Extended Fault Based Attack against Discrete Logarithm Based Public Key Cryptosystems

Sung-Ming Yen
Dept of Computer Science and Information Engineering
National Central University
Chung-Li, Taiwan 320, R.O.C.
Email: yensm@csie.ncu.edu.tw

Chi-Dian Wu
Dept of Computer Science and Information Engineering
National Central University
Chung-Li, Taiwan 320, R.O.C.
Email: cs122016@csie.ncu.edu.tw

***Abstract*—Since Bellcore's researchers proposed fault based attacks, these attacks have become serious threats to the implementation of cryptosystems. Boneh *et al.* first proposed a fault based attack against the exponentiation algorithm for RSA, and some variants of attack were proposed later. However, the previous variants of similar attack are applicable only to the right-to-left exponentiation algorithm and none of these attacks can be successfully applied to the left-to-right alternative algorithm since 1997. In this paper, we focus on cryptosystems operated under prime-order groups and emphasize that an extended fault based attack against implementations using the left-to-right exponentiation algorithm is possible. Our attack can also be applied to the Montgomery ladder algorithm which is a well-known countermeasure against some critical physical attacks.**

***Keywords*-exponentiation algorithm; hardware fault attack; physical attack; public key cryptosystem.**

I. INTRODUCTION

In the past, cryptographers only analyzed the security of cryptosystems by mathematics. However, when cryptosystems are implemented on physical devices, it brings new threats which had never been considered carefully. These new threats are called the physical attacks, such as the side-channel attack [14] and the fault based attacks [1], [4], [8]. Physical attacks utilize the power consumption and program execution time, or disturb the program execution to infer the secret information stored inside the devices, even though these cryptosystems have been proved secure with mathematical approach. So, when implementing cryptosystems, it is usually essential to prevent such kinds of attack.

Both exponentiation and scalar multiplication are the most central computations for many public key cryptosystems. To evaluate exponentiation or scalar multiplication, the left-to-right and the right-to-left algorithms are the two most widely employed methods. Fault based attack was first introduced in 1997, and afterwards many kinds of fault based attack have been proposed to break a variety of cryptosystems. For example, Boneh *et al.* [8] proposed a fault based attack against the right-to-left exponentiation algorithm for RSA by injecting random faults during the computation to reveal the private exponent. Biehl *et al.* presented a similar attack on elliptic curve cryptosystems (ECC) in 2000 and showed that the secret scalar of a scalar multiplication can be revealed by providing illegal input parameters [4]. Berzati *et al.* modified Boneh *et al.*'s attack in 2008 [2]. Biham and Shamir proposed a differential fault attack (DFA) [5] against symmetric key cryptosystems, e.g., DES. All these researches show that fault based attacks are powerful and dangerous to cryptosystem implementations, especially for those on smart cards.

The aforementioned fault based attacks against public key cryptosystems target at the right-to-left exponentiation algorithm (or the right-to-left scalar multiplication for ECC), while in this paper we extend this kind of fault based attack to the left-to-right exponentiation algorithm. The proposed attack assumes the knowledge of the order of a group and the order must be a prime. For performance reasons or security reasons, many important public key cryptosystems, such as Schnorr scheme [19] and ECC [13] (e.g., elliptic curve Diffie-Hellman key exchange [17]), the group order is a prime integer and it is a public information. So, this attack assumption is reasonable and the proposed attack can be applied to these widely employed cryptosystems. Moreover, the proposed attack can also be extended easily to the Montgomery ladder algorithm.

This paper is organized as follows. In Section II, we first introduce RSA and Diffie-Hellman cryptosystems. We also show the algorithms to compute exponentiation. In Section III, the previous fault based attacks against the exponentiation algorithm are reviewed. In Section IV, we propose an extended attack against the left-to-right exponentiation algorithm and show how to apply this attack to the Montgomery ladder algorithm. Section V concludes the paper.

II. PRELIMINARY BACKGROUND

A. The RSA Cryptosystem

In the RSA [18] cryptosystem, let p and q be two large primes kept secret to the public and $N = p \cdot q$ is the RSA public modulus. The public key e should be relatively prime to $\phi(N) = (p-1) \cdot (q-1)$, and d is the corresponding

 ISBN: 978-1-61208-010-9

private key satisfying $e \cdot d \equiv 1 \pmod{\phi(N)}$. To encrypt a message m with the public key e, we compute $C = m^e \bmod N$ and to decrypt a cipher C with the private key d, we compute $m = C^d \bmod N$. Signing a message m, the private computation $S = m^d \bmod N$ is performed and verification of a signature S is to check whether $m = S^e \bmod N$.

B. Public Key Cryptosystems Based on Discrete Logarithm

Many important public key cryptosystems have been designed with their security based on solving the discrete logarithm problem, e.g., Diffie-Hellman key exchange [9], ElGamal scheme [10], Schnorr scheme [19], and ECC [13] (e.g., elliptic curve Diffie-Hellman key exchange [17]). These cryptosystems are constructed over a finite cyclic group and solving the discrete logarithm problem over this group is believed to be hard. The multiplicative group and the additive group on an elliptic curve are two widely used groups for constructing this kind of cryptosystems. The Diffie-Hellman key exchange scheme is reviewed in the following.

Key generation: Let $\mathbb{G}$ be a cyclic group of order p and g is a generator. Each user selects a random integer $x_i \in \mathbb{Z}_p$ as the private key and the public key is $y_i = g^{x_i}$.

Key exchange: To exchange a shared key k_{ab} with another party, user a receives the public key y_b from user b and computes the shared key $k_{ab} = y_b^{x_a}$.

C. Exponentiation Algorithms

Let $d = \sum_{i=0}^{n-1} d_i\, 2^i$ be the binary expression of the exponent d. An exponentiation algorithm computes the value of m^d given the base number m and the exponent d. A variety of efficient exponentiation algorithms have been proposed so far to compute m^d while the binary left-to-right square-and-multiply algorithm (refer to Figure 1) and the right-to-left square-and-multiply algorithm (refer to Figure 2) are the two most widely employed methods [15].

In this paper, the iteration number of the left-to-right exponentiation algorithm is denoted decreasingly from $(n-1)$ downward towards zero and that of the right-to-left version is denoted increasingly from zero upward towards $(n-1)$.

III. Review of Fault Based Attacks against Exponentiation Algorithm

Some fault based attacks against the exponentiation or the scalar multiplication have been proposed [1], [2], [4], [7], [8], [11] so far and can be classified into two categories. The first category of attacks modify the value of the exponent and the second category of attacks disturb the intermediate value of the exponentiation computation, e.g., the value $R[0]$ in the right-to-left exponentiation algorithm.

A. Fault Based Attack on the Exponent

Bao *et al.* [1] proposed a fault based attack to threaten some cryptosystems, e.g., the RSA system. The fault model

```
     Input: m, d = (d_{n-1} ··· d_0)_2
     Output: m^d
01   R[0] ← 1
02   for i from n − 1 downto 0 do
03     R[0] ← R[0]^2
04     if (d_i = 1) then
         R[0] ← R[0] · m
05   return R[0]
```

Figure 1. Left-to-right exponentiation.

```
     Input: m, d = (d_{n-1} ··· d_0)_2
     Output: m^d
01   R[0] ← 1; R[1] ← m
02   for i from 0 to n − 1 do
03     if (d_i = 1) then
         R[0] ← R[0] · R[1]
04     R[1] ← R[1]^2
05   return R[0]
```

Figure 2. Right-to-left exponentiation.

of this attack is to induce a one-bit fault into the exponent d such that the binary value of certain bit, say d_j, will be inverted. Let d' be the faulty exponent and the faulty output of the exponentiation is $S' = m^{d'} = m^{(\sum_{i=0, i\neq j}^{n-1} d_i 2^i) + \overline{d_j} 2^j}$ where $\overline{d_j}$ is the one's complement of d_j. Given the aforementioned faulty output S' and the corresponding correct one S, the adversary can identify the value of the bit d_j by analyzing the value of $\frac{S'}{S} = m^{(\overline{d_j} - d_j) 2^j}$. We have $\frac{S'}{S} = \frac{1}{m^{2^j}}$ if $d_j = 1$, and $\frac{S'}{S} = m^{2^j}$ if $d_j = 0$.

The attack is also applicable to the multi-bit-fault model. Assume d_j and d_k are inverted, the adversary can derive the values of both bits by analyzing $\frac{S'}{S} = m^{(\overline{d_j} - d_j) 2^j} \cdot m^{(\overline{d_k} - d_k) 2^k}$. In [11], Joye *et al.* extended the attack such that only the faulty result S' is needed with the knowledge of the plaintext m and additionally its order.

B. Bellcore's Fault Based Attack against the Right-to-left Exponentiation Algorithm

Bellcore's researchers proposed the fault based attack [8] to defeat the RSA private computation with the right-to-left exponentiation algorithm (refer to Figure 2). The fault model of Bellcore's attack is a random one-bit fault injected into the intermediate value of $R[0]$ at the end of the iteration $(j-1)$ or at the end of the Step (03) of that iteration. The faulty result of $R[0]$ at the end of the iteration $(j-1)$ can be expressed as

$$R[0] = (m_j^{\sum_{i=0}^{j-1} d_i 2^i}) \pm 2^b \bmod N$$

where 2^b is the injected error in which $0 \leq b \leq n-1$ (n is the bit length of N) and m_j is the base number, e.g., the plaintext in a signature.

 ISBN: 978-1-61208-010-9

The principle of Bellcore's attack. Bellcore's attack consists of the following steps.

1) The adversary collects sufficient faulty signatures S'_j with the corresponding plaintexts m_j by injecting a random one-bit fault into $R[0]$ during each execution of m^d.
2) The adversary analyzes each faulty signature S'_j with the plaintext m_j and he has

$$\begin{aligned} S'_j &= (m_j^{\sum_{i=0}^{j-1} d_i 2^i} \pm 2^b) \cdot m_j^{\sum_{i=j}^{n-1} d_i 2^i} \bmod N \\ &= S_j \pm (2^b \cdot m_j^{\sum_{i=j}^{n-1} d_i 2^i}) \bmod N, \end{aligned}$$

or $S_j = S'_j \pm 2^b \cdot m_j^{\omega} \bmod N$ where $\omega = \sum_{i=j}^{n-1} d_i 2^i$.
3) With the public exponent e and the collected pairs of (S'_j, m_j), the adversary tests all the possible candidates of b and ω by checking whether

$$m_j = (S'_j \pm 2^b \cdot m_j^{\omega})^e \bmod N.$$

To derive the value of ω in each test needs a known part of binary representation of d and at most l unknown bits where l denotes the longest distance between two nearby iterations at which random faults occurred. Suppose the position at which the random fault occurred on $R[0]$ is unknown (i.e., unknown $0 \leq b \leq n-1$) and the time at which the random fault occurred during the exponentiation is unknown (i.e., unknown j) and uniformly distributed over $[0, n-1]$. Let k be the number of necessary collected pairs of (S'_j, m_j). The number of tests necessary to recover d is at most $k \cdot (n \cdot k \cdot \sum_{r=1}^{l} 2^r)$ and the complexity of this attack is

$$\mathrm{O}(n \cdot k^2 \cdot 2^l).$$

In [8, Theorem 3], Boneh *et al.* proved that to recover d with probability at least $\frac{1}{2}$ requires about $(n/l)\log(2n)$ pairs of (S'_j, m_j) and the complexity of the attack becomes

$$\mathrm{O}(n^3 \cdot \log^2(n) \cdot 2^l / l^2).$$

C. *Berzati* et al.*'s Fault Based Attack*

In 2008, Berzati *et al.* [2] modified the Bellcore's attack by injecting random one-byte faults into the RSA public modulus N right after some iterations instead of the intermediate value of $R[0]$ of the exponentiation computation. In Berzati *et al.*'s attack, the faulty modulus N can be expressed as $N' = N \pm R_8 \cdot 2^{8i}$ where R_8 is a nonzero random byte value and $i \in [0, \frac{n}{8} - 1]$.

The principle of Berzati *et al.*'s attack. This attack needs to collect a correct signature S and k faulty signatures S'_j. The values of $R[0]$ and $R[1]$ after the computation within the iteration $(j-1)$ are $m^{\sum_{i=0}^{j-1} d_i 2^i} \bmod N$ and $m^{2^j} \bmod N$, respectively. Suppose the fault upon N occurs at the end of the iteration $(j-1)$. The value of the collected faulty signature S'_j becomes

$$S'_j = \left((m^{\sum_{i=0}^{j-1} d_i 2^i} \bmod N) \cdot (m^{2^j} \bmod N)^{\sum_{i=j}^{n-1} d_i 2^i}\right) \bmod N'.$$

Let $\omega = \sum_{i=j}^{n-1} d_i 2^i$, so the correct signature can be expressed as $S = m^{\omega + \sum_{i=0}^{j-1} d_i 2^i} \bmod N$. Based on the above expression of S and all the possible candidates of ω and N', the adversary can compute

$$S'_{(\omega, N')} = \left((S \cdot m^{-\omega} \bmod N) \cdot (m^{2^j} \bmod N)^{\omega}\right) \bmod N'$$

and the correct values of ω and N' can be determined by checking whether

$$S'_{(\omega, N')} \equiv S'_j \pmod{N'}$$

on all collected faulty outputs S'_j.

Suppose each check requires to determine l unknown bits of ω and $(2^8 - 1) \cdot \frac{n}{8}$ possible byte faults on N, hence the complexity of Berzati *et al.*'s attack is

$$\mathrm{O}((2^8 - 1) \cdot \frac{n}{8} \cdot k \cdot 2^l).$$

It was claimed that under the assumption of known values of all j (i.e., the time the faults occurred) [2] the complexity of the attack becomes

$$\mathrm{O}((2^8 - 1) \cdot \frac{n^2}{8l} \cdot 2^l), \quad \text{if } k = \frac{n}{l}.$$

IV. The Proposed Fault Based Attack against the Left-to-right Exponentiation Algorithm

The proposed fault based attack is based on Bellcore's attack [8] and Berzati *et al.*'s attack [2], but the attack targets at the left-to-right exponentiation algorithm with the additional knowledge of the group order which is a prime.

A. *Fault Model*

The proposed fault based attack is based on modifying the intermediate value of $R[0]$ within the left-to-right exponentiation algorithm (refer to Figure 1) by injecting random one-byte faults. This random one-byte fault model (i.e., the fault model #3 in [6]) has been considered practical and widely adopted in many fault based attacks [2], [3], [20]. The faulty value of $R[0]$ can be expressed as

$$R[0]' = R[0] \pm R_8 \cdot 2^{8i}$$

where R_8 is a nonzero random byte value and $i \in [0, \frac{n}{8} - 1]$ both are unknown to the adversary.

 ISBN: 978-1-61208-010-9

B. Principle of the Proposed Attack

The proposed attack needs to collect a correct output S and k faulty outputs S'_j by injecting random one-byte faults into the intermediate value of $R[0]$ each at the end of the iteration j where the iteration number j is unknown to the adversary and is uniformly distributed over $[0, n-1]$.

The intermediate value of $R[0]$ after the computation within the iteration j (denoted as $R[0,j]$) is $R[0,j] = m^{\sum_{i=j}^{n-1} d_i 2^{i-j}}$. The value of correct output $S = m^d$ can be expressed as

$$S = m^d = R[0,j]^{2^j} \cdot m^{\sum_{i=0}^{j-1} d_i 2^i}.$$

Suppose the fault upon $R[0]$ occurs at the end of the iteration j. The value of the collected faulty output S'_j becomes

$$S'_j = (R[0,j] \pm \varepsilon)^{2^j} \cdot m^{\sum_{i=0}^{j-1} d_i 2^i} = (R[0,j] \pm \varepsilon)^{2^j} \cdot m^{\omega}$$

where $\varepsilon = R_8 \cdot 2^{8i},\ i \in [0, \frac{n}{8} - 1]$ and $\omega = \sum_{i=0}^{j-1} d_i 2^i$.

To derive the partial value of d, say ω, the adversary needs a previously known $\sum_{i=0}^{r} d_i 2^i$ $(r < j-1)$ and needs to guess at most l unknown bits of d (say $(d_{j-1}, \ldots, d_{r+1})_2$) where l denotes the longest distance between two nearby iterations at which random faults injected.

Suppose that the order of the group is a prime integer and which is known to the adversary. The correct value of $R[0,j]$ can therefore be derived from the correct output S by

$$R[0,j] = (S \cdot m^{-\sum_{i=0}^{j-1} d_i 2^i})^{(2^j)^{-1}} = (S \cdot m^{-\omega})^{(2^j)^{-1}}.$$

The reason of the assumption of a known prime order is to enable the adversary to compute $(2^j)^{-1}$.

Based on the derived ω, $R[0,j]$, and all the possible candidates of ε, the adversary can compute

$$S'_{(\omega,\varepsilon)} = \left((S \cdot m^{-\omega})^{(2^j)^{-1}} \pm \varepsilon\right)^{2^j} \cdot m^{\omega}$$

and the correct values of ω and ε can be verified by checking whether

$$S'_{(\omega,\varepsilon)} \equiv S'_j$$

on all collected faulty outputs S'_j.

Based on the above proposed attack the adversary can recover the binary expression of the private exponent d from the least significant bits towards the most significant bits. However, the last few bits with the most significant weightings cannot be derived by the attack. These few bits, say $(d_{n-1}, \ldots, d_t)_2$ and t is the maximum value for which a fault occurred at the iteration t, can only be obtained by other approaches, e.g., a brute force search. Bellcore's attack and Berzati *et al.*'s attack share the same property of the proposed attack but the brute force search happens at the least significant bits.

C. Practicability of the Attack

We wish to point out that injecting a one-byte fault into the intermediate value of a register of an exponentiation algorithm assumed in the proposed attack would be more practical than injecting a one-bit fault into a register assumed in the Bellcore's attack. Moreover, the aforementioned assumption made in the proposed attack might be much more practical than injecting a one-byte fault into the storage of a cryptographic parameter, e.g., the RSA public modulus N assumed in Berzati *et al.*'s attack. The reason is that usually a cryptographic parameter will be stored in a non-volatile storage, e.g., flash memory or ROM, and a previous random one-byte fault once injected will be difficult to remove and a new one-byte fault to be injected again which is implicitly assumed in Berzati *et al.*'s attack. So, among the aforementioned three fault based attacks, the proposed attack in this paper demonstrates a higher feasibility.

The computation time of an exponentiation algorithm dominates the performance of many cryptosystems. To improve the performance of cryptosystems, especially for those with their security based on solving the discrete logarithm problem, the group $\mathbb{G}$ is usually replaced by a subgroup with a prime order q of which when binary represented the number of bits is much smaller than that of p. This technique was first employed in the Schnorr scheme [19]. For security reasons, elliptic curve based cryptosystems, e.g., elliptic curve Diffie-Hellman key exchange [17], usually choose a prime order [16]. Both the aforementioned prime order of a multiplicative subgroup and the prime order of an elliptic curve are public informations. So, the assumption made in the proposed attack is reasonable.

D. Complexity of the Proposed Attack and Comparison with Other Attacks

Suppose the byte-fault *pattern* $R_8 \in [1, 2^8 - 1]$, the byte *position* $i \in [0, \frac{n}{8} - 1]$ at which the random byte fault occurred on $R[0]$, and the *time* (say, the iteration number $j \in [0, n-1]$) at which the random byte fault occurred during the exponentiation are all unknown to the adversary. In the proposed attack, to perform test of the relationship $S'_{(\omega,\varepsilon)} \equiv S'_j$, the adversary needs to try all the possible candidates of ω and ε to identify the correct values of both ω and ε.

A segment of at most l least significant bits of d will be derived first when the correct value of ω can be identified, and the exact value of the corresponding iteration number j will be found as well. At most $\sum_{r=1}^{l} 2^r$ possible ω will be tested. Other portion of the binary representation of d can be derived in the same approach towards the most significant bits. The correct value of ε can be identify from one of the possible $(2^8 - 1) \cdot \frac{n}{8}$ one-byte faults occurred on $R[0]$. All in all, the number of tests necessary to recover d is at most $k \cdot ((2^8 - 1) \cdot \frac{n}{8} \cdot k \cdot \sum_{r=1}^{l} 2^r)$ and the complexity of this

 ISBN: 978-1-61208-010-9

attack becomes

$$\mathrm{O}((2^8-1)\cdot\frac{n}{8}\cdot k^2\cdot 2^l).$$

The complexity of the proposed attack is basically similar to Bellcore's attack, and the difference is that we assume random one-byte faults while Bellcore's attack assumes random one-bit faults. The numbers of possible fault patterns in the proposed attack and Bellcore's attack are $(2^8-1)\cdot\frac{n}{8}$ and n, respectively. However, the numbers of necessary faulty outputs (i.e., k) of both attacks are different, and both attacks assume different fault models.

With the knowledge of all values of iteration number j (i.e., the time the random byte faults occurred) as assumed in Berzati *et al.*'s attack [2], the complexity of the proposed attack can be reduced to $\mathrm{O}((2^8-1)\cdot\frac{n}{8}\cdot k\cdot 2^l)$. Let $k=\frac{n}{l}$, the complexity becomes

$$\mathrm{O}((2^8-1)\cdot\frac{n^2}{8l}\cdot 2^l)$$

which is the same as Berzati *et al.*'s attack.

E. Attack Extension to the Montgomery Ladder

In [12], an exponentiation algorithm based on the Montgomery ladder was proposed to prevent the SPA attack [14], the computational safe-error attack [22], and the memory safe-error attack [21]. The Montgomery ladder algorithm shown in Figure 3 behaves regularly and accordingly it is secure against the SPA attack. Most specially, there is no dummy computation within the Montgomery ladder algorithm so it can be secure against the computational safe-error attack. Any random computational fault occurred will lead to a faulty result of m^d.

	Input: $m, d=(d_{n-1}\cdots d_0)_2$ Output: m^d
01	$R[0]\leftarrow 1$; $R[1]\leftarrow m$
02	for i from $n-1$ downto 0 do
03	$R[\overline{d_i}]\leftarrow R[0]\cdot R[1]$
04	$R[d_i]\leftarrow R[d_i]^2$
05	return $R[0]$

Figure 3. Montgomery ladder algorithm.

The proposed fault based attack can be extended to the Montgomery ladder algorithm with the same random one-byte fault model. The byte fault will be injected into the intermediate value of $R[0]$ at the end of a specific iteration j during the exponentiation. The adversary needs to collect sufficient faulty outputs S'_j and a correct output S.

Principle of the attack. According to the basic principle of Montgomery ladder, the intermediate values of $R[0]$ and $R[1]$ after the computation within the iteration j are $R[0,j]=m^{\sum_{i=j}^{n-1}d_i2^{i-j}}$ and $R[1,j]=m^{(\sum_{i=j}^{n-1}d_i2^{i-j})+1}=R[0,j]\cdot m$, respectively. So, the output of the algorithm can be expressed as

$$S=m^d=R[0,j]^{2^j}\cdot m^{\sum_{i=0}^{j-1}d_i2^i}.$$

Providing two initial values $B_0=m^a$ and $B_1=m^{a+1}$ for some integer a, and a k-bit binary bit string $(e_{k-1}\cdots e_0)_2$ representing an exponent $e=\sum_{i=0}^{k-1}e_i2^i$, we define a function $\mathtt{Mont}(B_0,B_1,(e_{k-1}\cdots e_0)_2)$ which represents the output B_0^e of the Montgomery ladder algorithm. The output $S=m^d$ of the Montgomery ladder algorithm can therefore be expressed as $\mathtt{Mont}(1,m,(d_{n-1}\cdots d_0)_2)$ or $\mathtt{Mont}(R[0,j],R[1,j],(d_{j-1}\cdots d_0)_2)$. If a fault ε is injected into $R[0]$ at the end of the iteration j, then the faulty output S'_j of the Montgomery ladder algorithm becomes

$$S'_j=\mathtt{Mont}(R[0,j]\pm\varepsilon,R[1,j],(d_{j-1}\cdots d_0)_2)$$

where $\varepsilon=R_8\cdot 2^{8i},\ i\in[0,\frac{n}{8}-1]$ and ω represents the value $\sum_{i=0}^{j-1}d_i2^i$. Here we also assume that the order of the group is a public prime integer, hence the values of $R[0,j]$ and $R[1,j]$ can therefore be derived based on the correct output S by

$$\begin{aligned}R[0,j] &= (S\cdot m^{-\sum_{i=0}^{j-1}d_i2^i})^{(2^j)^{-1}}=(S\cdot m^{-\omega})^{(2^j)^{-1}}\\ R[1,j] &= R[0,j]\cdot m.\end{aligned}$$

Based on all the possible candidates of bit string $(d_{j-1}\cdots d_0)_2$ (accordingly the value $\omega=\sum_{i=0}^{j-1}d_i2^i$) and injected byte fault ε, the adversary can compute

$$S'_{(\omega,\varepsilon)}=\mathtt{Mont}\left((Sm^{-\omega})^{(2^j)^{-1}}\pm\varepsilon,(Sm^{-\omega})^{(2^j)^{-1}}m,(d_{j-1}\cdots d_0)_2\right).$$

The correct values of ω and ε can be verified by checking whether $S'_{(\omega,\varepsilon)}\equiv S'_j$ on all collected faulty outputs S'_j.

The complexity of the above attack on the Montgomery ladder algorithm is basically the same as that attacking the left-to-right exponentiation algorithm. An alternative attack approach is that the byte faults are injected to the intermediate value of $R[1]$ instead of $R[0]$ and in this case the faulty output is assumed to be $S'_j=\mathtt{Mont}(R[0,j],R[1,j]\pm\varepsilon,(d_{j-1}\cdots d_0)_2)$.

Practicability of the attack. The proposed fault based attack can break not only the well-known conventional left-to-right exponentiation algorithm but also the enhanced algorithm against side-channel attack and safe-error attack, say the Montgomery ladder algorithm. In fact, the Montgomery ladder algorithm might be more vulnerable to the proposed attack because the algorithm behaves regularly in each iteration.

In the Montgomery ladder algorithm, each iteration performs two similar operations and the total number of operations to be performed within the algorithm is always $2n$. Therefore, a very precise estimation of the computation time

 ISBN: 978-1-61208-010-9

for a single iteration is accessible after a few experiments. These experiments can even be performed upon other similar devices. From the view point of controllability of fault occurrence time (e.g., fault injected at the end of an iteration) and accordingly the feasibility of an attack, the proposed extended attack on the Montgomery ladder algorithm becomes more practical than all the previous attacks.

V. Conclusion

In this paper, based on the previous fault based attacks against the right-to-left exponentiation algorithm, we propose a new attack against the left-to-right exponentiation algorithm on some public key cryptosystems, such as Diffie-Hellman key exchange and ECC, if they are constructed under a group with a prime order. The complexity of the proposed attack is the same as that of the previous related attacks. Moreover, the proposed attack can also be extended to threaten the Montgomery ladder algorithm and this extended attack could be even more practical than all other related attacks.

Acknowledgment

This research was supported in part by the National Science Council of the Republic of China under contract NSC 98-2221-E-008-047-MY3.

References

[1] F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimbalu, and T. Ngair, "Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults," in *Proc. Security Protocols Workshop 1997*, LNCS 1361, Springer-Verlag, pp. 115–124.

[2] A. Berzati, C. Canovas, and L. Goubin, "Perturbing RSA public keys: an improved attack," in *Proc. CHES 2008*, LNCS 5154, Springer-Verlag, pp. 380–395.

[3] A. Berzati, C. Canovas, and L. Goubin, "(In)security against fault injection attacks for CRT-RSA implementations," in *Proc. Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC 2008*, pp. 101–107.

[4] I. Biehl, B. Meyer, and V. Muller, "Differential fault attacks on elliptic curve cryptosystems," in *Proc. CRYPTO 2000*, LNCS 1880, Springer-Verlag, pp. 131–146.

[5] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Proc. CRYPTO 1997*, LNCS 1294, Springer-Verlag, pp. 513–525.

[6] J. Blömer, M. Otto, and J. P. Seifert, "A new CRT-RSA algorithm secure against bellcore attacks," in *Proc. ACM CCS 2003*, ACM Press, pp. 311–320.

[7] J. Blömer, M. Otto, and J. P. Seifert, "Sign change fault attacks on elliptic curve cryptosystems," in *Proc. Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC 2006*, LNCS 4236, Springer-Verlag, pp. 36–52.

[8] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Proc. EUROCRYPT 1997*, LNCS 1233, Springer-Verlag, pp. 37–51.

[9] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[10] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proc. CRYPTO 1984*, LNCS 196, Springer-Verlag, pp. 10–18.

[11] M. Joye, J. J. Quisquater, F. Bao, and R. H. Deng, "RSA-type signatures in the presence of transient faults," in *Proc. Cryptography and Coding 1997*, LNCS 1355, Springer-Verlag, pp. 155–160.

[12] M. Joye and S. M. Yen, "The Montgomery powering ladder," in *Proc. CHES 2002*, LNCS 2523, Springer-Verlag, pp. 291–302.

[13] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203–209, 1987.

[14] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. CRYPTO 1999*, LNCS 1666, Springer-Verlag, pp. 388–397.

[15] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.

[16] National Institute of Standards and Technology, *Recommended Elliptic Curves for Federal Government Use*. In the appendix of FIPS 186-2.

[17] National Institute of Standards and Technology, *Special Publication 800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*. March, 2006.

[18] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Comm. of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[19] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. Crypto 1989*, LNCS 435, Springer-Verlag, pp. 239–252.

[20] D. Wagner, "Cryptanalysis of a provably secure CRT-RSA algorithm," in *Proc. ACM CCS 2004*, ACM press, pp. 311–320.

[21] S. M. Yen and M. Joye, "Checking before output may not be enough against fault-based cryptanalysis," *IEEE Trans. Computers*, vol. 49, no. 9, pp. 967–970, 2000.

[22] S. M. Yen, S. Kim, S. Lim and S. Moon, "A countermeasure against one physical cryptanalysis may benefit another attack," in *Proc. ICISC 2001*, LNCS 2288, Springer-Verlag, pp. 414–427.

 ISBN: 978-1-61208-010-9