Power Analysis by Exploiting Chosen Message and Internal Collisions – Vulnerability of Checking Mechanism for RSA-Decryption[⋆]

Sung-Ming Yen[1], Wei-Chih Lien[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.
{yensm, cs222058}@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
sjmoon@ee.knu.ac.kr

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

Abstract. In this paper, we will point out a new side-channel vulnerability of cryptosystems implementation based on BRIP or square-multiply-always algorithm by exploiting specially chosen input message of order two. A recently published countermeasure, BRIP, against conventional simple power analysis (SPA) and differential power analysis (DPA) will be shown to be vulnerable to the proposed SPA in this paper. Another well known SPA countermeasure, the square-multiply-always algorithm, will also be shown to be vulnerable to this new attack. Further extension of the proposed attack is possible to develop more powerful attacks.

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

1 Introduction

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* (SCA).

The power analysis attack is an important category of SCA originally published by Kocher [1] in which both simple power analysis (SPA) and differential power analysis (DPA) were considered. SPA tries to extract the private key by

[⋆] This work was supported by University IT Research Center Project.

© Springer-Verlag Berlin Heidelberg 2005

observing on a single or a very few number of power consumption traces collected from the smart card. DPA consists in performing a statistical analysis of many power consumption traces (say a few thousands or more) of the same algorithm with different inputs.

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] and the elliptic curve cryptography [3,4]. Therefore, many side-channel attacks and also the related countermeasures on implementing exponentiation and point scalar multiplication have been reported in the literature.

Some recent works of power analysis attack, e.g., refined power analysis (RPA) [5], zero-value point attack (ZPA) [6], and doubling attack [7], threaten most existing countermeasures for implementing exponentiation and point scalar multiplication, e.g., some countermeasures in [8]. Recently, Mamiya *et al* proposed an enhanced countermeasure which was claimed to resist against RPA, ZPA, classical DPA and SPA, and also doubling attack by introducing a new random blinding technique and also exploiting a well known regular program execution trick (say the square-multiply-always like approach) for each loop iteration.

The main contribution of this paper is that a new SPA by exploiting specific chosen message is proposed in which collecting a single power trace is sufficient to mount a successful attack. An important result obtained is that both the well known SPA resistant countermeasure by using the square-multiply-always algorithm [8] and also the recent and enhanced BRIP algorithm [9] are shown to be vulnerable to this new attack. Further extension on the attack is also pointed out by selecting more general and random input messages which makes the detection of a specific message employed in the basic attack be infeasible and this leads to a more powerful extended attack. Furthermore, the proposed attack is also applicable to implementation of RSA with CRT speedup. Another important observation is that cryptographic padding (e.g., RSA-OAEP [10,11]) is not always useful against simple power attack.

2 Preliminary and Related Works

In this paper, we consider the problem of computing modular exponentiation. In the context of RSA private computation (for example, generating a digital signature or ciphertext decryption), we consider the computation of $S = M^d \bmod n$ where M, d, and n are the input message, the private key, and the modulus integer, respectively.

2.1 Exponentiation Algorithm

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 $M^d \bmod n$, practical solutions for devices with constraint computation and storage capabilities (e.g., smart cards) are usually restricted to the basic square-multiply algorithm (refer to Fig. 1 for the left-to-right/MSB-to-LSB version) and some slightly modified ones. The exponentiation algorithm in Fig. 1 processes the bits of the exponent d from the most significant bit (MSB) towards the least significant bit (LSB). An LSB-to-MSB counterpart of the algorithm in Fig. 1 can be available from most related literature.

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

Fig. 1. Classical left-to-right exponentiation algorithm

2.2 Side-Channel Attacks 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 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 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 algorithm is necessary.

A novel idea of introducing dummy 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 (refer to the well known one in Fig. 2 [8] and a recent improvement in Fig. 3 [9]) by employing this observation.

2.3 Doubling Attack

The doubling attack [7] (or called squaring attack for the scenario of exponentiation) is an SPA-based attack which works on the left-to-right square-multiply-always countermeasure (see Fig. 2). The main idea is simply to choose two

```
INPUT: $M, d = (d_{m-1} \cdots d_0)_2, n$
OUTPUT: $M^d \bmod n$
01 $T = 1$
02 for $i$ from $(m-1)$ downto 0 do
03     $T_0 = T^2 \bmod n$
04     $T_1 = T_0 \times M \bmod n$
05     $T = T_{d_i}$
06 return $T$
```

Fig. 2. (SPA protected) Square-multiply-always countermeasure

```
INPUT: $M, d = (d_{m-1} \cdots d_0)_2, n$
OUTPUT: $M^d \bmod n$
01 select a random integer $R$
02 $T_0 = R; T_1 = R^{-1} \bmod n; T_2 = M \times R^{-1} \bmod n$
03 for $i$ from $(m-1)$ downto 0 do
04     $T_0 = T_0^2 \bmod n$
05     if $(d_i = 0)$ then $T_0 = T_0 \times T_1 \bmod n$
06     else $T_0 = T_0 \times T_2 \bmod n$
07 return $T_0 \times T_1 \bmod n$
```

Fig. 3. BRIP countermeasure for exponentiation

strongly related inputs M and M^2 (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 attack 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 $79 = (1,0,0,1,1,1,1)_2$ and the two related input messages 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[1] $i-1$ for M^d and iteration i for $(M^2)^d$ will be exactly the same. So, observing collisions within computation on two collected power consumption traces enables the attacker to identify all private key bits of zero value.

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

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

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	1	$(M^9)^2$ $M^{18} \times M$	$(M^{18})^2$ $M^{36} \times M^2$
1	1	$(M^{19})^2$ $M^{38} \times M$	$(M^{38})^2$ $M^{76} \times M^2$
0	1	$(M^{39})^2$ $M^{78} \times M$	$(M^{78})^2$ $M^{156} \times M^2$
Return		M^{79}	M^{158}

To protect against the above doubling attack, the random message blinding (RMB) technique should be employed. The RMB technique blinds the original message M to $M \times R \bmod n$ before being signed with a random mask R, and removes a blinding factor $(R^d)^{-1} \bmod n$ from the result to obtain the signature S by computing $M^d = (M \times R)^d \times (R^d)^{-1} \bmod n$.

However, it has been shown in [7] that a regular (in order to be efficient) mask updating, e.g., by $R_i = R_{i-1}^2 \bmod n$ mentioned in [8], might be vulnerable to the doubling attack. So, it was suggested that a real random masking can be employed to avoid the attack.

2.4 The BRIP Countermeasure

Randomized exponentiation algorithms were recently considered as effective countermeasures against DPA by introducing randomization onto the input message or into the computational process of the algorithm in order to remove correlation between the private key and the collected power traces. One of such countermeasures is the BRIP algorithm [9] shown in Fig. 3 (it means binary expansion with random initial point/value) in which the input RSA message is blinded by multiplying with a random integer $R^{-1} \bmod n$.

It was claimed in [9] that the BRIP algorithm can be secure against SPA[2] since there will always be two operations in each iteration, i.e., the Step (04) and one of either the Step (05) or the Step (06).

[2] Of course, the version given in Fig. 3 needs some slight modification (mostly on using well known register indexing trick) to make it be truly SPA resistant. However, this version is sufficient for demonstration purpose.

Remarks. BRIP was originally proposed for ECC context to protect against RPA which requires an inversion for the computation. However, BRIP's authors say that their algorithm can also be applied in $\mathbb{Z}_n$ for cryptosystems based on integer factorization or discrete logarithm. In fact, BRIP can also work efficiently for the above systems if an efficient and secure (against related side-channel attacks, especially the doubling-like attack) random message blinding update process for $\{R_i, R_i^{-1}\}$ can be developed.

3 The Proposed Attack

In the following, an SPA by exploiting chosen input data will be pointed out which is generic and can be extended to some related attacks that will be described in this paper.

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 [12]. The validity and practicality of the employed attack assumption was claimed in [7] to be correct by experiment[3].

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 operation even if the adversary is not able to tell which exact computation is done.

Examples of collision instances to be distinguished include modular squaring and modular multiplication. For example, 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 Attack on the Square-Multiply-Always Algorithm

In the context of RSA system, given the modulus n, we observed that $(n-1)^2 \equiv 1 \pmod{n}$. This observation can be extended to obtain $(n-1)^j \equiv 1 \pmod{n}$ for any even integer j and $(n-1)^k \equiv n-1 \pmod{n}$ for any odd integer k.

Given $M = n - 1$, the square-multiply-always exponentiation algorithm in Fig.2 will have $T = (n-1)^{(d_{m-1}\cdots d_i)_2} \bmod n$ after the Step (05) of iteration i. If $T = 1$, then $(d_{m-1}\cdots d_i)_2$ is an even integer and $d_i = 0$. Otherwise, $T = n-1$ and $(d_{m-1}\cdots d_i)_2$ is an odd integer and $d_i = 1$.

By observing on a *single* collected power trace of performing the algorithm in Fig.2, the attacker can try to identify the value of T (it can only be either 1 or $n-1$) at the end of each iteration and to conduct the aforementioned derivation of each d_i. The approach used to identify the value of d_i is by SPA shown below. Given the two possible values of T at the end of iteration i, there will be only

[3] So, we did not perform another experiment.

two possible computations of the iteration $(i-1)$ shown below and which can be identified by using SPA. In the following statements, the symbol $x \rightarrow y$ means that the result of computation x will be assigned to the register y.

- if $d_i = 0$, Step (03) of the iteration $(i-1)$ performs: $1^2 \bmod n \rightarrow T_0$;
- if $d_i = 1$, Step (03) of the iteration $(i-1)$ performs: $(n-1)^2 \bmod n \rightarrow T_0$.

Notice that there are only two possible candidate computations of the Step (03). So, during the attack, it does not need to know exactly which of the two observed power consumption patterns of the Step (03) matches with the computation of $(n-1)^2 \bmod n$ (or $1^2 \bmod n$). Only two possible private keys d's will be derived and a trial-and-error approach can be used to select the correct d among the two possibilities. For example, if the MSB of d is presumed to be one, then one of the two possible d's can be selected easily.

All the private key bits can be derived except d_0 by the above SPA. However, d_0 can be known by detecting whether the final result is $T = 1$. On the other hand, in the context of RSA, d_0 is always binary one. Notice that this new SPA is much easier to mount than in the case of a conventional one (to attack the algorithm in Fig. 1) since now the square-multiply-always algorithm performs regularly such that each iteration has one modular squaring followed by a modular multiplication. Therefore, given a collected power trace, it would be much easier to identify the beginning and the end[4] of all iterations and this benefits the proposed new SPA.

Interestingly, a countermeasure originally developed to be resistant to SPA is unfortunately more vulnerable to a new SPA which is much easier to mount compared to the conventional SPA.

3.3 Attack on the BRIP Algorithm

It is interesting to note that the randomized version of square-multiply-always exponentiation in Fig. 3, the BRIP algorithm, is also vulnerable to the above proposed SPA. In the BRIP countermeasure, given $M = n-1$, we observe that at the end of iteration i:

- if $d_i = 0$: after the Step (05), $T_0 = (n-1)^{(d_{m-1} \cdots d_i)_2} \times R = R \bmod n$,
- otherwise if $d_i = 1$: after the Step (06), $T_0 = (n-1)^{(d_{m-1} \cdots d_i)_2} \times R = (n-1) \times R \bmod n$.

Based on the proposed chosen-message SPA, by observing on a *single* collected power trace, the attacker can try to identify the value of T_0 at the end of each iteration i in order to derive d_i. Given the two possible values of T_0 at the end of iteration i, there will be only two possible computations (shown below) of the iteration $(i-1)$ which can be identified by using SPA.

[4] Actually, the computation of the second part (say the Step (04)) of each iteration under the proposed chosen-message SPA are the same, i.e., $1 \times (n-1) \bmod n$. This collision helps to identify the end of each iteration.

- if $d_i = 0$, Step (04) of the iteration $(i-1)$ performs:
  $R^2 \bmod n \rightarrow T_0$;
- if $d_i = 1$, Step (04) of the iteration $(i-1)$ performs:
  $((n-1) \times R)^2 \bmod n \rightarrow T_0$.

In the above approach of SPA, all the private key bits can be derived except d_0. Similarly, d_0 can usually be obtained easily by some other approaches, and for RSA the value of d_0 is known to be one.

It is very important to notice that no matter what the value of R will be, the proposed SPA is applicable. Evidently, the initial random message blinding technique (at the Step (02)) proposed in the BRIP is not resistant against the proposed attack.

Another approach to mount the chosen-message SPA on the BRIP is possible and is shown below. Since there are only two possible input values of T_0 (either R or $(n-1)R \bmod n$) at the beginning of each iteration i, it is always true that $T_0 = R^2 \bmod n$ when finishing the Step (04). After that, one of the two possible modular *multiplications* (Step (05) or Step (06)) will be performed depending on the value of d_i.

- if $d_i = 0$, Step (05) of the iteration i performs:
  $R^2 \times R^{-1} \bmod n \rightarrow T_0$;
- if $d_i = 1$, Step (06) of the iteration i performs:
  $R^2 \times ((n-1)R^{-1}) \bmod n \rightarrow T_0$.

The same that the above attack is applicable no matter what the value of R will be. Hence, the initial random message blinding technique at the Step (02) is still in vain.

3.4 Applicability of the Attack

The proposed attack is applicable to the cases where element of order 2 exists or in any case $(-1)^k$ is computed in prime-order cases. So, the proposed attack now works against

- traditional textbook RSA decryption and signature
- RSA-OAEP decryption [10,11]
- ElGamal decryption [13]

when they are implemented based on the square-multiply-always or the BRIP algorithms. An interesting and important point to observe is that cryptographic padding (e.g., currently used RSA-OAEP) is not always useful against simple power attack. However, 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 trace(s).

But, the proposed attack does not work against the following systems

– most discrete logarithm based signature schemes
– elliptic curve discrete logarithm based decryption and signature (since ECC is usually implemented on the prime-order elliptic curves and there is no element with order 2)
– RSA signature with hash function and/or cryptographic padding.

4 Extension of the Proposed Attack

4.1 Extension to RSA with CRT

In the RSA cryptosystem, let the public modulus be $n = p \times q$ which is the product of two secret prime integers p and q each with roughly $|n|/2$ bits[5]. Prime factorization of the public modulus n can totally break the system.

The well known Chinese Remainder Theorem (CRT) technique [14,15] can be used to speedup the RSA private computation extensively, e.g., the RSA signature computation $S = M^d \bmod n$. In the RSA with CRT, we compute $S_p = M_p^{d_p} \bmod p$ and $S_q = M_q^{d_q} \bmod q$, where $M_p = M \bmod p$, $M_q = M \bmod q$, $d_p = d \bmod (p-1)$, and $d_q = d \bmod (q-1)$. Finally, the signature is computed by using the following Gauss's [14, p.68] (or other more efficient alternative) recombination algorithm

$$S = (S_p \times q \times (q^{-1} \bmod p) + S_q \times p \times (p^{-1} \bmod q)) \bmod n$$

where both $q^{-1} \bmod p$ and $p^{-1} \bmod q$ can be precomputed.

Basically, RSA with CRT is about four times faster than the straightforward approach to compute S directly in terms of bit operations. This CRT-based speedup for RSA private computation has been widely adopted in most systems, especially for implementations with smart card. In the following, we will show that the proposed SPA with chosen message is generic and can be applicable to the RSA with CRT even if the adversary does not know the secret prime integers p and q in advance.

In the attack, the adversary tries to derive d_p (or d_q) during the computation of $M_p^{d_p} \bmod p$ (or $M_q^{d_q} \bmod q$). The adversary provides the chosen message $M = n-1$ to the smart card and observes on the computation of $S_p = M_p^{d_p} \bmod p$ under the implementation of left-to-right square-multiply-always algorithm where $M_p = (n-1) \bmod p$.

We observed that $(n-1)^2 \equiv 1 \pmod n$ leads to $M_p^2 \equiv 1 \pmod p$ (and $M_q^2 \equiv 1 \pmod q$) where $M_p = (n-1) \bmod p$ (and $M_q = (n-1) \bmod p$). The above observation can be extended to obtain $M_p^j \equiv 1 \pmod p$ for any even integer j and $M_p^k \equiv M_p \pmod p$ for any odd integer k.

The square-multiply-always algorithm in Fig.2 will have $T = M_p^{(d_{p,\ell-1} \cdots d_{p,i})_2} \bmod p$ (where $\ell = |d_p|$) after the Step (05) of iteration i. If $T = 1$, then $(d_{p,\ell-1} \cdots d_{p,i})_2$ is an even integer and $d_{p,i} = 0$. Otherwise, $T = M_p$ and

[5] The symbol $|n|$ means the number of bits of binary representation of n.

$(d_{p,\ell-1}\cdots d_{p,i})_2$ is an odd integer and $d_{p,i} = 1$. By observing on a single collected power trace, the adversary can try to identify the value of T (in order to derive the value of $d_{p,i}$) by SPA. The analysis is summarized in the following.

– if $d_{p,i} = 0$, Step (03) of the iteration $(i-1)$ performs:
  $1^2 \bmod p \rightarrow T_0$;
– if $d_{p,i} = 1$, Step (03) of the iteration $(i-1)$ performs:
  $M_p^2 \bmod n \rightarrow T_0$.

Notice that in the proposed attack the adversary does not need to know the value of $M_p = (n-1) \bmod p$. Recall that p is unknown to the adversary. All the private key bits of d_p can be derived except $d_{p,0}$ by the above SPA. However, $d_{p,0}$ is binary one in the usual case of RSA parameters selection in which d is an odd integer and $p-1$ is an even integer.

Notice also that the proposed attack is still applicable to the RSA with CRT speedup if the BRIP exponentiation algorithm will be employed. The details of this attack can be obtained similarly, so the analysis is omitted here.

It was already well known that given the *partial* private key d_p (or d_q) and the public parameters n and e, both the factorization of n and also the private key d can be available directly. The approach to factorize n is given below. Randomly select an integer X and computes $Y = X^e \bmod n$. Evidently, $Y^d \equiv X \pmod{n}$ and this leads to $Y^{d_p} \equiv X \pmod{p}$ or equivalently $Y^{d_p} - X \equiv 0 \pmod{p}$. With the knowledge of d_p obtained by the proposed SPA, the adversary can derive p by computing

$$p = \gcd(Y^{d_p} - X, n) = \gcd(Y^{d_p} - X \bmod n, n).$$

With p and q, the RSA private key d can be computed. So, the adversary does not need to analyze on the computation of $M_q^{d_q} \bmod q$ in order to derive d_q.

4.2 Extension to Randomly Chosen-Message Attack

An important question to answer about the proposed attack is that whether identification of $n-1$ as input message can be a sufficient countermeasure. Basically, for some cases, the answer is negative because of the following extended and more powerful attack with slightly different assumption. In the extended attack, two power consumption traces are necessary, but the related input messages are far from a fixed and specific value of $n-1$.

The extended attack on the square-multiply-always algorithm (refer to Fig. 2) performs as follows. The adversary selects a pair of input messages M_1 (can be any random message) and $M_2 = M_1 \times (n-1) \bmod n$ and collects the two related power consumption traces of computing $M_1^d \bmod n$ and $M_2^d \bmod n$. The adversary can mount successfully an SPA by observing the collision of middle results between these two power consumption traces in order to identify zero bits of the private key d. The basic idea is that collision will happen when

$$M_1^k \equiv M_2^k \pmod{n}$$

if the exponent k is an even integer.

In the SPA-protected exponentiation algorithm in Fig. 2, if some key bit d_i is zero, then collision on values of T among the two collected power consumption traces can be detected at the end of iteration i since

$$M_2^{(d_{m-1}\cdots d_i)_2} \equiv M_1^{(d_{m-1}\cdots d_i)_2} \cdot (n-1)^{(d_{m-1}\cdots d_i)_2} \equiv M_1^{(d_{m-1}\cdots d_i)_2} \pmod{n}.$$

Therefore, the Step (`03`) of iteration $(i-1)$ will be a collision instance (the same operation with same operand). This is summarized in the following.

- if $d_i = 0$, Step (`03`) of the iteration $(i-1)$ of both the two observed computations perform the same:
  $(M_1^{(d_{m-1}\cdots d_i)_2})^2 \bmod n \rightarrow T_0$;
- if $d_i = 1$, Step (`03`) of the iteration $(i-1)$ of both the two observed computations perform differently:
  either $(M_1^{(d_{m-1}\cdots d_i)_2})^2 \bmod n \rightarrow T_0$
  or $((n-1) \times M_1^{(d_{m-1}\cdots d_i)_2})^2 \bmod n \rightarrow T_0$.

This SPA enables the adversary to derive all the private key bits of d except d_0.

It is interesting to notice that the above extended attack can be considered as a variant of the doubling attack [7]. Both attacks exploit two related chosen messages (however with different forms) and collision detection by SPA on squaring operations (the Step (`03`)) within two exponentiations.

There is however some difference between the two attacks. In the doubling attack, the adversary observes on collision occurred in two "different" iterations of two power consumption traces. On the contrary, in the proposed extended attack, the adversary observes on collision occurred in the "same" iteration (in fact, the exactly same timing duration) of two power consumption traces. ¿From practical point of view for the SPA scenario, the collision detection on the *same* iteration will be more or less easier than on *different* iterations. This is especially the case when both attacks deal with random input messages, and try to observe on collision of random computations that will be varying/different during the whole process of the algorithm.

One possibility to detect the proposed extended attack is to check the relationship[6] between two input messages on whether $M_j = M_i \times (n-1) \bmod n$ for every pair of i and j. It is however extremely difficult and infeasible to detect all the relationship of input messages since M_i and M_j may not be two consecutive input messages to mount the attack. By the way, it is infeasible to store all previous input messages in order to perform the detection, especially for the applications with smart card.

5 Conclusions

It was previously believed that the BRIP algorithm can be an effective countermeasure against SPA and DPA. However, our research reveals that the BRIP

[6] In the doubling attack, similar approach is to check whether $M_j = M_i^2 \bmod n$ for every pair of i and j.

algorithm is vulnerable to a new SPA. By the way, the well known left-to-right square-multiply-always algorithm (for SPA resistance) is also shown to be insecure against the proposed attack.

Notice especially that the proposed SPA can also be applicable to the scenario of RSA decryption even if RSA-OAEP padding will be considered. The reason is that the ciphertext validity checking is performed after the private exponentiation computation. So, the attacker still can collect the power trace(s).

Some extensions of the attack are also considered in this paper which include the application on RSA with CRT speedup and how to use randomly chosen messages to mount a similar attack.

Acknowledgment

The authors would like to thank Dr. Atsuko Miyaji, Dr. Shiho Moriai, and several anonymous referees for their professional and in-depth 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. V. Miller, "Uses of elliptic curve in cryptography," *Advances in Cryptology – CRYPTO '85*, LNCS 218, pp. 417-426, Springer-Verlag, 1985.
4. N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, Jan. 1987.
5. L. Goubin, "A refined power-analysis attack on elliptic curve cryptosystems," *Proc. of Public Key Cryptography – PKC '03*, LNCS 2567, pp. 199–210, Springer-Verlag, 2003.
6. T. Akishita and T. Takagi, "Zero-value point attacks on elliptic curve cryptosystem," *Proc. of Information Security Conference – ISC '03*, LNCS 2851, pp. 218–233, Springer-Verlag, 2003.
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. J. 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.
9. H. Mamiya, A. Miyaji, and H. Morimoto, "Efficient countermeasures against RPA, DPA, and SPA," *Proc. of Cryptographic Hardware and Embedded Systems – CHES '04*, LNCS 3156, pp. 343–356, Springer-Verlag, 2004.
10. PKCS #1 v2.1, "RSA Cryptography Standard", 5 January 2001. `http://www.rsasecurity.com/rsalabs/pkcs/`

11. 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.
12. 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.
13. 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.
14. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997.
15. J.-J. Quisquater and C. Couvreur, "Fast decipherment algorithm for RSA public key cryptosystem," *Electronics Letters*, vol. 18, no. 21, pp. 905–907, 1982.