Fault Attack of SMS4 Based on Internal Collisions

Renjie Zhu, Qingyu Ou and Fang Luo*

Department of Information Security, Naval University of Engineering, Wuhan, Hubei, 430000, China (Science and Technology on Communication Security Laboratory)

Corresponding author’s e-mail: ouqingyv@163.com

Abstract. A fault attack of SMS4 based on internal collision is proposed. The method assumes that the attacker can attack under the condition of selecting plaintext, and adopts the byte-oriented random fault model to successfully recover the original key by injecting the fault in the first few rounds of the encryption algorithm of the SMS4. Theoretical analysis and experimental results show that the 60 fault injections can successfully recover the original key of the SMS4, with a probability of over 99%. If the attacker can control the location of fault injection, then only 16 fault injections are needed to successfully recover the original key of the SMS4. In general, 30 to 40 fault injections can successfully restore the original key.

1. Introduction

A fault attack, also known as fault analysis, refers that an attacker uses a physical method (voltage glitch, electromagnetic radiation, or laser, etc.) to interfere with the operation of a software or cryptographic chip, causing it to perform some erroneous encryption, thereby revealing key information. D. Boneh first attacked the public key cryptosystem with random hardware faults, and then the fault attack occupied an important position in the crypt-analysis. In 1997, E. Biham and A. Shamir proposed a differential fault attack for DES algorithm [1]. Then, the fault attack of the cryptographic algorithm is classified from the bits, time and location of the fault, and the efficiency, performance, and difficulty of attack are studied thus obtaining cryptographic algorithms for fault attack including AES [2-5], SMS4 [6-11]. Later in the development process, many other methods of fault attack are derived. The security threat of fault attack is a long-term problem faced by the cryptographic field [12-15].

In order to promote and apply WLAN Security Standard of China, SMS4 block cipher was publicly released in 2006. the security of the cryptography determines the security of secret information. So far, many researchers at home and abroad have analyzed the security of the SMS4. Literature [6] proposes a differential fault attack (DFA) method for the SMS4 algorithm, which injects random byte faults in the 32nd round of the algorithm, and 8 times of it can successfully recover the subkey of the 32nd round. On the basis of the above, literature [7] proposes to inject the same fault in the 29th round. Under the same fault injection difficulty condition, one injection can recover the sub-key of the 32nd round on average. Literature [8] further expands the fault model and injects a random four-byte fault in the 29th round to recover the last round of subkey of the algorithm. In the fault attack method of literature [9], a random single-byte fault is injected into the 28th round of the algorithm, which expands the scope of fault injection, but also increases the computational complexity of the search key. Literature [10] puts forward that the fault attack based on the key expansion algorithm extends the attack scope of fault injection to the key expansion algorithm.
After deep analysis of the characteristics of SMS4 and the proposed fault attacks, this paper advances a fault attack method based on internal collisions of SMS4 algorithm. Using the byte-oriented random fault model combined with the differential analysis method, the 128-bit initial key of the algorithm can be completely recovered by performing fault injection multiple times in the first few rounds of the encryption algorithm.

2. SMS4 algorithm
The SMS4 is an iterative block cipher consisting of an encryption algorithm and a key expansion algorithm. The SMS4 block cipher uses an unbalanced Feistel structure with a packet length of 128 bits and a key length of 128 bits. Both the encryption algorithm and the key expansion algorithm use a 32 rounds nonlinear iterative structure. The encryption operation has the same structure as the decryption operation, and the order of use of the round key of the decryption operation is opposite to that of the encryption operation.

2.1 Encryption algorithm
The SMS4 encryption algorithm consists of 32 iterations and a reverse transformation \( R \).

Set the plaintext input as \((X_0, X_1, X_2, X_3) \in (Z_2^{128})^4\), the ciphertext output is \((Y_0, Y_1, Y_2, Y_3) \in (Z_2^{128})^4\), and the round key is \( r_{k_i} \in (Z_2^{128})^4 \), \((i = 0, 1, \cdots, 31)\). The operation process of the encryption algorithm is as follows.

First, for \( i = 0 \) to 31, perform 32 iterative operations:
\[
X_{i+4} = X_i \bigoplus T(X_{i+1} \bigoplus X_{i+2} \bigoplus X_{i+3} \bigoplus r_{k_i}).
\]

Second, proceed reverse transformation to the final round output data and get the ciphertext output:
\[
(Y_0, Y_1, Y_2, Y_3) = R(X_{32}, X_{33}, X_{34}, X_{35}) = (X_{35}, X_{34}, X_{33}, X_{32}).
\]

Where \( T : Z_2^{128} \rightarrow Z_2^{128} \) is a reversible transformation, which is composed of a nonlinear transformation \( S \) and a linear transformation \( L \), that is, \( T = L \circ S \). The nonlinear transformation \( S \) consists of 4 parallel \( S \) boxes, and set the input as \( A = (a_0, a_1, a_2, a_3) \in (Z_2^{128})^4 \), then the output is \( B = (Sbox(a_0), Sbox(a_1), Sbox(a_2), Sbox(a_3)) \). \( L \) is a linear transformation, and the output of the nonlinear transformation \( S \) is the input of the linear transformation. Set input to \( B \in Z_2^{128} \), and the output is \( C \in Z_2^{128} \), then: \( C = L(B) = B \bigoplus (B \ll 2) \bigoplus (B \ll 10) \bigoplus (B \ll 18) \bigoplus (B \ll 24) \)

Its encryption algorithm is shown in Figure 1:

![Figure 1 SMS4 algorithm encryption process](image-url)
2.2 Decryption algorithm

The decryption transform has the same structure as the encryption transform, and the only difference is the order in which the round keys are used. The order of the round key when decrypting is opposite to the order of encryption.

3. Fault attacks based on internal collisions

3.1 Fault model and basic assumptions

In this paper, the fault model adopted is a byte-oriented model, and the basic assumption is:

1) The attacker can induce a single-byte fault in the input data in the first few rounds of the algorithm encryption process, without knowing the specific location and value of the fault byte.
2) The attacker can choose plaintext and encrypt different plaintexts with the same key.

3.2 Basic steps

1) Select a known plaintext and perform fault induction in the second round of its encryption to obtain the error ciphertext. The plaintext is searched so that the correct encrypted ciphertext of the plaintext is the same as the error ciphertext obtained in this step. Using differential analysis, the partial byte information of the subkey of the first round is recovered. This process is repeated until the first round of subkeys is fully recovered.
2) Perform fault induction on the third round of known plaintext encryption to obtain the error ciphertext. The first round is decrypted with the subkey of the first round that is recovered in step 1). Search for the second round of input values, so that in the subsequent correct encryption, the output ciphertext is the same as the error ciphertext obtained in this step. Partial byte information of the second round of subkeys is recovered with differential analysis. This process is repeated until the second round of subkeys is fully recovered.
3) Similarly, random fault induction is performed with the same method as above, and the third and fourth rounds of the algorithm are attacked in turn, and the subkeys of these rounds are recovered.
4) Substituting the first four rounds of subkeys that have been recovered into the key expansion algorithm, and calculating the subkeys of each round and the values of the original encryption key.

4. Detailed description of fault attacks based on internal collisions

4.1 Basic signs and symbols

The signs and symbols are as follows, where \( 0 \leq i \leq 31 \). Let \((x_{i-1}, x_i, x_{i+1}, x_{i+2}) \in (\mathbb{Z}_2^{32})^4\) be the input of the \( i \)-th round and \((x_i, x_{i+1}, x_{i+2}, x_{i+3}) \in (\mathbb{Z}_2^{32})^4\) be the output of the \( i \)-th round.

\[ A_i = (x_i \oplus x_{i+1} \oplus x_{i+2}) \in \mathbb{Z}_2^{32} \] is the plaintext \( X \) in the encryption, and the exclusive OR values of \( x_i \), \( x_{i+1} \), and \( x_{i+2} \) are input in the \( i \)-th round. \[ A_i' = (x_i' \oplus x_{i+1}' \oplus x_{i+2}') \in \mathbb{Z}_2^{32} \] is the plaintext \( X' \) in the encryption, and the exclusive OR values of \( x_i' \), \( x_{i+1}' \), and \( x_{i+2}' \) are input in the \( i \)-th round. \( C' \) indicates the error ciphertext after the fault injection of the plaintext \( X \). C indicates the ciphertext after the encryption of the plaintext \( X' \). \( \Delta x_i = x_i \oplus x_i' \) represents the difference between the intermediate states of the plaintext \( X \) and \( X' \) encryptions.

4.2 Detailed description of the attack process

In this section, the attack process is described in detail.

1) The first round of attacks, the steps are as follows:
   ① Encrypt the plaintext \( X \) under the initial key \( MK \). Before the second round of the encryption, a single-byte fault is randomly injected into its storage unit \( x_1 \), \( x_2 \) or \( x_4 \), thereby obtaining the error ciphertext \( C' \).
② Search the plaintext and find the plaintext \( X' \) that meets the requirement (\( C \) is the encryption result of \( X' \) under the \( MK \), \( C = C' \)).

Eligible plaintext for \( (X, X') \) must satisfy one of the \((x_1, x_2, x_3, x_4) = (x_1 \oplus \varepsilon, x_2, x_3, x_4)\), \((x_1, x_2, x_3, x_4) = (x_1, x_2 \oplus \varepsilon, x_3, x_4)\) and \((x_1, x_2, x_3, x_4) = (x_1, x_2, x_3 \oplus \varepsilon, x_4)\). The three equations respectively correspond to the fault injection of into storage units \( x_1 \), \( x_2 \) and \( x_3 \). Thus,
\[
C = \bigoplus_{i=2}^{32} T_{r_{k_i}} (A_i \oplus \varepsilon, x_i) = C'
\]  

Therefore, in the case where the error vector \( \varepsilon \) and the fault injection storage unit are unknown, in order to find the plaintext \( X' \) such that \((X, X')\) is set to be the correct eligible plaintext, the attacker encrypts \( X' = X \oplus \delta_\varepsilon = (x_1 \oplus \varepsilon, A_i \oplus \varepsilon) \). Then, the plaintext \( X' \) with the ciphertext equal to \( C' \) is found. When the fault is injected to the storage unit \( x_1 \), the differential path is shown in Figure 2.

It is similar when a fault is injected to \( x_2 \) or \( x_3 \).

![Figure 2 Differential path for fault is injected to \( x_i \)](image)

③ Determine value \( \delta_\varepsilon \).

First determine the value of \( \varepsilon \). \( \varepsilon \) is a single-byte fault at an unknown location, so the value of \( \varepsilon \) may be \((0,0,0,0), (0,0,\theta,0), (0,\theta,0,0)\) or \((\theta,0,0,0)\), all of which are 32-bit data. There are 255 possible values for \( \theta \), so the possible value of \( \varepsilon \) is \( 4 \times 255 \). Since the fault may be injected to \( x_1 \), \( x_2 \) or \( x_3 \), the \((x_1, x_2, x_3)\) in the plaintext \( X' \) has possible values of \( 3 \times 4 \times 255 \).

Next, determine the value of \( \hat{\varepsilon} \). In the first round of encryption, the nonlinear conversion \( S \) is input as \( A_i \oplus A_i = \varepsilon \), and only one \( S \) box becomes active. The possible values of \( \varepsilon \) are divided into 4 cases. For each case of \( \varepsilon \), the difference of the output of the nonlinear conversion \( S \) has 255 possible values. After the linear conversion \( L \), there are still 255 possible values. Therefore, \( a \) has a total of \( 4 \times 255 \) possible values.

The differential characteristic of the \( S \) box in the SMS4 satisfies that there is a total of 127 possible non-zero output differences for any non-zero input differential. By taking 255 possible input differences into the \( S \) box for calculation, it is enabled to get 127 possible output differential values for each input difference. Therefore, there are \( 3 \times 4 \times 255 \times 127 \) possible values of \( \delta_\varepsilon \), which is approximately equal to \( 2^{18.56} \).

④ Restore some bytes of the first round of subkeys. After obtaining the eligible plaintext pair \((X, X')\), by calculating the difference between \( X \) and \( X' \), value \( \varepsilon \) can be obtained to determine the position of the active \( S \)-box. The correct plaintext pair satisfies equation (2).
\[
S(A_i \oplus r_{k_0}) \oplus S(A_i \oplus r_{k_0}) = L^{-1}_i(x_0 \oplus x_0)
\]  

Wherein, the value of \( i \) corresponds to the position of the active \( S \) box. \( A_i \) and \( A_i \) are 8-bit data XOR values of the positions respectively corresponding to the plaintext \( X \) and \( X' \). \( r_{k_0} \) is an 8-bit key corresponding to the position of the sub-key \( r_{k_0} \).

A fault injection can recover the 8-bit data of the subkey in the first round.
In the second round of the plaintext encryption, a random single-byte fault is injection multiple times. According to the above steps, part of the subkey of $rk_0$ is restored. In this way, after a plurality of fault injections, the complete key $rk_0$ in the first round can be restored.

It should be noted that different faults may make the same S box turn active. After injecting the fault and searching for the plaintext pair, if a part of the subkeys corresponding to the location of the active S box has been obtained, the key need not be filtered.

2) The second round of attacks, the steps are as follows:

1. After $rk_0$ is determined, the first round of encryption is performed on the plaintext $X$ to obtain the output value $(x_1, x_2, x_3, x_4)$ of the first round.

2. The original plaintext is encrypted by the key $MK$. Before the third round of encryption, a single-byte fault is randomly injected into its storage unit $x_1$, $x_2$, $x_3$, or $x_4$ to get the error ciphertext.

3. Use a completely similar approach to search for the eligible correct plaintext. The encrypted output of the correct plaintext in the second round conforms to one of the $(x_2, x_3, x_1, x_4) = (x_2 \oplus \varepsilon, x_3, x_1, x_4)$ and $(x_2, x_3, x_1, x_4) = (x_2, x_3, x_1, x_4 \oplus \varepsilon, x_4)$. Thus,

$$C' = \bigoplus_{i=3}^{12} T_{rk_1} (A_2 \oplus \varepsilon, x_i) = C'' \quad (3)$$

4. Restore the partial byte information of the key $rk_i$ in the second round with a completely similar method in step 2).

Repeat the above process until $rk_i$ is fully recovered.

3) Recover the $rk_2$ and the $rk_3$ in a similar way.

4) The subkey is derived from the above with the key expansion algorithm, and the initial key $MK$ can be recovered.

4.3 Complexity analysis

The sub-key recovery process in the first round is taken as an example to analyze the complexity of the attack process.

In the preparation phase of the attack, it is necessary to calculate the possible values of the output differences corresponding to all input differences of the S box. It is calculated that in a matrix of $255 \times 127$. When the $127$ data of the $i$-th row indicating that the input difference is $i$ in the S box, there are $127$ possible output differences. The complexity is $256 \times 255 \approx 2^{16}$.

As can be seen from the detailed description of the attack process, the computational complexity required to search for a correct pair of plaintexts is approximately $2^{18.56}$, and the computational complexity of recovering 8-bit data of the subkey is $2^8$. Fault injection causes one of the four S boxes to turn active. Since the fault injection is random, the probability of the four S boxes turning active is equal. During the attack, the probability of 4 fault injections turning the four S-boxes active is about 93.8%. 15 fault injections can make the probability of successfully turning all the 4 S boxes active exceed more than 99%.

Based on the above analysis, the average computational complexity of restoring the subkey in the first round does not exceed $15 \times 2^{18.56} + 4 \times 2^8 \approx 2^{22.48}$, and the complexity of recovering 4 rounds of keys does not exceed $2^{24.48}$.

In addition, if the attacker can accurately control the location of the single-byte fault injection, the computational complexity will be reduced to $255 \times 127$ in the search plaintext of the attack process. At the same time, the subkey can be successfully restored only with 4 rounds of fault injections with the complexity of $4 \times 255 \times 127 + 4 \times 2^8 \approx 2^{16.99}$. The complexity of recovering 4 rounds of keys is $2^{18.99}$. 

5
5. Attack experiments and results comparison

5.1 Attack Experiment

The attack method proposed in this paper is implemented by MATLAB on a common PC. The fault injection during the attack is simulated. A total of 7 experiments are performed and the results are shown in Table 1. In the first 5 experiments, the attacker cannot control the position of the single-byte fault injection, while the attacker can locate the position of the single-byte fault injection in the latter 2 experiments.

| No. of the experiment | Number of fault injections required to attack the four-round subkey |
|-----------------------|---------------------------------------------------------------|
|                       | first round | second round | third round | fourth round | total    |
| 1                     | 11          | 6            | 5           | 9            | 31       |
| 2                     | 14          | 7            | 10          | 6            | 37       |
| 3                     | 12          | 7            | 9           | 11           | 39       |
| 4                     | 13          | 14           | 9           | 5            | 41       |
| 5                     | 5           | 7            | 12          | 7            | 31       |
| 6                     | 4           | 4            | 4           | 4            | 16       |
| 7                     | 4           | 4            | 4           | 4            | 16       |

5.2 Results Comparison

Compared with the previous methods, the fault attack method based on internal collisions of SMS4 algorithm has the following two features.

1) The premise of the fault is different. The attacker selects a plaintext attack and can encrypt several selected plaintexts with the same key. In addition, an attacker can inject a random single-byte fault into the first few rounds of encryption in the algorithm.

2) The proposed attack method effectively expands the scope of the original fault attack. In the existing fault attack methods, the attacker performs fault injection in the last rounds of the encryption algorithm or the last few rounds of the key expansion algorithm, and finally restores the initial key by recovering the last 4 rounds. The fault attack proposed in this paper can realize the fault injection in the first few rounds of the encryption algorithm, and obtain the initial key of the algorithm by restoring the first 4 rounds.

6. Conclusion

This paper proposes a fault attack method based on internal collisions of SMS4 algorithm. The principle, method and steps of the attack are introduced in detail, and the complexity of the attack is analyzed. Theoretical analysis and experimental results demonstrate that 16 attack injection can successfully recover the initial key of the SMS4 when the attacker can control the fault injection location. When the attacker cannot locate the fault injection, the probability of 60 fault injections successfully recovering the initial key of the SMS4 algorithm exceeds 99%.

Compared with the attack method proposed by the predecessors, this method requires more fault injection times. Nevertheless, it expands the attack range of fault injection and improves the usability of fault attacks.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable suggestions and comments. The work in this paper is supported by Foundation Item: Foundation of Key Laboratory of Defense Science and Technology for Secret Communication (No.614210301030117).
References

[1] Biham, E., Shamir, A. (1997) Differential fault analysis of secret key cryptosystems. In: Annual international cryptology conference. Berlin, pp. 513-525.

[2] Roche, T., Lomné, V., Khalfallah, K. (2011) Combined fault and side-channel attack on protected implementations of aes. In: International Conference on Smart Card Research and Advanced Applications. Berlin, pp. 65-83.

[3] Fuhr, T., Jaulmes, E., Lomné, V., Thillard, A. (2013) Fault attacks on AES with faulty ciphertexts only. In: 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography. IEEE. pp. 108-118.

[4] Ghalaty, N. F., Yuce, B., Taha, M., Schaumont, P. (2014) Differential fault intensity analysis. In: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography. IEEE. pp. 49-58.

[5] Lashermes, R., Reymond, G., Dutertre, J. M., Fournier, J., Robisson, B., Tria, A. (2012) A DFA on AES based on the entropy of error distributions. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography. IEEE. pp. 34-43.

[6] Zhang, L., Wu, W. L. (2006) Differential fault analysis on SMS4. CHINESE JOURNAL OF COMPUTERS. CHINESE EDITION., 29(9): 1596.

[7] Li, W., Gu, D. W. (2007) An improved method of differential fault analysis on the SMS4 cryptosystem. In: The First International Symposium on Data, Privacy, and E-Commerce (ISDPE 2007). IEEE. pp. 175-180.

[8] Song, X. F., Wu, Z., Wang, M., Du, Z. B., Rao, J. T. (2016) Differential Fault Attack Method on SM4 Based on Random Fault Injection. CHINESE JOURNAL OF COMPUTERS. CHINESE EDITION., 42(7): 129-133.

[9] Li, R., Sun, B., Li, C., You, J. (2011) Differential fault analysis on SMS4 using a single fault. Information Processing Letters., 111(4): 156-163.

[10] Li, W., Gu, D. W. (2008) Differential fault analysis on the SMS4 cipher byinducing faults to the key schedule. Journal on Communications., 29(10): 135-142.

[11] Lu, J. (2007) Attacking reduced-round versions of the SMS4 block cipher in the Chinese WAPI standard. In: International Conference on Information and Communications Security. Berlin. pp. 306-318.

[12] Ren, Y., Wang, A., Wu, L. (2015). Transient-steady effect attack on block ciphers. In: International Workshop on Cryptographic Hardware and Embedded Systems. Berlin. pp. 433-450.

[13] Briais, S., Cioranescu, J. M., Danger, J. L., Guilley, S., Naccache, D., Porteboeuf, T. (2012). Random active shield. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography. IEEE. pp. 103-113.

[14] Patranabis, S., Chakraborty, A., Mukhopadhyay, D. (2017) Fault tolerant infective countermeasure for AES. Journal of Hardware and Systems Security., 1(1): 3-17.

[15] Luo, P., Fei, Y. (2016) Faulty Clock Detection for Crypto Circuits Against Differential Fault Analysis Attack.IACR Cryptology ePrint Archive., 2016: 967.

[16] Liu, F., Ji, W., Hu, L., Ding, J., Lv, S., Pyshkin, A., Weinmann, R. P. (2007). Analysis of the SMS4 block cipher. In: Australasian Conference on Information Security and Privacy. Berlin. pp. 158-170.