Further Improved Differential Fault Analysis on Camellia by Exploring Fault Width and Depth

Xin-jie Zhao, Tao Wang

Abstract—In this paper, we present two further improved differential fault analysis methods on Camellia by exploring fault width and depth. Our first method broadens the fault width of previous Camellia attacks, injects multiple byte faults into the r^{th} round left register to recover multiple bytes of the r^{th} round equivalent key, and obtains Camellia-128,192/256 key with at least 8 and 12 faulty ciphertexts respectively; our second method extends fault depth of previous Camellia attacks, injects one byte fault into the r-2^{th} round left register to recover full 8 bytes of the r^{th} round equivalent key, 5-6 bytes of the r-1^{th} round equivalent key, 1 byte of the r-2^{th} round equivalent key, and obtains Camellia-128,192/256 key with 4 and 6 faulty ciphertexts respectively. Simulation experiments demonstrate: due to its reversible permutation function, Camellia is vulnerable to multiple bytes fault attack, the attack efficiency is increased with fault width, this feature greatly improves fault attack's practicalities; and due to its Feistel structure, Camellia is also vulnerable to deep single byte fault attack, 4 and 6 faulty ciphertexts are enough to reduce Camellia-128 and Camellia-192/256 key hypotheses to $2^{22.2}$ and $2^{31.8}$ respectively.

Index Terms—Differential fault analysis, Feistel structure, SPN structure, Camellia, Block cipher, Fault width and depth

1 Introduction

The idea of fault attack was first suggested Boneh et al. in 1997 [1]. Boneh et al. showed that this attack succeeded when it was applied to RSA based on the Chinese Remainder Theorem using a faulty ciphertext. Shortly after, Biham and Shamir proposed that the idea of fault analysis could be applied to symmetric-key cryptography DES and showed that the attack succeeded in obtaining an entire DES key[2]. They called this attack differential fault analysis (DFA), which is executed with some pairs of correct and faulty ciphertexts. Since that, many research papers have been published using this cryptanalysis technique to successfully attack various cryptosystems, including ECC[3], 3DES[4], AES[5][6][7][8][9][10][11], Camellia[12][13][14], ARIA[15], CLEFIA[16][17], RC4[18][19], Trivium[20][21] and so on.

Camellia is a 128-bit block cipher jointly developed by NTT and Mitsubishi Electric Corporation in 2000[22]. It was chosen as a recommended algorithm by the NESSIE project in 2003 and certified as the IETF standard cipher for XML security URIs, SSL/TLS cipher suites and IPsec in 2005. In March 2009, Camellia was integrated into the OPENSSL-1.0.0-beta1. In Jan 2009, Zhou et. al. proposed the first DFA on Camellia using approximately 64 pairs of correct and faulty ciphertexts to recover a 128-bit key and 96 pairs to recover 192/256-bit keys[12]. After one byte random fault injected into r^{th},r-1^{th},r-2^{th},r-3^{th} etc round register, they simply extended DFA attack against DES in [2], many faulty pairs were needed to obtain Camellia key. In Dec 2009, Zhao et. al. proposed an improved fault attack extending the fault depth of [12], and used 16 pairs of correct and faulty ciphertexts to recover a 128-bit key and 24 pairs to recover 192/256-bit keys[13]. They injected one byte fault into the r-1^{th},r-2^{th},r-3^{th} etc rounds, utilized the revese feature of Camellia permutation function to recover 5-6 key byte at one time, and improved the key recovery efficiency. At the same time, Li et. al. also proposed the same method as [13] in [14], used 30 pairs to recover Camellia 128/192/256-bit keys.

In this paper, we analyze the basic DFA attack principle and summarize the DFA on block ciphers with S-box into computing the S-box input and output differential problems, then present two further improved DFA methods on Camellia. Note that how to induce the specific fault is not covered in this paper, since this is not the main concern of our paper and many literatures on fault inductions are available [23].

Our first attack is based on the multiple byte faults model, which is verified by hardware fault attack experiments in [10] and [11] recently. In these two experiments, both SPN and Feistel structure block cipher can be injected fault with any width from 8-bit to 64-bit, sometimes even 128-bit. In this paper, we inject multiple byte faults into the r^{th},r-1^{th},r-2^{th},r-3^{th} etc round register instead of one byte in [12] and provide the complexity analysis of the attack. The proposed attack requires 16 pairs and 24 pairs to obtain 128-bit, 192/256 key with feasible calculation time and improve the fault analysis efficiency, attack in [12] is a specialized case of our attack when the faulty byte number is one, most importantly, multiple byte faults attack is more practical than single byte fault attack.

Our second attack further extends the fault depth of [13] and [14] from the r-1^{th} round to the r-2^{th} round. We inject multiple byte faults into the r-2^{th},r-4^{th} etc round left register instead of the r-1^{th},r-2^{th},r-3^{th} etc round left register in [13]. The proposed attack takes advantage of the generalized Feistel structure of Camellia, one byte fault in the r-2th round can recover 8 bytes of the rth round equivalent key, 5-6 bytes of the r-1th round equivalent key and 1 byte of the r-2th round equivalent key, so 4 and 6 faulty pairs are enough to recover Camellia-128,192/256 key hypotheses to $2^{22.2}$ and $2^{31.8}$ respectively. Compared with [12],[13],[14], Our DFA method not only enhances the fault depth, but also improves the fault analysis efficiency by 16 times and 4 times respectively, and decreases the faulty ciphertexts number. Besides, our second attack can be easily extended to DFA on Camellia key schedule case, while [12] can not.

As induction of faults requires high precision instruments (more the precision, more the cost!) and is harder to guarantee, an attack which requires large number of faulty pairs is impractical. We can, therefore, state that the most efficient attacks are those that require the fewest assumptions on the effect of a fault and least faulty ciphertexts. Our first attack looses the assumptions on the fault width to improve the fault injection practicability, the second attack deduces the number of faulty ciphertexts to improve the key retrieve efficiency, so we believe that both attacks are much more strongger than previous DFA attacks on Camellia.

This work is organized as follows. In Section 2, we present the basic DFA model and how it can be used into SPN and Feistel block ciphers. Section 3 presents the general overview of DFA on Camellia. Section 4 and Section 5 present several further improved DFA attacks on Camellia by broadening fault width and enhancing fault depth respectively. Section 6 displays the complexity analysis and experimental results of the attacks. Section 7 is the conclusion.

2 DFA ATTACK MODEL

Most block ciphers are composed of Substitution function S and Permutation function P. In DFA attacks, the adversary usually injects single byte fault before the final S function, after the S function, the S-box lookup index byte a becomes a^*, the S-box input differential value Δa ($\Delta a = a \oplus a^*$) can be either known or unknown. But usually, the adversary can obtain the S-box output differential Δc by observing the correct and faulty ciphertext. So, it always holds the following formula

$$S[a] \oplus S[a \oplus \Delta a] = \Delta c \qquad (1)$$

The output of the S function usually has extra post-whitenings by Xored the last round key to generate the ciphertexts C. As C is known, if a is obtained, the last round key can be recovered. According to Δa is known or unknown, we present two DFA models for Feistel and SPN structure block ciphers.

1 Δa is known

This case is usually related with Feistel structure block cipher. If one byte fault Δa is injected into the rth round left register, due to the feature of Feistel structure, both Δa and Δc can be obtained after analyzing the cipher differential ΔC. If we input every possible candidate of a to formula (1), we can get limited candidates of a satisfying formula (1).

Fig. 1 is the Camellia S-box and differential S-box (Δ=1) elements sorted ascending, the gray block denotes candidates of S-box, and the white block denotes the impossible candidates of S-box. It's clear to see that the Camellia S-box S has covered with every distinct value from 0x00 to 0xff (total number is 256), and every candidate is used only once. However, when it comes to the differential S-box S' ($S'[a]=S[a] \oplus S[i \oplus \Delta a]$, Δa=0x01), S' can't cover every distinct candidate value from 0x00 to 0xff(total number is 127), usually every possible candidate of S' is used twice or more. If we input every candidate of a into formula (1), 2-4 candidates of a can be obtained, which means that we can get 2-4 candidates for the last round key.

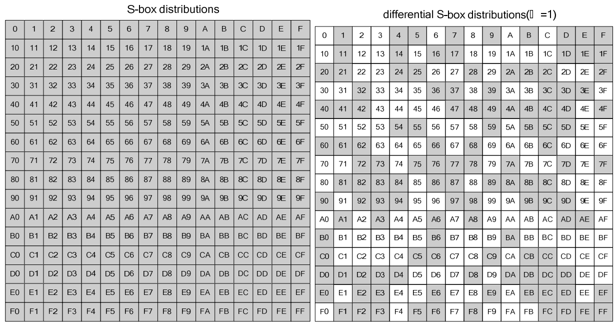

Fig. 1. Camellia S box and differential S box (Δ=1) elements sorted ascending.

2 Δa is unknown

This case is usually related with SPN structure block cipher. As to Feistel structure block cipher, when one byte fault is injected before the last permutation layer of the r-1th round, and both the rth round S-box input and output differential are known. However, as to SPN structure block cipher, when one byte fault is injected before the last permutation layer of the r-1th round, after the permutation layer, m faulty state with the same differential fault value as Δa can be generated, but after the rth S function, the faults are propagated into m differential faults, ususlly, the S-box output differential Δc is known, but the input differential Δa is unknown. In order to recover a, Δa has to be guessed firstly. Suppose Δa is an 8-bit non-zero value, which has 255 candidates. Usually, the m different output differential values should be related with the same input differential S-box, if these 7 output values are not in the same differential S-box when Δa =n, we can eliminate Δa =n, using this technique, we can get limited candidates of Δa, then case 2 (Δa is unknown) can be transferred into Case 1 (Δa is known), finally, after analyzing enough samples, a and the secret key can be obtained.

3 GENERAL OVERVIEW OF DFA ON CAMELLIA

3.1 Description of Camellia

A full description of the Camellia cipher is provided in [22], but below is a brief description of the cipher's properties that are utilized in this study.

Encryption Procedure: Camellia is an iterated cipher, its block length is 128-bit, and support 128,192,256-bit three key length types, it adopts the Feistel structure. In order to improve the security of the cipher, in the first and last round, the 128-bit data block is XORed with 128-bit pre-whitening and post-whitening keys, and the *FL* and FL^{-1} functions inserted every 6 rounds are used to provide non-regularity between the rounds. The Feistel structure for Camellia encryption can be written as follows:

$$\begin{cases} L_r = R_r \oplus F(L_{r-1}, k_r) \\ R_r = L_{r-1} \end{cases}$$

k_r denotes the r^{th} round key, $F=P \cdot S$ is the round function, the definition of the *S* and *P* function is

$$S: F_2^{64} \rightarrow F_2^{64}$$
$$l_{0(8)} \| l_{1(8)} \| l_{1(8)} \| l_{2(8)} \| l_{3(8)} \| l_{4(8)} \| l_{5(8)} \| l_{6(8)} \| l_{7(8)}$$
$$\rightarrow l'_{0(8)} \| l'_{1(8)} \| l'_{2(8)} \| l'_{3(8)} \| l'_{4(8)} \| l'_{5(8)} \| l'_{6(8)} \| l'_{7(8)}$$
$$l'_0 = s_0(l_0), l'_1 = s_1(l_1), l'_2 = s_2(l_2), l'_3 = s_3(l_3)$$
$$l'_4 = s_1(l_4), l'_5 = s_2(l_5), l'_6 = s_3(l_6), l'_7 = s_0(l_7)$$
$$P: F_2^{64} \rightarrow F_2^{64}$$
$$z_{0(8)} \| z_{1(8)} \| z_{2(8)} \| z_{3(8)} \| z_{4(8)} \| z_{5(8)} \| z_{6(8)} \| z_{7(8)}$$
$$\rightarrow z'_{0(8)} \| z'_{1(8)} \| z'_{2(8)} \| z'_{3(8)} \| z'_{4(8)} \| z'_{5(8)} \| z'_{6(8)} \| z'_{7(8)}$$
$$z'_0 = z_0 \oplus z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_7, z'_4 = z_0 \oplus z_1 \oplus z_5 \oplus z_6 \oplus z_7$$
$$z'_1 = z_0 \oplus z_1 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7, z'_5 = z_1 \oplus z_2 \oplus z_4 \oplus z_6 \oplus z_7$$
$$z'_2 = z_0 \oplus z_1 \oplus z_2 \oplus z_4 \oplus z_5 \oplus z_7, z'_6 = z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_7$$
$$z'_3 = z_1 \oplus z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6, z'_7 = z_0 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6$$

Key Schedule: Firstly, 4 12-bit variable K_L, K_R, K_A, and K_B are generated from the initial key and 6 64-bit constant $\sum_i$ (i = 1, 2, …, 6) by several F functions, then the 64-bit sub-keys k_{wt}, k_u, and k_{lv} are generated from K_L, K_R, K_A, and K_B, specific procedure is provided in [22].

3.2 Basic assumption and Notation

1 Assumption:

(1) One byte or more bytes random fault is induced into the memory registers storing the intermediate results in one fault induction. Notice that the attacker knows neither the location nor the concrete value of the fault.

(2) For any one plaintext adaptively selected, two different ciphertexts under the control of the same secret key are available, the right ciphertext and the faulty one.

(3) The faulty ciphertexts of the required type are presumably available. How to induce the specific fault is not covered in this paper, since this is not the main concern of our paper and many literatures on fault inductions are available [23]. The attacker should be able to identify the required faulty ciphertexts from a mass of faulty ciphertexts and discard faults occurring at a wrong timing.

(4) Only one master key is used during one attack.

2 Notation:

(1) K_r: the equivalent subkey of the r^{th} round, also is the exclusive OR half of the post-whitening subkey and the r^{th} subkey. In case of Camellia-128, the equivalent subkey for the 18^{th}, 17^{th}, 16^{th}, 15^{th}, 14^{th}, 13^{th} round is $K_{18}=k_{18} \oplus k_{w3}$, $K_{17}=k_{17} \oplus k_{w4}$, $K_{16}=k_{16} \oplus k_{w3}$, $K_{15}=k_{15} \oplus k_{w4}$, $K_{14}=k_{14} \oplus k_{w3}$, $K_{13}=k_{13} \oplus k_{w4}$ respectively.

(2) L_{r-1}, R_{r-1}: the 64-bit left and right half of the r^{th} round inputs.

(3) k_r: the 64-bit r^{th} round subkey.

(4) ΔIL_r^i, ΔIR_r^i: the i^{th} byte of the r^{th} round left and right half input differential value.($i \in [0,7]$)

(5) ΔOLr^i, ΔORr^i: the i^{th} byte of the r^{th} round left and right half output differential value.($i \in [0,7]$)

(6) ΔS_r^i, ΔP_r^i: the i^{th} byte of the r^{th} round *S* function and *P* function output differential value.($i \in [0,7]$)

(7) ΔCL^i, ΔCR^i: the i^{th} byte of the left and right half ciphertext differential value.($i \in [0,7]$)

(8) Fault: If not specially stated, fault denotes the non-zero differential value except for the faulty ciphertext.

3.3 Main idea of DFA on Camellia

The main idea of DFA on Camellia is as follows:

(1) Choose any plaintext *P*, and obtain the corresponding correct ciphertext *C*.

(2) Inject specific fault into the encryption procedure or key schedule, and obtain the faulty ciphertext *C**.

(3) Deduce one byte or several bytes of K_r using DFA technique.

(4) Repeat the above steps, until full K_r is recovered.

(5) Proceed in the same way and attack the previous round, and deduce K_{r-1}, K_{r-2}, K_{r-3}…., accordingly.

(6) Recover Camellia-128 key by K_{r-3}, K_{r-2}, K_{r-1},K_r and Camellia-192/256 key by K_{r-5}, K_{r-4},K_{r-3},K_{r-2},K_{r-1}, K_r with key reversion techniques.

(7) Verify the correctness of the recovered Camellia key.

In the next Sections, two improved differential fault analysis methods on Camellia by broadening fault width and enhancing fault depth are described, and the experimental results and comparisons are given to prove the correctness of the analysis theory.

4 Improved DFA on Camellia By Broadening Fault Width

4.1 Previous Study

Zhou's attack [12] is a generic attack based on model of Section 2. Its main idea is to inject single byte fault on the rth round left register L_{r-1} and use equitation (1) to retrieve K_r. Let's take recovering K_{18} as an example, the fault propagation is depicted in Fig. 2.

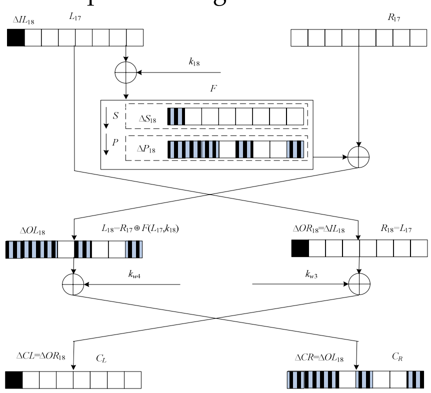

Fig. 2. Fault propagation of one byte fault in L_{17}.

The adversary first induces one byte fault ΔIL_{18}^0 to L_{17}^0, after the *S* function, ΔIL_{18} is transferred to single byte fault ΔS_{18}^0, after the *P* function, 5 or 6 bytes of ΔP_{18} have the same fault as ΔS_{18}^0, after the final swap and exclusive OR of k_{w3} and k_{w4}, ΔCL is equal to ΔIL_{18} and also is the *S* function input differential, ΔCR is equal to ΔP_{18} and its nonzero value is also the S function output differential. By applying DFA methods of Section 2, the adversary can recover K_{18}^0. Note that one byte fault can only recover one K_{18} byte from 256 to 2-4 candidates, and two times of the same location single byte fault can recover one K_{18} byte, so at least 16 faults are needed to recover K_{18}. By applying this method to the 17th, 16th, 15th round, K_{17}, K_{16}, K_{15} can be recovered, combing the key reverse techniques, the initial key K can be obtained.

4.2 Improved DFA on Camellia by Broadening Fault Width

In this section, we suppose the adversary has the ability of injecting multiple byte faults into L_{r-1}, this is much more practical than single byte fault attack in [12], also has been verified by hardware fault attack experiments in [11] recently. Let's take injecting m ($1 \le m \le 8$) faults into the 18th round left register L_{17} to recover m bytes of K_{18} as an example, the fault propagation is depicted in Fig. 3.

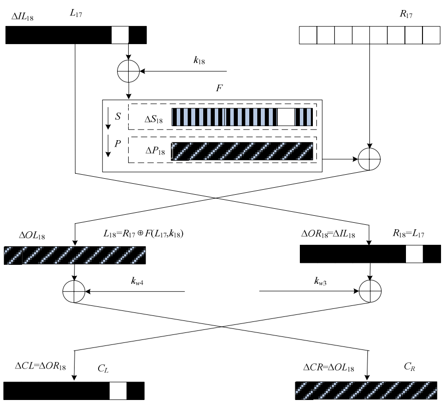

Fig. 3. Fault propagation of multiple byte faults in L_{17}.

Specific attacking procedure is as follows:

(1) Choose random plaintext P and obtain the correct ciphertext C under the secret key K.

(2) Induce m bytes random faults ΔIL_{18} into L_{17}, and obtain the faulty ciphertext C^*.

(3) Deduce the fault location.

Different fault locations injected into L_{17} can propagate the same location faults into C_L, according to the nonzero byte of ΔC_L, the attacker can easily identify the fault location injected into L_{17}.

(4)Compute the 18th round S-box input and output differential ΔIL_{18} and ΔS_{18}

According to Fig.3, the 18th round S-box input differential ΔIL_{18} is equal to the left half ciphertext differential ΔC_L, the 18th round S-box output differential ΔS_{18} can be computed by the Camellia reverse P function

$$
\begin{aligned}
\Delta S_{18}^{0} &= \Delta P_{18}^{1} \oplus \Delta P_{18}^{2} \oplus \Delta P_{18}^{3} \oplus \Delta P_{18}^{5} \oplus \Delta P_{18}^{6} \oplus \Delta P_{18}^{7}, \\
\Delta S_{18}^{1} &= \Delta P_{18}^{0} \oplus \Delta P_{18}^{2} \oplus \Delta P_{18}^{3} \oplus \Delta P_{18}^{4} \oplus \Delta P_{18}^{6} \oplus \Delta P_{18}^{7}, \\
\Delta S_{18}^{2} &= \Delta P_{18}^{0} \oplus \Delta P_{18}^{1} \oplus \Delta P_{18}^{3} \oplus \Delta P_{18}^{4} \oplus \Delta P_{18}^{5} \oplus \Delta P_{18}^{7}, \\
\Delta S_{18}^{3} &= \Delta P_{18}^{0} \oplus \Delta P_{18}^{1} \oplus \Delta P_{18}^{2} \oplus \Delta P_{18}^{4} \oplus \Delta P_{18}^{5} \oplus \Delta P_{18}^{6}, \\
\Delta S_{18}^{4} &= \Delta P_{18}^{0} \oplus \Delta P_{18}^{1} \oplus \Delta P_{18}^{4} \oplus \Delta P_{18}^{6} \oplus \Delta P_{18}^{7}, \\
\Delta S_{18}^{5} &= \Delta P_{18}^{1} \oplus \Delta P_{18}^{2} \oplus \Delta P_{18}^{4} \oplus \Delta P_{18}^{5} \oplus \Delta P_{18}^{7}, \\
\Delta S_{18}^{6} &= \Delta P_{18}^{2} \oplus \Delta P_{18}^{3} \oplus \Delta P_{18}^{4} \oplus \Delta P_{18}^{5} \oplus \Delta P_{18}^{6}, \\
\Delta S_{18}^{7} &= \Delta P_{18}^{0} \oplus \Delta P_{18}^{3} \oplus \Delta P_{18}^{5} \oplus \Delta P_{18}^{6} \oplus \Delta P_{18}^{7}
\end{aligned} \quad (2)
$$

(5) Recover K_{18}.

From step (1)-(4), we can recover the m bytes input and output differential of the 18th round S function ΔIL_{18}, ΔS_{18}, using DFA model of Section 2, it's easy to recover m bytes S function input value, which can be expressed as $L_{17} \oplus k_{18}$, as $L_{17} \oplus k_{w3} = C_L$, C_L is known, so m bytes of K_{18} can be recovered. Repeat above steps to recover full 64-bit K_{18}.

(6) Recover K_{17}, K_{16}, K_{15} …etc equivalent subkeys.

Proceed in the same way and attack, in turn, deduce the equivalent subkeys K_{r-2}, K_{r-3}…, accordingly, and retrieve the initial Camellia-128/192/256 key with methods in [13].

5 Improved DFA on Camellia By Extending Fault Depth

5.1 Previous Study

In Dec 2009, Zhao et. al. proposed an improved fault attack extending the fault depth of [12] in [13], the same method was also proposed in [14] . They supposed the adversary has the ability of injecting single byte fault into the r-1th round left Camellia register L_{r-2}. Let's take injecting one byte fault into the 17th round left register L_{16}^0 as an example, the fault propagation is depicted in Fig. 4.

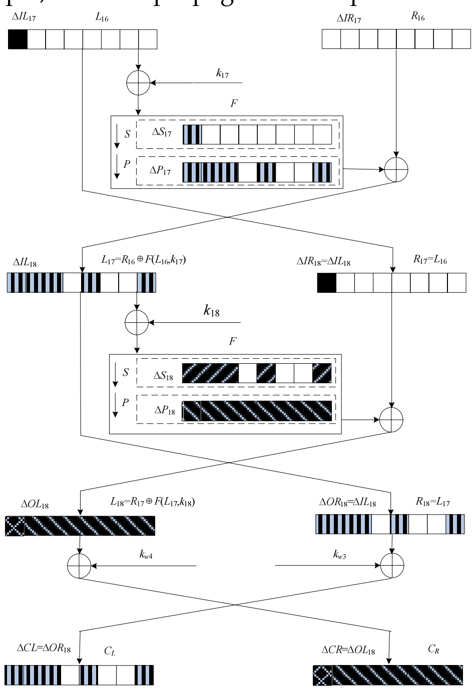

Fig. 4. Fault propagation of one byte fault in L_{16}.

Specific attacking procedure is as follows:

(1) Choose random plaintext P and obtain the correct ciphertext C under the secret key K.

(2) Induce one byte fault ΔIL_{17} into L_{16}, and obtain the faulty ciphertext C^* under the secret key K.

(3) Deduce the fault location.

Different fault locations injected into L_{16} can generate different indices sets of the fault C_L, we can identify the fault location by methods in [13].

(4)Compute the 18th round S-box input and output differential ΔIL_{18} and ΔS_{18}

ΔIL_{18} is equal to the left half ciphertext differential ΔC_L. ΔOL_{18} is equal to the left half ciphertext differential ΔC_R, and is generated by ΔS_{18}^0, ΔS_{18}^1, ΔS_{18}^2, ΔS_{18}^4, ΔS_{18}^7 and ΔIL_{17}^0 using 8 equations. It's simple to recover these 6 unknown differentials (ΔS_{18}^0, ΔS_{18}^1, ΔS_{18}^2, ΔS_{18}^4, ΔS_{18}^7 and ΔIL_{17}^0) by equation (3).

$$\left.\begin{aligned}
\Delta OL_{18}{}^0 &= \Delta IL_{17}{}^0 \oplus \Delta S_{18}{}^0 \oplus \Delta S_{18}{}^2 \oplus \Delta S_{18}{}^7\\
\Delta OL_{18}{}^1 &= \Delta S_{18}{}^0 \oplus \Delta S_{18}{}^1 \oplus \Delta S_{18}{}^4 \oplus \Delta S_{18}{}^7\\
\Delta OL_{18}{}^2 &= \Delta S_{18}{}^0 \oplus \Delta S_{18}{}^1 \oplus \Delta S_{18}{}^2 \oplus \Delta S_{18}{}^4 \oplus \Delta S_{18}{}^7\\
\Delta OL_{18}{}^3 &= \Delta S_{18}{}^1 \oplus \Delta S_{18}{}^2 \oplus \Delta S_{18}{}^4\\
\Delta OL_{18}{}^4 &= \Delta S_{18}{}^0 \oplus \Delta S_{18}{}^1 \oplus \Delta S_{18}{}^7\\
\Delta OL_{18}{}^5 &= \Delta S_{18}{}^1 \oplus \Delta S_{18}{}^2 \oplus \Delta S_{18}{}^4 \oplus \Delta S_{18}{}^7\\
\Delta OL_{18}{}^6 &= \Delta S_{18}{}^2 \oplus \Delta S_{18}{}^4 \oplus \Delta S_{18}{}^7\\
\Delta OL_{18}{}^7 &= \Delta S_{18}{}^0 \oplus \Delta S_{18}{}^4
\end{aligned}\right\}
\Rightarrow
\left.\begin{aligned}
\Delta S_{18}{}^0 &= \Delta OL_{18}{}^2 \oplus \Delta OL_{18}{}^5\\
\Delta S_{18}{}^1 &= \Delta OL_{18}{}^5 \oplus \Delta OL_{18}{}^6\\
\Delta S_{18}{}^2 &= \Delta OL_{18}{}^1 \oplus \Delta OL_{18}{}^2\\
\Delta S_{18}{}^4 &= \Delta OL_{18}{}^1 \oplus \Delta OL_{18}{}^4\\
\Delta S_{18}{}^7 &= \Delta OL_{18}{}^3 \oplus \Delta OL_{18}{}^5\\
\Delta IL_{17}{}^0 &= \Delta OL_{18}{}^0 \oplus \Delta OL_{18}{}^1 \oplus \Delta OL_{18}{}^3
\end{aligned}\right\} \quad (3)$$

(5) Recover 5 or 6 bytes of K_{18}.

By applying the general DFA model of Section 2, it's easy to recover 5 bytes S function input value of the 18th round, which can be expressed as $L_{17}^i \oplus k_{18}^i$ (i=0,1,2,4,7), as $L_{17}^i \oplus k_{w3}^i = C_L^i$, C_L is known, $k_{18}^i \oplus k_{w3}^i$ (i=0,1,2,4,7) can be recovered.

(6) Recover full 8 bytes of K_{18} and several bytes of K_{17}.

Repeat step (1)-(5) to recover full 8 bytes of K_{18}. Note that after K_{18} is recovered, as ΔIL_{17} is deduced, the output 17th round S-box differential ΔS_{17} can be obtained through ΔCL, the adversary can recover several bytes of K_{17}.

(7)Proceed in the same way as step (1)-(6) and attack the previous round, and deduce the equivalent subkeys K_{17}, K_{16}, K_{15}, …, accordingly, and retrieve the initial Camellia-128/192/256 key with methods in [13].

5.2 Improved DFA on Camellia by Extending Fault Depth

In this section, we describe idea of proposed DFA method to retrieve Camellia round keys by analyzing S-box input and output differentials.

Suppose the adversary has the ability of injecting single byte fault into the r-2th round left Camellia register L_{r-3}, let's take injecting one byte fault into the 16th round left register L_{15} as an example, the fault propagation is depicted in Fig. 5.

Specific attacking procedure is as follows:

(1) Choose random plaintext P and obtain the correct ciphertext C under the secret key K.

(2) Induce random single byte fault ΔIL_{16} into L_{15}, obtain the faulty ciphertext C^*.

(3) Compute the 18th round S-box lookup input differential ΔIL_{18}.

ΔIL_{18} can be computed from the left half ciphertext differential ΔCL.

(4) Deduce the 17th round S-box lookup output differential ΔS_{17}.

$$\Delta CL = \Delta IL_{18} = P(\Delta S_{17}) \oplus \Delta IR_{17} = P(\Delta S_{17}) \oplus \Delta IL_{16} \quad (4)$$

ΔCL has 8 nonzero bytes, ΔS_{17} has 5-6 nonzero bytes, ΔIL_{16} has only 1 nonzero byte, using similar equation as equation (3) above (if L_{15} fault byte index is 0), ΔS_{17} can be obtained, if the adversary didn't known the accurate fault index, there are 8 possibilities, specific method of solving the equations can be get from [13], finally, 8 candidates of ΔS_{17} can be obtained.

(5) Deduce the 17th round S-box lookup input differential ΔIL_{17}.

Next, we should recover ΔIL_{17} from ΔS_{17}. First we compute the 255 Camellia differential S-boxes, for each input differential S-box value ε ($\varepsilon = \Delta IL_{17}^0$) from 1 to 255, we can get 4 type of differential Camellia S-boxes, if 5 nonzero bytes of ΔS_{17} are among these 4 differential S-boxes (satisfying equation (5)), we can get one ΔIL_{17}^0 candidate, else ε will be eliminated, after 255 iterations, the adversary can get limited ΔIL_{17}^0 candidates.

$$\begin{aligned}
\varepsilon &= S^{-1}(IL_{17}{}^0 \oplus k_{17}{}^0) \oplus S^{-1}(IL_{17}{}^0 \oplus k_{17}{}^0 \oplus \Delta S_{17}{}^0)\\
\varepsilon &= S^{-1}(IL_{17}{}^1 \oplus k_{17}{}^1) \oplus S^{-1}(IL_{17}{}^1 \oplus k_{17}{}^1 \oplus \Delta S_{17}{}^1)\\
\varepsilon &= S^{-1}(IL_{17}{}^2 \oplus k_{17}{}^2) \oplus S^{-1}(IL_{17}{}^2 \oplus k_{17}{}^2 \oplus \Delta S_{17}{}^2)\\
\varepsilon &= S^{-1}(IL_{17}{}^4 \oplus k_{17}{}^4) \oplus S^{-1}(IL_{17}{}^4 \oplus k_{17}{}^4 \oplus \Delta S_{17}{}^4)\\
\varepsilon &= S^{-1}(IL_{17}{}^7 \oplus k_{17}{}^7) \oplus S^{-1}(IL_{17}{}^7 \oplus k_{17}{}^7 \oplus \Delta S_{17}{}^7)
\end{aligned} \quad (5)$$

(6) Deduce the 18th round S-box lookup output differential ΔS_{18}.

In order to recover K_{18}, the adversary should recover the 18th round S-box lookup output differential ΔS_{18}.

$$\Delta CR = \Delta OL_{18} = P(\Delta S_{18}) \oplus \Delta IR_{18} = P(\Delta S_{18}) \oplus \Delta IL_{17} \quad (6)$$

The equation (6) can be transferred into equation (7):

$$\Delta S_{18} = P^{-1}(\Delta CR \oplus \Delta IL_{17}) \quad (7)$$

ΔCR is known, if ΔIL_{17} is obtained, ΔS_{18} can be deduced.

(7) Recover full 8 bytes of K_{18}.

ΔS_{18} candidates can be computed from ΔIL_{17} candidates, ΔIL_{18} is known, by applying the general DFA model of Section 2, it's easy to recover limited K_{18} candidates. Repeat step (1)-(7) to recover K_{18}.

(8) Recover 5-6 bytes of K_{17}.

From step (7) the adversary can recover the 18th round S function input value $L_{17} \oplus k_{18} = C_L \oplus K_{18}$, and compute the 18th round P function output value, so the correct and faulty P_{18} can be computed, ΔP_{18} and unique 17th round S-box input differential ΔIL_{17} can be deduced, as the 17th round S-box output differential ΔS_{17} is recovered in step (4), by applying the general DFA model of Section 2, it's easy to recover $L_{16}^i \oplus k_{17}^i$ (i=0,1,2,4,7), as $L_{16} \oplus P_{18} \oplus kw_4 = C_R$, $S_{18}=S[C_L \oplus K_{18}]$, P_{18} denotes the 18th round P function output, it can be computed by S_{18}, so $k_{17}^i \oplus kw_4^i$ (K_{17}^i,i=0,1,2,4,7) can be recovered. Repeat step (1)-(8) to recover full 8 bytes of K_{17}.

(9) Recover 1 byte of K_{16}.

Note that after K_{17} is recovered, as the 16th round S-box input and output differential ΔIL_{16} and ΔS_{16} (equals ΔIL_{17}) is recovered, using basic DFA on Camellia, the adversary can even recover one byte of K_{16}.

(10) Proceed in the same way as step (1)-(9) and attack the previous round, and deduce the equivalent subkeys K_{16}, K_{15} … accordingly, and retrieve the initial Camellia-128/192/256 key with methods in [13].

The DFA method on Camellia encrypt procedure above can be extended into Camellia key schedule fault attack. Note that there is one thing different, injecting one byte fault into $k_{r\text{-}2}$ can recover 8 bytes of K_r, 5-6 bytes of $K_{r\text{-}1}$, but can not recover any byte of $K_{r\text{-}2}$, as the input differential of the r-2th round S function is unknown.

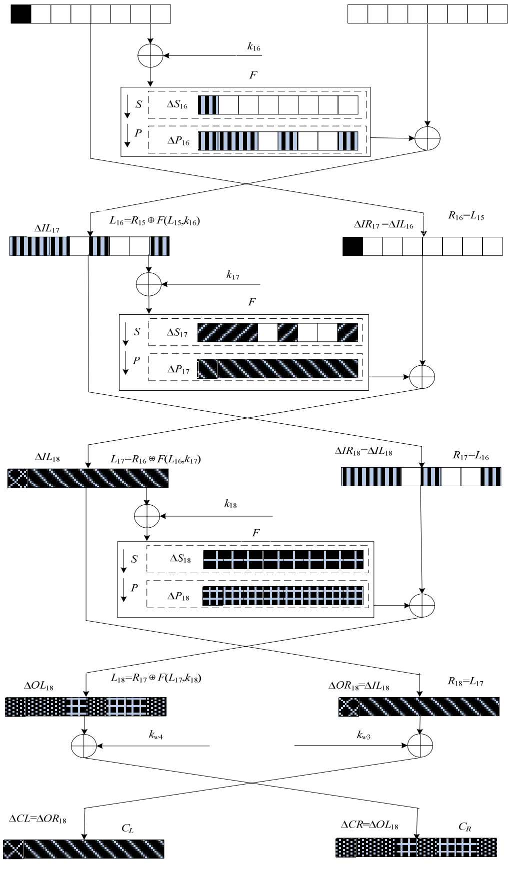

Fig. 5. Fault propagation of one byte fault in L_{15}.

6 Complexity Analysis and Experimental Results

6.1 Complexity Analysis

Due to the limit abilities of the attacker, it's very difficult to induce accurate and effective faults for DFA, so how many faulty ciphertexts are needed to crack the cipher is also very crucial. Next, we make a sketch of this complexity analysis.

1. Complexity Analysis of one byte fault analysis

We describe the characteristics of the equation for the S-box in the determinate methods. Let's just consider the simple one byte S-box model shown in Fig. 2. When we know the input S-box differential ΔIL_{18}^0 and the out differential ΔS_{18}^0, we can obtain a set of L_{17}^0 satisfying equation (8).

$$\left.\begin{aligned} &S[L_{17}^0 \oplus k_{18}^0] \oplus S[L_{17}^0 \oplus k_{18}^0 \oplus \Delta IL_{18}^0] = \Delta S_{18}^0 \\ &L_{17}^0 \oplus k_{w3}^0 = C^0 \\ &k_{18}^0 \oplus k_{w3}^0 = K_{18}^0 \end{aligned}\right\} \qquad (8)$$
$$\Rightarrow S[C^0 \oplus K_{18}^0] \oplus S[C^0 \oplus K_{18}^0 \oplus \Delta IL_{18}^0] = \Delta S_{18}^0$$

The number of K_{18}^0 depends on C^0, ΔIL_{18}^0, ΔS_{18}^0 and the S-box. In Camellia, four S-boxes S_0, S_1, S_2, S_3 are used in the F function. By solving equation (8), we examine the size of the key candidates $|K_{18}^0|$ for all combinations of C^0, ΔIL_{18}^0, ΔS_{18}^0 related with S_0. The total number of combinations is 16711680(ΔIL_{18}^0 is a non-zero value). The results are shown in Table 1, it's clear to see that by injecting single byte fault into L_{17}^0, about 2.0312 K_{18}^0 candidates can be obtained at one time, and the statistics of other three S-boxes are the same as S_0.

TABLE 1
Camellia S-box Statistics

$\lvert K_{18}^0\rvert$	Number	P	$E(\lvert K_{18}^0\rvert)$
2	16450560	0.9844	1.9688
4	261120	0.0156	0.0624
Total	16711680	1	$2.0312=2^{1.02}$

2. Complexity analysis of attack in Section 4.2.

Suppose the adversary injects m byte faults into $L_{r\text{-}1}$, according to Section 4.2, m bytes of K_r can be obtained. If m=8, one time 8-bytes faults in $L_{r\text{-}1}$ can reduce K_r space from 2^{64} to 289.75 (2.0312^8). As almost two times of the same index fault byte can recover one key byte, so theoretically speaking, the relationship between faulty number N and m obtaining unique K_r is

$$N = 16/m \qquad (9)$$

It's clear to see that the key recovery efficiency is increased with the faulty bytes width, and two times full 8 faulty bytes of $L_{r\text{-}1}$ can recover K_r.

3. Complexity analysis of attack in Section 5.2.

Injecting one byte fault in the r-2th round can propagate 5-6 faulty bytes in the r-1th round and 8 faulty bytes in the rth round, combined the analysis of Camellia S-box statistics and fault propagation feature, 2 faults are enough to recover K_r and reduce the search space of $K_{r\text{-}1}$ from 2^{64} to about $2^{10.6}$(1555.56) on average with 87.5% probabilities (if the two fault indices in the r-2th round are not identical), 3 faults are enough to recover K_r and reduce the search space of $K_{r\text{-}1}$ from 2^{64} to about $2^{3.8}$ on average with 98.4% probabilities (if all of the three fault indices in the r-2th round are not identical). As to Camellia-128 with and without FL/FL^{-1} layer, 2 faults in each of the 16th, 14th round, 4 faults are enough to reduce Camellia-128 key searching space to $2^{22.2}=2^{21.2+1}$. As to Camellia-192/256 without FL/FL^{-1} layer, 2 faults in each of the 16th, 14th, 12th round, 6 faults are enough to reduce Camellia-192/256 key searching space to $2^{31.8}$.

Note that if the attacker does not need to know the specific fault byte location, as for 2 times random fault injected into L_{15}, there are 64 fault location combinations. Only the correct combination can recover unique K_{18}, and the wrong combinations usually get empty candidates of

K_{18}. So, we can use this feature to deduce the fault location of these 2 faults, recover unique K_{18}, and choose corresponding method to recover 5-6 bytes of K_{17}.

6.2 Experimental Results and Comparisons

We have implemented simulations of the attacks in this paper written by Visual C++6.0 on Windows XP. Our simulations run on a personal computer (Athlon 64-bit 3000+ 1.81 GHz CPU and 1GB RAM) and successfully extract the Camellia-128/192/256 key.

In order to verify the complexity analysis of attack in Section 4.2 and 6.1, we implemented the 8 bytes width fault attack on Camellia 18th round, the statistics of 22 sets for 2000 sample's average 8 byte faults in L_{17} and K_{18} candidate number is depicted in Fig. 6. It's clear to see that one time 8 bytes fault on L_{17}, on average 289.74 candidates of K_{18} are obtained, which is almost the same as the 289.75 of the theory value in Section 6.1.

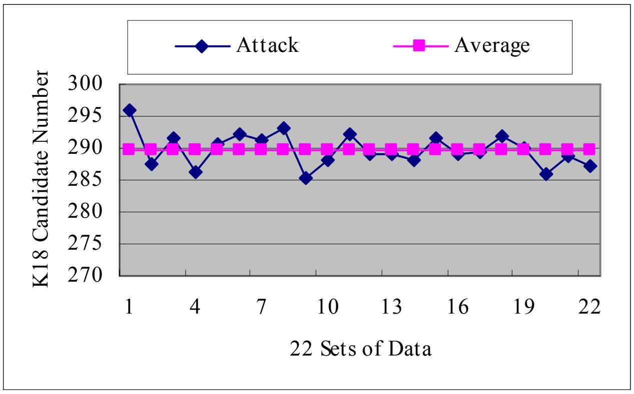

Fig. 6. 22 sets of Statistics on K_{18} candidate number.

In order to verify the complexity analysis of attack in Section 5.2 and 6.1, we injected two times single byte fault to L_{15}(two fault indices are not identical), used the analysis method of Section 5.2, unique K_{18} can be obtained, and the statistics of 10 sets for 5000 sample's K_{17} candidate number is depicted in Fig. 7. It's clear to see that two times single byte fault on L_{15}, on average 1557.01 candidates of K_{17} are obtained, which is almost the same as 1555.56 of the theory value in Section 6.1.

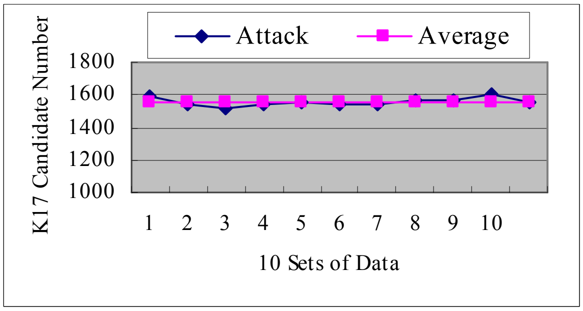

Fig. 7. 10 sets of Statistics onK_{17} candidate number.

Also, the experimental results in Table 2 strongly support the complexity analysis presented in Section 6.1.

It's clear to see that our methods are more effective than previous attacks and have the following properties:

1. Firstly, we broaden the fault depth of [10].

This is much more practical in the real attack scenarios, and verified by hardware fault attack experiments in [11] recently. We find out the key recovery efficiency is increased with the width of the faulty bytes width, two times full 8 faulty bytes in the $L_{r\text{-}1}$ can recover K_r, which is about 8-16 times efficient than [10]. Note that if the attacker can not inject single byte fault into Camellia encryption procedure accurately, the attack of Section 4.2 can be seen as the most powerful attack.

2. Secondly, we extend the fault depth of [10],[13],[14].

In Section 5.2, instead of injecting 1 byte fault into $L_{r\text{-}1}$ to recover 1 byte of K_r in [10], injecting 1 byte fault into $L_{r\text{-}2}$ to recover 5-6 bytes of K_r and 1 byte of $K_{r\text{-}1}$ in [13] and [14], we further enhance the fault depth by injecting 1 byte fault into $L_{r\text{-}3}$ to recover full 8 bytes of K_r, 5-6 bytes of $K_{r\text{-}1}$ and 1 byte of $K_{r\text{-}2}$. Under this assumption, the faulty ciphertexts number of our methods is further far less than [10],[13],[14], 4 faults are enough to reduce Camellia-128 key searching space to 2^{22}, 6 faults are enough to reduce Camellia-192/256 key searching space to $2^{31.5}$, which is almost 16 times and 4 times more efficient than [10] and [13],[14] respectively.

3. Thirdly, our DFA methods on Camellia encryption procedure in Section 5.2 can be easily adapted into DFA on Camellia key schedule, while [10] can not. It's impossible for [10] to inject fault into k_r to recover K_r (the adversary can not get the r^{th} round S-box input differential), however, according to the analysis of Section 5.2, it's easy to inject fault into $k_{r\text{-}1}$ and $k_{r\text{-}2}$ to recover K_r.

TABLE 2
IMPROVEMENT OF OUR DFA ATTACKS OVER PREVIOUS

Camellia	Attack	Fault Type	Fault Location	FL/ FL^{-1}	Fault Number
128	[10]	Single byte	L_{14} - L_{17}	×/√	64
128	Section 4.2	Multiple bytes	L_{14} - L_{17}	×/√	8
128	[13]	Single byte	L_{13} - L_{16}	×/√	16
128	Section 5.2	Single byte	L_{13}, L_{15}	×/√	4
128	[13]	Single byte	k_{14} - k_{17}	×/√	16
128	[14]	Single byte	k_{12} - k_{17}	×	30
128	Section 5.2	Single byte	k_{14}, k_{16}	×/√	4
192/256	[10]	Single byte	L_{12} - L_{17}	×/√	96
192/256	Section 4.2	Multiple bytes	L_{12} - L_{17}	×/√	12
192/256	[13]	Single byte	L_{11} - L_{16}	×	24
192/256	Section 5.2	Single byte	L_{11}, L_{13}, L_{15}	×	6
192/256	[13]	Single byte	k_{12} - k_{17}	×	24
192/256	[14]	Single byte	k_{12} - k_{17}	×	30
192/256	Section 5.2	Single byte	k_{12}, k_{14}, k_{16}	×	6
192/256	[13]	Single byte	L_{11} - L_{16}	√	32
192/256	Section 5.2	Single byte	L_{12}, L_{13}, L_{15}	√	16
192/256	Section 5.2	Single byte	k_{13}, k_{14}, k_{16}	√	16

7 CONCLUSION

Current studies of fault analysis on block ciphers are devoted to mathematical analysis on cryptographic algorithms, fault injection and detection of cryptographic algorithms in software and hardware implementation. In this paper, we examine the mathematical analysis of Camellia with its fault injection simulation in software implementation. For the hardware situation, we will leave it for the future research.

In this paper we present two improved DFA methods on Camellia. Our methods not only broaden the fault width, expand the fault depth, but also improve the efficiency of fault injection and decrease the number of faulty ciphertexts. Our best results demonstrate that 4 faults are enough to reduce Camellia-128 key searching space to $2^{22.2}$, 6 faults are enough to reduce Camellia-192/256 key searching space to $2^{31.8}$. Besides, our attack model can be adapted into most block ciphers with S-box, such as AES, ARIA, CLEFIA, SMS4, HYRAL, and MIBS.

Future analysis should be able to supply fault injection and detection of Camellia in hardware implementation. Moreover, we are working on the generic DFA sample size analysis model for block ciphers using S-box.

Acknowledgment

The authors wish to thank the anonymous reviewers for many helpful comments and suggestions. This work was supported in part by National Natural Science Foundation of China (Grant No. 60772082) and the Natural Science Foundation of Hebei Province. China (Grant No. 08M010).

References

[1] Boneh, D., DeMillo, R.A., Lipton, R.J. On the importance of checking cryptographic protocols for faults. In: Proc. of Advances in Cryptology–EUROCRYPT 2007, Fumy, W. (ed.), LNCS, vol. 1233, Berlin: Springer-Verlag, pp. 37–51, 1997.

[2] Biham, E., Shamir, A. Differential fault analysis of secret key cryptosystem. In: Proc. of Advances in Cryptology–CRYPTO 1997, Kaliski Jr., B.S. (ed.), LNCS, vol. 1294, Berlin: Springer-Verlag, pp. 513–525, 1997.

[3] Biehl, I., Meyer, B., Muller, V. Differential fault analysis on elliptic curve cryptosystems. In: Proc. of Advances in Cryptology–CRYPTO 2000, Bellare, M. (ed.), LNCS, vol. 1880, Berlin: Springer-Verlag, pp. 131–146, 2000.

[4] Hemme, L.: A differential fault attack against early rounds of (Triple-) DES. In: Proc. of Cryptographic Hardware and Embedded Systems–CHES2004, Joye, M., Quisquater, J.-J. (eds.) , LNCS, vol. 3156, Berlin: Springer-Verlag, pp. 254–267, 2004.

[5] Blomer, J., Seifert, J.P.: Fault based cryptanalysis of the Advanced Encryption Standard (AES). In: Proc. of Finiancial Cryptology–FC 2003, Wright, R.N. (ed.), LNCS, vol. 2742, Berlin: Springer-Verlag, pp. 162–181, 2003.

[6] Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on AES. In: Proc. of International Conference on Applied Cryptography and Network Security–ACNS 2003, Zhou, J., Yung, M., Han, Y. (eds.), LNCS, vol. 2846, Berlin: Springer-Verlag, pp. 293–306, 2003.

[7] Piret, G., Quisquater, J.J. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad. In: Proc. of Cryptographic Hardware and Embedded Systems–CHES 2003, Walter, C.D., Koç, Ç.K., Paar, C. (eds.), LNCS, vol. 2779, Berlin: Springer-Verlag, pp. 77–88, 2003

[8] Debdeep Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. In: Proc. of Advances in Cryptology–AFRICACRYPT 2009, B. Preneel (eds.), LNCS, vol. 5580, Berlin: Springer-Verlag, pp. 421–434, 2009.

[9] Michael Tunstall, Debdeep Mukhopadhyay. Differential Fault Analysis of the Advanced Encryption Standard using a single Fault. Cryptology ePrint Archive 2009/575, available at http://eprint.iacr.org/2009/575.pdf.

[10] Dhiman Saha, Debdeep Mukhopadhyay, Dipanwita Roy-Chowdhury. A Diagonal Fault Attack on the Advanced Encryption Standard. Cryptology ePrint Archive 2009/581, available at http://eprint.iacr.org/2009/581.pdf

[11] Toshinori Fukunaga and Junko Takahashi. Practical Fault Attack on a Cryptographic LSI with ISO/IEC 18033-3 Block Ciphers. In: Proc. of 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography–FDTC2009, IEEE Computer Society, pp.84-92, 2009.

[12] Yong-bin Zhou, Weng-ling WU, Nan-nan XU, Deng-guo FENG. Differential Fault Attack on Camellia. Chinese Journal of Electronics, Vol.18, No.1, pp. 13–19, 2009.

[13] Xin-jie Zhao, Tao Wang. An Improved Differential Fault Attack on Camellia, Cryptology ePrint Archive 2009/585, available at http://eprint.iacr.org/2009/585.pdf.

[14] Wei Li, Da-wu Gu, Juan-ru Li et. al. Differential fault analysis on Camellia. The Journal of Systems and Software, Vol.83, Issue 5, pp. 844–851, 2010.

[15] Wei Li, Da-wu Gu, Juan-ru Li. Differential fault analysis on the ARIA algorithm. Information Sciences, Vol.178, Issue 19, pp.3727–3737, 2008.

[16] Hua Chen, Wen-ling Wu, and Deng-guo Feng. Differential Fault Analysis on CLEFIA. In: Proc. of Eighth International Conference on Information and Communications Security–ICICS 2007, S. Qing, H. Imai, and G. Wang (Eds.), LNCS, vol. 4861, Berlin: Springer-Verlag, pp. 284–295, 2007.

[17] Junko Takahashi and ToshinoriFukunaga. Improved Differential Fault Analysis on CLEFIA. In: Proc. of the 5th Workshop on Fault Diagnosis and Tolerance in Cryptography– FDTC 2008, IEEE Computer Society, pp.25-34, 2008.

[18] Hoch, J.J., Shamir, A.: Fault analysis of stream ciphers. In: Proc. of Cryptographic Hardware and Embedded Systems–CHES 2004, Joye, M., Quisquater, J.-J. (eds.), LNCS, vol. 3156, Berlin: Springer-Verlag, pp. 240–253, 2004.

[19] Biham, E., Granboulan, L., Nguyn, P.Q. Impossible fault analysis of RC4 and differential fault analysis of RC4. In: Proc. of Fast Software Encryption–FSE 2005, Gilbert, H., Handschuh, H. (eds.), LNCS, vol. 3557, Berlin: Springer-Verlag, pp. 359–367, 2005.

[20] M. Hojsik and B. Rudolf. "Floating fault analysis of Trivium," In: D.R. Chowdhury, V. Rijmen, and A. Das (eds.) In: Proc. of INDOCRYPT 2008, LNCS, vol. 5365, Berlin: Springer-Verlag, pp. 239–250, 2008.

[21] Yupu Hu, Juntao Gao and Qing Liu. Hard Fault Analysis of Trivium. Cryptology ePrint Archive 2009/333, available at http://eprint.iacr.org/2009/333.pdf.

[22] K. Aoki, T. Ichikawa, M. Kansa, M. Matsui, S. Moriai, Nakajima, and T. Tokita, Specification of Camellia - a 128-bit Block Cipher,2000, available at http://www.cosic.esat.kuleuven.be/nessie/workshop/submissions.

[23] C.Giraud, H.Thiebeauld, A survey on fault attacks. In: Proc. of 6th International Conference on Smart Card Research and Advanced Applications–CARDIS O4, Toulouse, France, pp. 22—27, 2004.

Xin-jie Zhao, Tao Wang: Department of Computer Engineering, Ordnance Engineering College, Shijiazhuang 050003, China, zhaoxinjieem@163.com.