. However, the following result will show that another doubling-like attack can still be applicable to the Montgomery ladder.

Fact 1 *Given $d_i = 0$, then we have $L_i = 2L_{i+1}$.*

Proof. This can be obtained directly from the definition of $L_i = \sum_{j=i}^{m-1} d_j\, 2^{j-i}$ since $d_i = 0$. □

Fact 2 *Given $d_i = 1$, then we have $H_i = 2H_{i+1}$.*

Proof. From the definitions of $L_i = \sum_{j=i}^{m-1} d_j\, 2^{j-i}$, $H_i = L_i + 1$, and also $d_i = 1$, we have $H_i = L_i + 1 = (2L_{i+1} + 1) + 1 = 2(L_{i+1} + 1) = 2H_{i+1}$. □

From Eq.(2), we understand that if $d_i = d_{i-1} = 0$ then both

$$\begin{cases} R_0 \leftarrow (M^{L_i})^2 : \text{ Step 05 of iteration } i-1 \text{ when evaluating } M^d \\ R_0 \leftarrow ((M^2)^{L_{i+1}})^2 : \text{ Step 05 of iteration } i \text{ when evaluating } (M^2)^d, \end{cases} \quad (4)$$

will perform the same computation because of $L_i = 2L_{i+1}$ (see Fact 1). Due to this observation of collision on computation, a new doubling-like attack can be mounted to derive the knowledge of $d_i = d_{i-1} = 0$.

On the other hand, from Eq.(3), we also observe that if $d_i = d_{i-1} = 1$ then both

$$\begin{cases} R_1 \leftarrow (M^{H_i})^2 : \text{ Step } \texttt{05} \text{ of iteration } i-1 \text{ when evaluating } M^d \\ R_1 \leftarrow ((M^2)^{H_{i+1}})^2 : \text{ Step } \texttt{05} \text{ of iteration } i \text{ when evaluating } (M^2)^d, \end{cases} \tag{5}$$

will perform the same computation because of $H_i = 2H_{i+1}$ (see Fact 2). This observation of collision on computation leads to the knowledge of $d_i = d_{i-1} = 1$.

All other cases, say $d_i \neq d_{i-1}$, will lead to either one of the following results

case (1): $d_i = 0$ **and** $d_{i-1} = 1$

$$\begin{cases} R_1 \leftarrow (M^{H_i})^2 : \text{ Step } \texttt{05} \text{ of iteration } i-1 \text{ when evaluating } M^d \\ R_0 \leftarrow ((M^2)^{L_{i+1}})^2 : \text{ Step } \texttt{05} \text{ of iteration } i \text{ when evaluating } (M^2)^d, \end{cases} \tag{6}$$

case (2): $d_i = 1$ **and** $d_{i-1} = 0$

$$\begin{cases} R_0 \leftarrow (M^{L_i})^2 : \text{ Step } \texttt{05} \text{ of iteration } i-1 \text{ when evaluating } M^d \\ R_1 \leftarrow ((M^2)^{H_{i+1}})^2 : \text{ Step } \texttt{05} \text{ of iteration } i \text{ when evaluating } (M^2)^d. \end{cases} \tag{7}$$

Based on the definition of Montgomery ladder, it is evident that in the case (1) we have $H_i \neq 2L_{i+1}$ and no collision of computation can be detected. Similarly, in the case (2) no collision of computation can be detected since $L_i \neq 2H_{i+1}$.

The Relative Doubling Attack. Recall that collision of two computations will not reveal the value of the operand. So, in the proposed attack, a collision of computations detected by the property of Eq.(4) and another collision of computations detected by the property of Eq.(5) cannot be distinguished. The only knowledge obtained is that $d_i = d_{i-1}$ if a collision is detected. On the other hand, the properties in Eq.(6) and Eq.(7) tell us that $d_i \neq d_{i-1}$ if no collision is detected. Due to its special property of the derived knowledge, the proposed attack is called the relative doubling attack to manifest the difference to the original doubling attack [7].

Based on the derived relationship between every two adjacent private key bits (either $d_i = d_{i-1}$ or $d_i \neq d_{i-1}$) and a given bit (e.g., d_0 or d_{m-1}), all other private key bits can be derived uniquely. For example, in RSA private computation it is assumed that $d_0 = 1$ under the same assumption made in the original doubling attack mentioned previously. Most importantly, we observed that it is sufficient to derive all the private key bits when given any d_i $(0 \leq i \leq m-1)$.

An Example of Attack. Following the same example of assuming the private exponent d to be $75 = (1,0,0,1,0,1,1)_2$ and the two related input data to be M and M^2 respectively, Table 2 provides the details of the proposed relative doubling attack on Montgomery ladder. The computational process of raising M^d and $(M^2)^d$ reveals the fact that given[4] $d_0 = 1$ and the observation of collision

[4] The RSA private exponent d is an odd integer. The original doubling attack also exploits this knowledge in order to obtain d_0.

on Step 05 of the iteration 1 of $(M^2)^d$ and Step 05 of the iteration 0 of M^d will lead to the result of $d_1 = d_0 = 1$. Given $d_0 = 1$, if no collision is detected, then $d_1 = 0$ since in this case d_1 should be different from d_0.

Table 2. Computations of M^d and $(M^2)^d$ in the Montgomery ladder.

i	d_i	Process of M^d	Process of $(M^2)^d$
6	1	$R_0 = 1 \times M$ $R_1 = M^2$	$R_0 = 1 \times M^2$ $R_1 = (M^2)^2$
5	0	$R_1 = M^2 \times M$ $R_0 = M^2$	$R_1 = M^4 \times M^2$ $R_0 = (M^2)^2$
4	0	$R_1 = M^3 \times M^2$ $R_0 = (M^2)^2$	$R_1 = M^6 \times M^4$ $R_0 = (M^4)^2$
3	1	$R_0 = M^4 \times M^5$ $R_1 = (M^5)^2$	$R_0 = M^8 \times M^{10}$ $R_1 = (M^{10})^2$
2	0	$R_1 = M^{10} \times M^9$ $R_0 = (M^9)^2$	$R_1 = M^{20} \times M^{18}$ $R_0 = (M^{18})^2$
1	1	$R_0 = M^{18} \times M^{19}$ $R_1 = (M^{19})^2$	$R_0 = M^{36} \times M^{38}$ $R_1 = (M^{38})^2$
0	1	$R_0 = M^{37} \times M^{38}$ $R_1 = (M^{38})^2$	$R_0 = M^{74} \times M^{76}$ $R_1 = (M^{76})^2$
Return		$R_0 = M^{75}$	$R_0 = M^{150}$

3.3 Comparison of Doubling Attack and Relative Doubling Attack

The original doubling attack (against square-multiply-always algorithm) focuses on deriving the private key bit d_i by checking whether $d_i = 0$. So, the original doubling attack tries to obtain the knowledge of *absolute* value of each d_i. On the contrary, the proposed relative doubling attack (against Montgomery ladder) focuses on deriving the knowledge of whether $d_i = d_{i-1}$ (relationship between every two adjacent key bits), but not the knowledge of either d_i or d_{i-1} directly. Nonetheless, given the value of either d_i or d_{i-1} will provide the exact value of the other one indirectly.

Furthermore, the original doubling attack fully exploits the existence of redundant computation. But the proposed relative doubling attack on the Montgomery ladder does not exploit the existence of any redundant computation. Evidently, in this paper, we showed a totally different approach of deriving the private key. The primary similarity of these two attacks is that both of them use (M, M^2) as the chosen input data.

3.4 Applicability of The Proposed Attack

Most published research results on side-channel attack considered the potential vulnerability on the computational algorithm (e.g., modular exponentiation) but not on a real cryptosystem and under a specific cryptographic standard (e.g., some padding or message format). This is basically reasonable since the computational algorithm itself is generic and can be employed as implementation

to many different cryptosystems (or some cryptosystems to be designed in the future) each may have different padding or message format. So, to point out potential attacks to the computational algorithm is still important.

Nonetheless, we still wish to point out clearly that the proposed relative doubling attack is applicable at least to the following systems if they are implemented based on the Montgomery ladder or its enhanced version in [8].

(1) Traditional textbook RSA decryption and signature.
(2) The RSA-OAEP decryption [15, 16]. It should be noted that the proposed attack does work on RSA-OAEP decryption since the ciphertext validity checking is performed after the RSA private exponentiation computation. So, the attacker still can collect the necessary power traces.
(3) ElGamal decryption [3].

4 Possible Enhancement Against Relative Doubling Attack

4.1 Remarks on Random Blinding Technique

One may argue that the standard blinding technique can easily prevent the proposed relative doubling as well as the original doubling attacks. However, we have some remarks on this claim.

The first disagreement is that the standard blinding technique is well known as a countermeasure against DPA. The second disagreement is that in a standard blinding technique the input data should be protected by a random mask which will then be removed from the result. However, it has been pointed out clearly in [7] that a regular mask updating (meant to be efficient), e.g., the one mentioned in [6], will be vulnerable to the doubling attack. It can be verified easily that the regular mask updating in [6] is also vulnerable to the proposed relative doubling attack. It was suggested eventually that it had better use a real random mask to avoid the attack. Unfortunately, the computational overhead of employing a real random mask is usually very high.

4.2 Is Upward Exponentiation Necessary Against Doubling Attack

The work and especially the title of [7] imply that upward (right-to-left) exponentiation could be better than downward (left-to-right) exponentiation when considering vulnerability from the doubling attack. This is also the case for the proposed relative doubling attack. However, the above mentioned superiority of the upward exponentiation is not obtained without any additional cost. It is evident that the upward square-multiply-always exponentiation in Fig. 2 (b) needs one more temporary memory than the downward exponentiation does.

Purpose of the following discussion is to clarify that upward exponentiation is not a necessary requirement meant to be immune from the doubling attack and the proposed relative doubling attack. The following SPA-protected and

safe-error-protected exponentiation algorithm [17] in Fig. 4 is a downward exponentiation algorithm. A limitation of this algorithm is that $d_{m-1} = 1$ is assumed. It can be verified easily that this algorithm is secure against the doubling and the relative doubling attacks.

	`Input:` $M, d = (d_{m-1}, d_{m-2} \cdots d_0)_2, d_{m-1}{=}1, n$
	`Output:` $M^d \bmod n$
01	$R_0 \leftarrow 1$; $R_1 \leftarrow M$; $d_{-1} \leftarrow 1$
02	`for` $i = m-1$ `downto` 0 `do`
03	$b \leftarrow \neg d_i$; $c \leftarrow d_{i-1}$
04	$R_0 \leftarrow R_0 \times R_b \bmod n$
05	$R_0 \leftarrow R_0 \times R_c \bmod n$
06	`return` R_0

Fig. 4. SPA-protected and safe-error-protected downward exponentiation.

Notice that the algorithm (Fig. 4) needs only two temporary memory (same as that in Fig. 2 (a)) and this leads to *one* less temporary memory requirement than the doubling attack immune upward algorithm in Fig. 2 (b). Recall that if we take into account the fact that the input datum M is also stored inside the smart card (as already described previously), then the algorithm in Fig. 4 needs only one temporary memory which leads to *two* less temporary memory requirement than the doubling attack immune upward algorithm in Fig. 2 (b). However, it is worth noting that protection against relative doubling attack does not necessarily ward off other potential attacks.

5 Conclusions

The Montgomery ladder can be secure against both the ordinary SPA and the ordinary doubling attack. But, in this paper we showed that the Montgomery ladder is vulnerable to the proposed relative doubling attack. Both the ordinary doubling attack and the proposed relative doubling attack share the same reasonable attack assumption of observing collision on computations. One difference is that the original doubling attack (against square-multiply-always algorithm) fully exploits the existence of redundant computation, while the proposed relative doubling attack (against Montgomery ladder) does not exploit any redundant computation. Our relative doubling attack uses a different approach to derive the private key.

6 Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful suggestions and comments on both technical and editing issues. These suggestions improve extensively to the final version of this paper.

References

1. P. Kocher, J. Jaffe and B. Jun, "Differential power analysis," *Advances in Cryptology – CRYPTO '99*, LNCS 1666, pp. 388–397, Springer-Verlag, 1999.
2. R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystem," *Commun. of ACM*, vol. 21, no. 2, pp. 120–126, 1978.
3. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985.
4. V. Miller, "Uses of elliptic curve in cryptography," *Advances in Cryptology – CRYPTO '85*, LNCS 218, pp. 417–426, Springer-Verlag, 1985.
5. N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, Jan. 1987.
6. J.-S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '99*, LNCS 1717, pp. 292–302, Springer-Verlag, 1999.
7. P.-A. Fouque and F. Valette, "The doubling attack – why upwards is better than downwards," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '03*, LNCS 2779, pp. 269–280, Springer-Verlag, 2003.
8. M. Joye and S.M. Yen., "The Montgomery powering ladder," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '02*, LNCS 2523, pp. 291–302, Springer-Verlag, 2003.
9. S. M. Yen and M. Joye, "Checking Before Output May Not be Enough against Fault-Based Cryptanalysis," *IEEE Trans. on Computers*, 49(9):967-970, September 2000.
10. S.M. Yen, S.J. Kim, S.G. Lim and S.J. Moon, "A countermeasure against one physical cryptanalysis may benefit another attack," *Proc. of Information Security and Cryptology – ICISC '01*, LNCS 2288, pp. 414–427, Springer-Verlag, 2002.
11. D.M. Gordon, "A survey of fast exponentiation methods," *Journal of Algorithms*, vol. 27, pp. 129–146, 1998.
12. P.L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Mathematics of Computation*, vol. 48, no. 177, pp. 243–264, Jan. 1987.
13. S.M. Yen and C.S. Laih, "Fast algorithms for LUC digital signature computation," *IEE Proc. Computers and Digital Techniques*, vol. 142, no. 2, pp. 165–169, March 1995.
14. K. Schramm, T. Wollinger, and C. Paar, "A new class of collision attacks and its application to DES," *Proc. of Fast Software Encryption – FSE '03*, LNCS 2887, pp. 206–222, Springer-Verlag, 2003.
15. PKCS #1 v2.1, "RSA Cryptography Standard", 5 January 2001. `http://www.rsasecurity.com/rsalabs/pkcs/`
16. M. Bellare and P. Rogaway, "Optimal asymmetric encryption padding – How to encrypt with RSA," *Advances in Cryptology – EUROCRYPT '94*, LNCS 950, pp. 92–111, Springer-Verlag, 1995.
17. S.M. Yen, C.C. Lu, and S.Y. Tseng, "Method for protecting public key schemes from timing, power and fault attacks," U.S. Patent Number US2004/0125950 A1, July 2004.