Research Article

Side-Channel Attacks and Countermeasures for Identity-Based Cryptographic Algorithm SM9

Qi Zhang ^{1,2}, An Wang ¹, Yongchuan Niu ³, Ning Shang ¹, Rixin Xu ¹,
Guoshuang Zhang ⁴, and Liehuang Zhu ¹

¹School of Computer Science, Beijing Institute of Technology, Beijing 100081, China

²State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

³Data Communication Science and Technology Research Institute, Beijing 100191, China

⁴Science and Technology on Information Assurance Laboratory, Beijing 100072, China

Correspondence should be addressed to An Wang; wanganl@bit.edu.cn and Liehuang Zhu; liehuangz@bit.edu.cn

Received 2 November 2017; Revised 8 February 2018; Accepted 21 February 2018; Published 5 April 2018

Academic Editor: Amir Anees

Copyright © 2018 Qi Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Identity-based cryptographic algorithm SM9, which has become the main part of the ISO/IEC 14888-3/AMD1 standard in November 2017, employs the identities of users to generate public-private key pairs. Without the support of digital certificate, it has been applied for cloud computing, cyber-physical system, Internet of Things, and so on. In this paper, the implementation of SM9 algorithm and its Simple Power Attack (SPA) are discussed. Then, we present template attack and fault attack on SPA-resistant SM9. Our experiments have proved that if attackers try the template attack on an 8-bit microcontrol unit, the secret key can be revealed by enabling the device to execute one time. Fault attack even allows the attackers to obtain the 256-bit key of SM9 by performing the algorithm twice and analyzing the two different results. Accordingly, some countermeasures to resist the three kinds of attacks above are given.

1. Introduction

With the development of integrated circuit and communication technology, smart devices are not only widely spread in our daily life with the proliferation of Internet of things, but also extensively used in the global IT environments and critical infrastructures. Security becomes a critical issue since attacks on these devices may directly harm the consumers. Several papers [1–4] have studied related security and wireless issues.

Identity-Based Cryptography (IBC) which applies user identity as the public key was proposed by Shamir in 1984 [5] to reduce the complexity of key and certificate management. Developed by the Commercial Cryptography Administration of China in 2016, SM9 [6] has become the most typical identity-based cryptographic algorithm in China. Compared with traditional cryptographic algorithms, SM9 not only omits the exchange of digital certificates and public key processes, but also simplifies the deployment and management of

the security systems. Because of its usability and simplicity, SM9 has been employed as the standard for commercial cryptography in China. Its digital signature algorithm has become an international standard as the main part of the ISO/IEC 14888-3/AMD1 in November 2017 [7] too. It is also adopted to secure various systems and scenarios like E-mail [8], cloud storage, intelligent devices [9], industrial control, online communications, mobile payment, and so on.

As described in [10] by Kocher et al. in 1999, it has been proved that even though mathematical characteristics can guarantee the security of cryptographic algorithms in theory, their implementation may suffer from Side-Channel Attack (SCA). SCA allows attackers to reveal secrets by analyzing the side information of an attacked device which is running a cryptographic algorithm, such as power consumption, electromagnetic radiation, and execution time. Because of the low cost and high efficiency, SCA has successfully cracked lots of devices which run DES [11], AES [12], RSA [13], and ECC

[14]. Despite SM9 algorithm being secure in cryptography theory, whether it is against SCA is still a matter of concern.

At present, the three main SCA techniques are Simple Power Attack (SPA), template attack, and fault attack. Due to the versatility and operability, they have been studied in depth and used to crack various cryptographic algorithms. SPA [15] exploits one trace to reconstruct the sequence of operations during the secret computation and derive information about the secrets from this sequence. As a special power analysis, template attack [16] makes a better use of all information present in each sample. And it is hence the strongest form of SCA possible in an information theoretic sense given the few samples that are available. Fault attack, as another main branch of SCA, often injects errors into cryptographic computation processes and identifies the secret key by analyzing the mathematical and statistical properties of wrong calculation results. Proposed by Biham and Shamir in [17], the fault attack on RSA has become a milestone for the security of public key cryptographic devices.

In this paper, we show that SCA does have a practical threat to the implementation of SM9. We propose the above three kinds of SCA attacks on SM9 algorithm. After this, some corresponding countermeasures are also introduced. The main contributions of this paper are as follows.

(1) A SPA attack on SM9 algorithm is proposed. And we also introduce some countermeasures to resist SPA.

(2) Different from general Elliptic Curve Digital Signature Algorithm (ECDSA), the key is a point on elliptic curves rather than a scalar in scalar multiplication for SM9 algorithm. According to this feature, a template attack is presented for SPA-resistant SM9 implementation and several countermeasures are provided.

(3) We propose a fault attack and conduct experiments to prove that software implementation of SM9 algorithm is vulnerable to this scheme. And then, some corresponding countermeasures are also presented.

This paper is organized as follows. In Section 2, the summarization of SM9 algorithm and its implementation are introduced. We give the basic idea of SPA on SM9 in detail and put forward some countermeasures in Section 3. Then, in Section 4, a template attack is provided to attack the protected SM9 which can resist SPA and the corresponding countermeasures are also given. In Section 5, a fault attack for SPA-resistant SM9 algorithm is presented and several countermeasures are introduced against this scheme. Finally, we conclude this paper in Section 6.

2. The Preliminaries

2.1. SM9 Digital Signature Generation Algorithm. SM9 digital signature algorithm usually assumes a scenario where Alice communicates with Bob. Alice generates the signature (h, S) of message M by SM9 digital signature generation algorithm for authentication and sends them to Bob. Bob validates the received message M' and its signature (h', S') with signature verification algorithm to ensure the authenticity and integrity of this digital signature.

In order to express clearly, we give the meanings of letters as follows. The signature private key of Alice denoted as ds_A

is provided by Key Generation Center (KGC). Group G_1 and group G_2 are addition cyclic groups of order N and their generators are denoted as P_1 and P_2 , respectively. Group G_T is a multiplicative cyclic group with order N . Let e denote the bilinear pairs mapping function from $G_1 \cdot G_2$ to G_T . KGC generates a random number ks as the signature master private key and computes $P_{pub-s} = [ks]P_2$ as the master public key. P_{pub-s} is well-known, and ks is kept secretly by KGC. A cryptographic hash function is denoted as $H_2(m, N)$.

Algorithm 1 shows SM9 digital signature generation algorithm. At the beginning of the signature process, system parameters are provided as a part of inputs to make this algorithm work.

2.2. The Implementation of SM9 Digital Signature Algorithm

2.2.1. Scalar Multiplication. Scalar multiplication is the most important part in elliptic curve cryptography algorithm and its fast implementation is an inevitable demand of practical applications. As shown in Algorithm 1, scalar multiplication is directly related to the signature private key ds_A in SM9 digital signature generation algorithm.

The operation of adding a point P to itself for k times is called scalar multiplication and is denoted as $Q = kP$. For decades, many methods have been proposed to implement this operation and the most common is binary algorithm. There are two ways to implement scalar multiplication, left-to-right and right-to-left. And the former is shown in Algorithm 2 and the special point O is called the point at infinity. In the following sections, we assume that scalar multiplication is executed with the left-to-right binary algorithm.

In Algorithm 2, point doubling is executed n times. In probability, the count of 1 in a scalar integer k is close to $n/2$, so point addition is executed nearly $n/2$ times. Let A represent point addition and D represent point doubling; the operation quantity of binary method is approximately $(n/2) \cdot A + n \cdot D$.

2.2.2. Point Addition and Point Doubling. An elliptic curve is a set of points (x, y) in which $x, y \in F$. F denotes a finite field. The set of points on an elliptic curve, together with a special point O called the point at infinity, can be equipped with an Abelian group structure by addition operation. An elliptic curve over F can be expressed as the form of Weierstrass equation:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (1)$$

where $a_i \in F$. If prime number $p \neq 2$ and 3, the Weierstrass equation can be transformed to

$$y^2 = x^3 + ax + b, \quad (2)$$

with $a, b \in F$.

For prime number $p \neq 2$ and 3, let $P = (x_1, y_1) \neq O$ be a point. The inverse of P is $-P = (x_1, -y_1)$ and $Q = (x_2, y_2) \neq O$ is a second point with $Q \neq -P$. Point addition $P+Q = (x_3, y_3)$ can be calculated as

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned} \quad (3)$$

Input: system parameters, signature master public key $P_{\text{pub-s}}$, message M , signature private key ds_A .
Output: $M, (h, S)$.
(1) Compute $g = e(P_1, P_{\text{pub-s}})$ in G_T .
(2) Generate a random number $r \in [1, N - 1]$.
(3) Compute $w = g^r$ in G_T , and convert the data type of w to bit stream.
(4) Compute $h = H_2(M \parallel w, N)$.
(5) Compute $l = (r - h) \bmod N$.
(6) **if** $l == 0$ **then**
(7) **goto** (2).
(8) Compute $S = [l]ds_A$ in G_1 .
(9) Convert the data type of h and S to byte stream.
(10) **return** $M, (h, S)$

ALGORITHM 1: SM9 digital signature generation algorithm.

Input: point P , n -bit integer $k = \sum_{i=0}^{n-1} k_i 2^i$, $k_i \in \{0, 1\}$.
Output: $Q = kP$.
(1) $Q = O$.
(2) **for** $i = n - 1$ **to** 0 **do**
(3) $Q = 2Q$.
(4) **if** $k_i == 1$ **then**
(5) $Q = Q + P$.
(6) **end for**
(7) **return** Q

ALGORITHM 2: Left-to-right binary algorithm.

with

$$\lambda = \begin{cases} \frac{(y_2 - y_1)}{(x_2 - x_1)}, & P \neq Q \\ \frac{(3x_1^2 + a)}{(2y_1)}, & P = Q. \end{cases} \quad (4)$$

The operation of $P + Q$ is called point addition if $P \neq Q$. And it is called point doubling if $P = Q$. Obviously, point addition differs from point doubling in the form of formula.

2.2.3. Montgomery Modular Multiplication. Given a modulus N and two integers X and Y of size N in base b , with $\gcd(N, b) = 1$ and $r = b^{\lceil \log_b(N) \rceil}$, Montgomery modular multiplication algorithm [18] computes:

$$\text{MontMul}(X, Y, N) = X \cdot Y \cdot r^{-1} \bmod N. \quad (5)$$

Montgomery modular multiplication algorithm is shown in Algorithm 3. Its essence is to combine $X \cdot Y$ and $U \cdot N$ by traditional multiplication method. As described in line 2 to line 8, U is unknown at first. With obtaining u_i calculated by $x_i \cdot y_0$, each $u_i \cdot N$ can be calculated too. Finally, U can be derived. $\text{MontMul}(X, Y, N)$ can be computed by alternating all $x_i \cdot Y$ and $u_i \cdot N$ and adding up to A . Figure 1 illustrates the intuitive graphical representation of CIOS modular multiplication [19, 20].

2.3. Power Analysis Attack. In SPA [15], attackers directly observe power consumption for a single execution of target operation without any statistical methods. As a special power analysis, template attack [16] generally consists of the following three phases. The first is template building phase, and attackers build templates to characterize devices by executing a sequence of instructions on fixed data. Next, it allows attackers to match the templates to the power consumption traces of devices in template matching phase. Finally, attackers can do some analysis and derive secret information during offline searching phase.

Hamming weight model proposed in [10, 21] analyzes the correlation between power consumption and the register switching from one state to the other. It is generally assumed that power consumption depends on the number of bits switching from 0 to 1 or 1 to 0 within the corresponding time. For n -bit register, binary data $D = [d_{n-1}d_{n-2} \cdots d_1d_0]$ is coded as $D = \sum_{i=0}^{n-1} d_i 2^i$ with $d_i = 0$ or 1; its Hamming weight $\text{HW}(D) = \sum_{i=0}^{n-1} d_i$ is the number of bits set to 1. Considering a chip as a large set of elementary electrical components, its power consumption contains not only the state changes but also other variables' consumption, such as offsets, time dependent components, and noise. Therefore, the basic model for the data dependency can be described as $W = a\text{HW}(D) + b$, where a is a constant and b indicates the other consumption.

2.4. Fault Attack. Fault attack [17] allows attackers to disturb cryptographic devices by physical methods to make them

Input: $X = (x_{n-1} \cdots x_1 x_0)_b$, $Y = (y_{n-1} \cdots y_1 y_0)_b$, $N = (n_{n-1} \cdots n_1 n_0)_b$, $n' = -N^{-1} \bmod b$, $\gcd(N, b) = 1$, $r = b^{\lceil \log_b(N) \rceil}$.

Output: $X \cdot Y \cdot r^{-1} \bmod N$.

```

(1)  $A = (a_{n-1} \cdots a_1 a_0)_b \leftarrow 0$ .
(2) for  $i = 0$  to  $n - 1$  do
(3)    $A \leftarrow A + x_i \cdot Y$ .
(4)    $u_i \leftarrow a_0 \cdot n' \bmod b$ .
(5)    $A \leftarrow (A + u_i \cdot N)/b$ .
(6)   if  $A \geq N$  then
(7)      $A \leftarrow A - N$ .
(8) end for
(9) return  $A$ 

```

ALGORITHM 3: Montgomery modular multiplication algorithm.

FIGURE 1: CIOS modular multiplication.

run in wrong states. Due to the injected fault, the devices perform some operations in modified environment and produce incorrect results. Combining with the algorithm in the devices, some knowledge related to the secret key could be gained from the results. Because of the lower cost, simple operation, and obvious effect, fault attack has become one of the most concerned SCA techniques.

Faults in devices can be made for a variety of reasons. In general, variations in normal working conditions can be injecting faults into a system effectively. For example, changing supply voltage or clock frequency can disrupt execution process and cause the processor to skip some instructions or change its output. Exposing devices in the temperatures outside its operational range usually makes random modifications to the memory. It is also possible to inject faults more accurately by using the inherent photoelectric effects of electric circuits. Under the exposure of photons, devices can produce induced currents and disrupt normal operations. In fact, lasers can make faults more precise in terms of target area and injection time. Also, faults can be injected in packaged circuits without removing the packaging by X-rays and ion beams.

2.5. Problem Formulation. Different from the general ECDSA and SM2 algorithm, the secret key in scalar

multiplication is the point on elliptic curves rather than the scalar for SM9 algorithm. The scalar of the classical ECC and SM2 is a secret and the point is known, while the scalar and the point are both unknown in SM9. Therefore, the attack methods of the two are fundamentally different. In this paper, we focus on this issue to present template attack described in Section 4 and fault attack described in Section 5 which are only applied to SM9.

3. Simple Power Attack and Countermeasures on SM9

3.1. Simple Power Attack on SM9. In this paper, we focus the computation of scalar multiplication $S = [l]ds_A$ to apply our SPA attack which is described in line 8 of Algorithm 1.

Our SPA attack against SM9 recovers l by observing the differences in power consumption caused by the difference operations for bit 1 and bit 0. As described in Algorithm 2, it always performs a point doubling operation whether the bit is 0 or 1. And an extra point addition will be performed if the bit is 1. Because of the differences between side-channel pattern of doubling and that of addition, attackers can easily reveal l from a single power trace. As N and S are both known, ds_A can be calculated by the formula $ds_A = l^{-1}S$.

Attackers can also perform SPA attack on $w = g^r$ as shown in line 3 of Algorithm 1. The SPA attack on modular exponentiation is similar to that of scalar multiplication. Employing the different power consumption by manipulating 1 and 0 can derive the exponent r . The lengths of r , l , and N are 256 bits so that l can be obtained by the formula $l = (r - h) \bmod N$. According to the scheme described above, attackers can also restore the secret key ds_A .

3.2. Countermeasures against Simple Power Attack. Based on our SPA attack, we can draw a conclusion that r and l are equally important in the security of SM9 digital signature algorithm. It is necessary to deploy some countermeasures on r and l against SPA attack.

There are five ways [22, 23] against SPA scheme for l in SM9 digital signature algorithm. In addition, countermeasures to protect the exponent r should be implemented. We would not repeat the descriptions about the methods for r here as they are similar to that of l . We also can refer to the SPA countermeasures of RSA [13] for r against SPA attack.

In conclusion, the five ways are as follows.

(1) *Indistinguishable Point Operation Formulae.* Indistinguishable Point Operation Formulae (IPOF) try to eliminate the difference between point addition and point doubling. The usage of unified formulae for point doubling and addition is a special case of IPOF. However, even when unified formulae are in use, the implementation of the underlying arithmetic, especially the operations with conditional instructions, may still reveal the type of the point operation (addition or doubling).

(2) *Double-and-Add-Always Algorithm.* The double-and-add-always algorithm ensures that the sequence of operations during a scalar multiplication is independent of the scalar by inserting dummy point additions. However, due to the use of dummy operations, it makes the time complexity doubled and may cause safe-error fault attack.

(3) *Atomic Block Algorithm.* Instead of making the group operations indistinguishable, one can rewrite them as sequences of side-channel atomic blocks that are indistinguishable for SPA attack. If dummy atomic blocks are added, then this countermeasure may enable safe-error attack.

(4) *Montgomery Ladder Method.* Because the intermediate values are stored in registers randomly in Montgomery ladder method, the Hamming weight of secret information would not leak out to attackers.

(5) *Random Splitting l and r .* There are two different ways to split l and r . Here we take l as an example and r also should be protected by the same principle. One is to transform l to $l + kN$ where k is a random integer and do scalar multiplication by the formula:

$$\begin{aligned} S &= [l] ds_A \longrightarrow \\ S &= [l + kN] ds_A. \end{aligned} \quad (6)$$

Another is to convert l to $l + k$ and k where k is a 256-bit random value. The formula is

$$\begin{aligned} S &= [l] ds_A \longrightarrow \\ S &= [l + k] ds_A. \end{aligned} \quad (7)$$

4. Template Attack and Countermeasures on SM9

4.1. Template Attack on SM9. The template attack proposed in this paper reveals ds_A in the case that both l and r are unknown. As shown in Section 2, the first step of scalar multiplication performs a point doubling operation. And λ needs to be calculated by

$$\lambda = \frac{3x^2 + a}{2y}, \quad (8)$$

where x and y represent the x -coordinate and y -coordinate of ds_A . We focus this computation x^2 to perform our template attack. For ease of description, we use X (256-bit) to replace x in Formula (8) and x_i ($0 \leq i < 32$) denotes one byte of X .

Assume that SM9 digital signature algorithm is executed on an 8-bit microcontroller, and the power consumption of intermediate values in the calculation process of CIOS modular multiplication algorithm can be acquired. We give the letters meanings as follows. $X = (x_{31}, \dots, x_1, x_0)$ is the x -coordinate of $ds_A(X, Y)$. And $t_h(x_i x_j)$ and $t_l(x_i x_j)$ are the power consumption traces with the high 8-bit and low 8-bit of the intermediate value $x_i x_j$ ($0 \leq i < 32, 0 \leq j < 32$). The template with Hamming weight from 0 to 8 is denoted as $T_\tau = \{\tau_0, \tau_1, \dots, \tau_8\}$ and $\text{Match}(t, T_\tau)$ is a method to reflect the degree of t and T_τ . $\text{HW}_h(x_i x_j)$ and $\text{HW}_l(x_i x_j)$ denote the Hamming weight of high 8-bit and low 8-bit of $x_i x_j$, respectively.

As shown in Figure 2, there are three phases in our template attack. Firstly, in the template building phase, we focus on the operation $x_i x_j$ ($0 \leq i < 32, 0 \leq j < 32$) and calculate the Hamming weight of high 8-bit and low 8-bit of $x_i x_j$ as the target of template. We build templates of Hamming weight $T_\tau = \{\tau_0, \tau_1, \dots, \tau_8\}$ to characterize devices. Next, we match the templates to the power consumption traces of devices with the match function $\text{Match}(t, T_\tau)$ to obtain $\text{HW}_h(x_i x_j)$ and $\text{HW}_l(x_i x_j)$ in template matching phase. Finally, two searching operations are carried out during offline searching phase and the secret key ds_A can be derived by analysis.

Algorithm 4 shows our template attack. There are two searching operations used to narrow the range of candidates in offline searching phase. The first is shown in offline searching phase from the line 1 to the line 6. For each x_i ($0 \leq i < 32$), we traverse j from 0 to 255, and add j satisfying $\text{HW}_h(j^2) == \text{HW}_h(x_i x_i)$ and $\text{HW}_l(j^2) == \text{HW}_l(x_i x_i)$ to set $U_{1,i}$ where candidates of x_i are stored. The second searching is based on $U_{1,i}$, as demonstrated in line 7 to the line 14. For each element a in $U_{1,i}$ ($0 \leq i < 32$) and each element b in $U_{1,j}$ ($i < j < 32$), we calculate the high 8-bit and low 8-bit Hamming weight of ab . Selecting a and b with the conditions

FIGURE 2: The flowchart of template attack on SM9.

of $HW_h(ab) == HW_h(x_i x_j)$ and $HW_l(ab) == HW_l(x_i x_j)$ and adding them in set U_2 , U_2 consists of many pairs (i, j, a, b) and represents the correspondence between x_i and x_j . The pair (i, j, a, b) means that if $x_i == a$ so $x_j == b$. Next, the possible values of $X = (x_{31}, \dots, x_1, x_0)$ can be obtained. It is necessary to validate whether they are the points of the elliptic curve. Finally, the secret key ds_A can be recovered.

4.2. Template Attack Experiments on SM9. We present concrete experiments on side-channel traces captured from a real device. We implemented the Montgomery modular multiplication algorithm and focused on a single precision multiplication power consumption on AT89S52 8-bit microcontroller. Traces were acquired on a Lecroy WaveRunner oscilloscope with a sampling rate of 10 GS/s. In our experiments, the parameters X and Y of ds_A are shown in Table 1.

The templates of Hamming weight from 0 to 8 built in our attack are illustrated in Figures 3 and 4.

TABLE 1: Template attack experiment parameters.

Parameter	Value (hexadecimal)
X	93DE051D 62BF718F F5ED0704 487D01D6
	E1E40869 09DC3280 E8C4E481 7C66DDDD
Y	21FE8DDA 4F21E607 63106512 5C395BBC
	1C1C00CB FA602435 0C464CD7 0A3EA616

During our template attack, the power consumption of intermediate values in the calculation process of CIOS modular multiplication algorithm is acquired. Figure 5 shows the $t_h(x_0 x_0)$ (black-line) and $t_l(x_0 x_0)$ (dark-gray-line) which are the power consumption traces with the high 8-bit and low 8-bit of $x_0 x_0$. The $Match(t, T_\tau)$ applied in our attack is least square method to reflect the distance of t and T_τ . Hence, $HW_h(x_0 x_0)$ and $HW_l(x_0 x_0)$ were revealed which were equal to 6 and 4, respectively. For each i from 0 to 31, $HW_h(x_i x_i)$ and $HW_l(x_i x_i)$ can be recovered by the steps described above.

Input: $t_h(x_i x_j), t_l(x_i x_j)$ ($0 \leq i < 32, 0 \leq j < 32$).

Output: $ds_A(X, Y)$.

Template Building Phase:

- (1) Build power consumption templates $T_\tau = \{\tau_0, \tau_1, \dots, \tau_8\}$ based on $HW_h(x_i x_j)$ and $HW_l(x_i x_j)$ where i, j from 0 to 31 and x_i, x_j from 0 to 255.

Template Matching Phase:

- (1) **for** $i = 0$ to 31 **do**
- (2) **for** $j = 0$ to 31 **do**
- (3) calculate $p_{h_{\max}} = \text{Match}(t_h(x_i x_j), T_\tau)$ to recover $HW_h(x_i x_j)$.
- (4) calculate $p_{l_{\max}} = \text{Match}(t_l(x_i x_j), T_\tau)$ to recover $HW_l(x_i x_j)$.
- (5) **end for**
- (6) **end for**

Off-line Searching Phase:

- (1) **for** $i = 0$ to 31 **do**
- (2) **for** $j = 0$ to 255 **do**
- (3) **If** $HW_h(j^2) == HW_h(x_i x_i) \ \&\& \ HW_l(j^2) == HW_l(x_i x_i)$ **then**
- (4) Add j to set $U_{1,i}$ where stores candidates of x_i .
- (5) **end for**
- (6) **end for**
- (7) **for** $i = 0$ to 31 **do**
- (8) **for** $j = i + 1$ to 31 **do**
- (9) **for each** a in $U_{1,i}$ **&& each** b in $U_{1,j}$ **do**
- (10) **If** $HW_h(ab) == HW_h(x_i x_j) \ \&\& \ HW_l(ab) == HW_l(x_i x_j)$ **then**
- (11) Add the pair (i, j, a, b) to set U_2 .
- (12) **end for**
- (13) **end for**
- (14) **end for**
- (15) Search $X = (x_{31}, \dots, x_1, x_0)$ in U_2 .
- (16) **for each** X in U_2 **do**
- (17) Compute the corresponding Y and verify X and Y .
- (18) **return** $ds_A(X, Y)$

ALGORITHM 4: Template attack on SM9.

TABLE 2: The sets of candidates of x_i .

Parameter	$U_{1,i}$	The set of candidates of x_i (hexadecimal)
x_0	$U_{1,0}$	DB, DD, ED
x_1	$U_{1,1}$	DB, DD, ED
x_2	$U_{1,2}$	1D, 1F, 26, 29, 31, 32, 33, 43, 4F, 61, 66, 9D, B6
...	...	...
...	...	...
...	...	...
x_{29}	$U_{1,29}$	05, 07, 09, 0A, 0E
x_{30}	$U_{1,30}$	1E, 34, 5E, 6F, DE
x_{31}	$U_{1,31}$	2B, 37, 3B, 47, 55, 56, 67, 71, 93, AA, B9, E7

Then, the first searching phase is performed. For each i from 0 to 31, we traverse j from 0 to 255 and calculate high 8-bit and low 8-bit Hamming weight of j^2 . Selecting j with the conditions of $HW_h(j^2) == HW_h(x_i x_i)$ and $HW_l(j^2) == HW_l(x_i x_i)$ and adding j to set $U_{1,i}$, the set $U_{1,i}$ where candidates of x_i are stored is given as shown in Table 2.

And then, the second searching phase is carried out. For each element a in $U_{1,i}$ ($0 \leq i < 32$) and each element b in $U_{1,j}$ ($i < j < 32$), we calculate the high 8-bit

Hamming weight of ab . Select a and b with the conditions of $HW_h(ab) == HW_h(x_i x_j)$ and $HW_l(ab) == HW_l(x_i x_j)$ and adding them in set U_2 . U_2 consists of many pairs (i, j, a, b) and represents the correspondence between x_i and x_j . Next, the possible values of $X = (x_{31}, \dots, x_1, x_0)$ can be obtained. The result of offline searching phase is shown in Figures 6 and 7. Finally, the value of X is 93DE051D 62BF718F F5ED0704 487D01D6 E1E40869 09DC3280 E8C4E481 7C66DDDD and it is validated to be correct. And ds_A can be revealed

FIGURE 3: The templates of Hamming weight from 0 to 8.

FIGURE 4: The templates of Hamming weight from 0 to 8 (zoom in).

successfully. The experiments have proved that our attack works well. If attackers employ the template attack on an 8-bit microcontrol unit, the key often can be obtained just by analyzing algorithm procedure of one message.

4.3. Countermeasures Against Template Attack. There are four ways [23, 24] to resist the template attack presented above.

(1) *Base Point Blinding.* This method is to select a random point R and convert $S = [l]ds_A$ to $S = [l](R + ds_A) - [l]R$. The value of $[l](R + ds_A)$ is computed first and the known value $[l]R$ is subtracted at the end of scalar multiplication to ensure its correctness. The $[l]R$ and R are stored secretly in the cryptographic devices and updated at each iteration.

(2) *Random Projective Coordinates.* Let $ds'_A(X', Y', Z)$ denote the mapping value of $ds_A(X, Y)$ in the Jacobian coordinates

FIGURE 5: Template matching results of $t_h(x_0 x_0)$ and $t_l(x_0 x_0)$.

where $X = X'/Z^2$ and $Y = Y'/Z^3$. Point $ds'_A(X', Y', Z)$ is considered to be the same as point $Q(r^2 X', r^3 Y', rZ)$ where r is a random number. For different r , ds_A is not the same. The random variable r can be updated in every execution or after each doubling or addition. Attackers could not accurately gain the values of ds_A and $2ds_A$, so this method does resist the attack.

(3) *Random EC Isomorphism.* Based on the isomorphism of elliptic curves, we transform scalar multiplication algorithm to another stochastic mapping domain. A random isomorphism curve $E'(F_p)$ is generated and ds_A is mapped to ds'_A which is on $E'(F_p)$. The calculation of S is on $E'(F_p)$ rather than $E(F_p)$. Finally, we should map ds'_A to ds_A and get S . As ds'_A can not be estimated, this method can resist the template attack.

(4) *Random Field Isomorphism.* This method makes use of isomorphisms between fields. To compute $S = [l]ds_A$, it first randomly chooses a field K' to K through isomorphism ψ and then computes

$$Q = \psi^{-1}(k(\psi(P))). \quad (9)$$

It means that $Q \in E'(K')$ is used to represent $ds_A \in E(K)$ and $S' = [l]Q$ is calculated on $E'(K')$. Finally we need to transform S' to $S \in E(K)$. The value of ds_A is hidden in this method so as to resist the above attack.

Figures 8 and 9 show the templates of Hamming weight from 0 to 8 after applying some countermeasures.

5. Fault Attack and Countermeasures on SM9

5.1. Fault Attack on SM9. Base point is one of the system parameters in scalar multiplication, and it is determined by protocols. So we inject a single-bit fault on base point to recover the secret key. Two assumptions are made before

FIGURE 6: The result of offline searching phase.

the fault attack on SM9 is performed. One hypothesis is that attackers can inject a fault on $ds_A(X, Y)$ at an unknown location during the moments of error detection and signature computation. Let $ds'_A(X', Y')$ denote the fault key. Another assumption is that the injected fault causes a bit value to flip from 0 to 1.

Figure 10 illustrates the flowchart of fault attack on SM9. As an example, we performed the fault attack on $E : y^2 = x^3 + b$. The correct signature for message M is calculated by the formula $S(X_S, Y_S) = [l]ds_A(X, Y)$. Then, a single-bit fault is injected to the x -coordinate of $ds_A(X, Y)$ and the fault key is denoted as $ds'_A(X', Y')$. The wrong signature for message M is calculated by the formula $S'(X'_S, Y'_S) = [l']ds'_A(X', Y')$. Obviously, $S(X_S, Y_S)$ and $ds_A(X, Y)$ are on $E : y^2 = x^3 + b$, while $S'(X'_S, Y'_S)$ and $ds'_A(X', Y')$ are on $E' : y'^2 = x'^3 + b'$. As $S(X_S, Y_S)$ and $S'(X'_S, Y'_S)$ are both known, b and b' can be calculated. So we have

$$\begin{aligned} Y^2 &= X^3 + b, \\ Y'^2 &= X'^3 + b', \\ Y &= Y'. \end{aligned} \quad (10)$$

Then, (11) can be gained:

$$X^3 - X'^3 + b - b' = 0 \pmod{p}. \quad (11)$$

Let $\alpha = b - b'$; (11) is equal to

$$X^3 - X'^3 + \alpha = 0 \pmod{p}. \quad (12)$$

Because only one bit is different between X and X' , (12) can be expressed as

$$X^3 - (X + 2^n)^3 + \alpha = 0 \pmod{p}, \quad (13)$$

where $n \in \{0, 1, \dots, 255\}$. And n represents the location of the different bit.

Convert (13) into the following form:

$$3 \cdot 2^n \cdot X^2 + 3 \cdot 2^{2n} \cdot X + 2^{3n} - \alpha = 0 \pmod{p}. \quad (14)$$

For each n from 0 to 255, its corresponding equation like (14) can be obtained. All values of X satisfying this equation are calculated by mathematical methods and added to set U_X . If U_X is an empty set, that means the fault position n corresponding to this equation is incorrect. Otherwise, the fault position n is correct and we need to validate each $X \in U_X$ for getting the real X . It is worth noting that as long as n is correct, the equation must have solutions. Once the X is derived, the value of $ds_A(X, Y)$ can be calculated correctly.

Obviously, if the fault is injected into the y -coordinate of $ds_A(X, Y)$, the above fault attack is still performed validly. And our fault attack also has the ability to break the SPA-resistant SM9 implementation.

To sum up the above, the fault attack algorithm is shown in Algorithm 5.

Remark 1. In the above attack, we assume that the injected fault causes a bit to flip from 0 to 1. As we all know, common faults are single-bit flipping fault, single-bit constant fault, and multibits fault. In fact, no matter which fault is injected, as long as $\alpha \neq 0$, the above attack can be performed. If $\alpha \neq 0$, for single-bit constant fault, we can think of it as a single-bit flipping fault. For multibits fault, the only difference

FIGURE 7: The set U_2 after offline searching phase.

FIGURE 8: The protected templates of Hamming weight.

FIGURE 9: The protected templates of Hamming weight (zoom in).

FIGURE 10: The flowchart of fault attack on SM9.

Input: SM9 system parameters, message M .

Output: $ds_A(X, Y)$.

- (1) Calculate the correct signature value $S(X_S, Y_S) = [l]ds_A(X, Y)$.
- (2) Inject fault on $ds_A(X, Y)$ and denote as $ds'_A(X', Y')$.
- (3) Calculate the wrong signature value $S'(X'_S, Y'_S) = [l']ds'_A(X', Y')$.
- (4) Calculate $b = Y_S^2 - X_S^3$.
- (5) Calculate $b' = Y_S'^2 - X_S'^3$.
- (6) Establish equations of X and X' .
- (7) **for** $n = 0$ to 255 **do**
- (8) Compute X by $3 \cdot 2^n \cdot X^2 + 3 \cdot 2^{2n} \cdot X + 2^{3n} - \alpha = 0 \pmod{p}$.
- (9) **end for**
- (10) Calculate Y by $y^2 = x^3 + b$ and verify X and Y .
- (11) **return** $ds_A(X, Y)$

ALGORITHM 5: Fault attack on SM9.

is that multi- n should be searched. We believe that using mathematical methods to solve this issue is not a difficult task.

5.2. Fault Attack Experiments on SM9. Assume that the elliptic curve used in our experiments is $E : y^2 = x^3 + b$. The q is a prime number, and N means the order of group G_1 . The $ds_A(X, Y)$ denotes a point of G_1 , and the scalar is denoted as l . $S(X_S, Y_S)$ is computed by $S(X_S, Y_S) = [l]ds_A(X, Y)$. Let $ds'_A(X', Y')$ denote the fault point, and the wrong signature of message M is computed by the formula $S'(X'_S, Y'_S) = [l']ds'_A(X', Y')$. The experiment parameters are listed in Table 3. We can see that the fault location is equal to 1.

We conduct some experiments with the attack in Algorithm 5, and the intermediate values of SM9 digital signature algorithm are shown in Table 4. When $n = 1$, the equation has two solutions that may be the correct X . Let X_1 and X_2 denote the two solutions, and Y_{11}, Y_{12}, Y_{21} , and Y_{22} denote the corresponding Y calculated by $y^2 = x^3 + b$. Comparing with Table 3, X_1 and Y_{12} are the correct x -coordinate and y -coordinate of ds_A , so the fault attack also shows its feasibility.

TABLE 3: Fault attack experiment parameters.

Parameter	Value (hexadecimal)
q	B6400000 02A3A6F1 D603AB4F F58EC745
	21F2934B 1A7AEEDB E56F9B27 E351457D
N	B6400000 02A3A6F1 D603AB4F F58EC744
	49F2934B 18EA8BEE E56EE19C D69ECF25
X	93DE051D 62BF718F F5ED0704 487D01D6
	E1E40869 09DC3280 E8C4E481 7C66DDDD
X'	93DE051D 62BF718F F5ED0704 487D01D6
	E1E40869 09DC3280 E8C4E481 7C66DDDF
Y	21FE8DDA 4F21E607 63106512 5C395BBC
	1C1C00CB FA602435 0C464CD7 0A3EA616
l	291FE3CA C8F58AD2 DC462C8D 4D578A94
	DAFD5624 DDC28E32 8D293668 8A86CF1A

Usually, attackers can reveal the 256-bit ds_A by enabling the device to execute twice and comparing the two different results by our fault attack.

TABLE 4: The intermediate values of fault attack.

Intermediate values	Value (hexadecimal)
X_S	A5702F05 CF131530 5E2D6EB6 4B0DEB92 3DB1A0BC F0CAFF90 523AC875 4AA69820
Y_S	78559A84 4411F982 5C109F5E E3F52D72 0DD01785 392A727B B1556952 B2B013D3
b	05
X'_S	48F78AA4 9A9443DE 0DC72D8C A91BB73C 1B2D6C4F 1FB2DA61 2E9EFACA 45D87116
Y'_S	1FC550A9 CE2DCEAD 83CF500D AEB690A4 4401D3E6 11DC9F02 B3D5FE40 482C72E6
b'	66B99EE0 079E3769 9982B5B7 280386ED D0DA26F4 BB2CE030 3F07C542 841F257C
α	66B99EE0 079E3769 9982B5B7 280386ED D0DA26F4 BB2CE030 3F07C542 841F2577
n	01
X_1	93DE051D 62BF718F F5ED0704 487D01D6 E1E40869 09DC3280 E8C4E481 7C66DDDD
Y_{11}	94417225 B381C0EA 72F3463D 99556B89 05D6927F 201ACAA6 D9294E50 D9129F67
Y_{12}	21FE8DDA 4F21E607 63106512 5C395BBC C1C00CB FA602435 0C464CD7 0A3EA616
X_2	2261FAE2 9FE43561 E016A44B AD11C56E 400E8AE2 109EBC5A FCAAB6A6 66EA679E
X_{21}	9A426326 D3E2BCF1 AFA21657 5CF85BA8 B347DB94 ECF14C97 104FCE07 67C23979
Y_{22}	1BFD9CD9 2EC0EA00 266194F8 98966B9C 6EAAB7B6 2D89A244 D51FCD20 7B8F0C04

5.3. *Countermeasures against Fault Attack.* We introduce three countermeasures against the above fault attack [25–27]; they are as follows.

(1) *Point Validation.* This method verifies if a point lies on the specified curve or not. It should be performed before and after scalar multiplication. If the point ds_A or result does not belong to the original curve, no output should be given. It is an effective countermeasure against our fault attack.

(2) *Curve Integrity Check.* The curve integrity check is to detect faults on curve parameters. Before starting an SM9 algorithm the curve parameters are read from the memory and verified using an error detecting code (i.e., cyclic redundancy check) before the algorithm execution. It is an effective method to prevent the fault attack above.

(3) *Coherence Check.* A coherence check verifies the intermediate or final results with respect to a valid pattern. Randomly coding the intermediate variables such as scalar, base point, and curve parameters is a most common operation.

(4) *Combined Curve Check.* This method uses a reference curve to detect faults. It makes use of two curves: a reference curve $E := E(F_p)$ and a combined curve E' that is defined over the ring Z' . In order to compute $S = [l]ds_A$ on curve E ,

it first generates a combined point ds'_A from ds_A and $ds_A \in E(F_p)$ (with prime order). Two scalar multiplications are then performed: $S' = [l]ds'_A$ on E' and $S = [l]ds_A$ on E . If no error occurred, S and $S' \pmod{p}$ will be equal. Otherwise, the one of the results is faulty and the results should be aborted. It is also an effective countermeasure against the fault attack presented in this paper.

(5) *Security Curve Selection.* Using the NIST curves [28] with the fragile twin curves should be avoided.

6. Conclusion

In this paper, we propose SPA attack, template attack, and fault attack on SM9 algorithm. After this, some corresponding countermeasures are also introduced. We also conduct some experiments to prove the validity of these attacks. Although this paper mainly studies SCA attacks of SM9 digital signature algorithm, we have reason to believe that these schemes are equally effective for SM9 encryption algorithm.

Overall, a security SM9 digital signature algorithm implementation should pay attention to these points shown in Table 5. And we also provide the overhead to deploy these countermeasures as a reference guidance for the SM9 algorithm security implementation [27].

TABLE 5: Countermeasures and overhead. Cost estimation: negligible (<10%), low (10%–50%), and high (>50%).

Countermeasures	Target attacks	Computation overhead
Indistinguishable point operation	SPA	Low
Double-and-add-always	SPA	Low
Atomic block	SPA	Negligible
Montgomery ladder	SPA	Low
Random splitting	SPA	High
Base point blinding	Template attack	Negligible
Random projective coordinates	Template attack	Negligible
Random EC isomorphism	Template attack	Low
Random field isomorphism	Template attack	Low
Point validation	Fault attack	Negligible
Curve integrity check	Fault attack	Negligible
Coherence check	Fault attack	Low
Combined curve check	Fault attack	Low
Security curve selection	Fault attack	-

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by Beijing Natural Science Foundation (no. 4162053), the National Cryptography Development Fund (no. MMJJ20170201), the Foundation of Science and Technology on Information Assurance Laboratory, and Beijing Institute of Technology Research Fund Program for Young Scholars.

References

- [1] X. Hei, X. Du, J. Wu, and F. Hu, "Defending resource depletion attacks on implantable medical devices," in *Proceedings of the of IEEE GLOBECOM*, MIA, FL, USA, 2010.
- [2] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen, *An Effective Key Management Scheme for Heterogeneous Sensor Networks*, *Ad Hoc Networks*, vol. 5, Elsevier, 2007.
- [3] X. J. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "Transactions papers a routing-driven Elliptic Curve Cryptography based key management scheme for Heterogeneous Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1223–1229, 2009.
- [4] X. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "Secure and efficient time synchronization in heterogeneous sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2387–2394, 2008.
- [5] A. Shamir, *Identity-based cryptosystems and signature schemes Crypto*, vol. 84, 1984.
- [6] F. Yuan and Z. Cheng, "Overview on SM9 Identity-Based Cryptographic Algorithm," *Journal of Information Security Research*, 2016.
- [7] ISO/IEC 14888-3:2016 DAmD 1, *Information technology - Security techniques - Digital signatures with appendix*, Part 3: Discrete logarithm based mechanisms, <https://www.iso.org/standard/70631.html>.
- [8] Z. Cheng, *E-mail security system*, 2016, <https://www.olytech.net/products/application/mail-client.html>.
- [9] M. Zhang, X. Du, and K. Nygard, "Improving Coverage Performance in Sensor Networks by Using Mobile Sensors," in *Proceedings of the IEEE MILCOM*, Atlantic City, NJ, USA, 2005.
- [10] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology*, pp. 388–397, Springer, 1999.
- [11] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard Test Conference," in *Proceedings of the ITC International IEEE*, pp. 339–344, USA, October 2004.
- [12] E. Oswald, *A side-channel analysis resistant description of the AES S-box International Workshop on Fast Software Encryption*, Sprinige, Berlin, Heidelberg, Germany, 2005.
- [13] Y. Yarom and K. Falkner, *FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack USENIX Security Symposium*, 2014.
- [14] T. Izu and T. Takagi, *A fast parallel elliptic curve multiplication resistant against side channel attacks Public Key Cryptography*, 2002.
- [15] S. Mangard, *A simple power-analysis (SPA) attack on implementations of the AES key expansion International Conference on Information Security and Cryptology*, vol. 2587 of Springer, Berlin, Heidelberg, Germany, 2003.
- [16] S. Chari, J. R. Rao, and P. Rohatgi, *Template attacks International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 2003.
- [17] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology CRYPTO'97*, pp. 513–525, 1997.
- [18] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, 1985.
- [19] Ç. K. Koç, T. Acar, and B. S. Kaliski, "Analyzing and comparing montgomery multiplication algorithms," *IEEE Micro*, vol. 16, no. 3, pp. 26–33, 1996.
- [20] M. McLoone, C. McIvor, and J. V. McCanny, "Coarsely integrated operand scanning (CIOS) architecture for high-speed Montgomery modular multiplication Field-Programmable Technology," in *Proceedings of the IEEE International Conference*, pp. 185–191, December 2004.

- [21] E. Brier, C. Clavier, and F. Olivier, *Correlation Power Analysis with a Leakage Model International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 2004.
- [22] L. Goubin and J. Patarin, *DES and Differential Power Analysis the Duplication Method Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 1999.
- [23] P.-Y. Liardet and N. P. Smart, *Preventing SPA/DPA in ECC systems using the Jacobi form International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, Germany, 2001.
- [24] J. Cheol Ha and S. Jae Moon, *Randomized Signed-Scalar Multiplication of ECC to Resist Power Attacks CHES*, 2003.
- [25] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, *An Efficient Countermeasure against Correlation Power-Analysis Attacks with Randomized Montgomery Operations for DF-ECC Processor CHES*, 2012.
- [26] A. Barengi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.
- [27] J. Fan and I. Verbauwhede, "An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost," in *Cryptography and security*, vol. 6805, pp. 265–282, 2012.
- [28] M. Brown, D. Hankerson, J. Lpez, and A. Menezes, "Software implementation of the NIST elliptic curves over prime fields," *Topics in Cryptology/CT-RSA*, vol. 2001, pp. 250–265, 2001.