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

Xin-jie ZHAO, Tao WANG

Department of Computer Engineering, Ordnance Engineering College, Shijiazhuang 050003, China, zhaoxinjieem@163.com

Abstract. This paper presents several further improved attacks on Camellia. In Jan 2009, Yong-bin ZHOU proposes the first DFA attack on Camellia by injecting 1 byte fault into the r^{th} round left register to recover 1 K_r equivalent subkey byte and obtains Camellia-128,192/256 key with 64 and 96 faulty ciphertexts. In Dec 2009, Xin-jie ZHAO proposes an improved attack extending the fault depth by injecting single byte fault into the r-1^{th} round left register to recover 5-6 bytes of K_r, 1 byte of K_{r-1} and obtains Camellia-128,192/256 key with 16 and 24 faulty ciphertexts. In this work, we present two further improved DFA attacks on Camellia. Our first attack broadens Yong-bin ZHOU's fault width, injects multiple byte faults into the r^{th} round left register to recover multiple bytes of K_r, and obtains Camellia-128,192/256 key with at least 8 and 12 faulty ciphertexts; our second attack further extends Xin-jie ZHAO's fault depth, injects single byte fault into the r-2^{th} round left register or r-2^{th} round key to recover full 8 bytes of K_r, 5-6 bytes of K_{r-1}, 1 byte of K_{r-2}, and obtains Camellia-128,192/256 key with 4 and 6 faulty ciphertexts. Simulation experiments demonstrate that: due to the reversible permutation function of Camellia, Camellia is quite weak for multiple byte faults attack, and the attack efficiency is even increased with fault width, this feature great improves fault attack's practicalities; due to the Feistel structure of Camellia, Camellia is also quite weak for deeper single byte fault attack, 4 faulty ciphertexts are enough to recover Camellia-128 with 2^{22} brute force search, 6 faulty ciphertexts are enough to recover Camellia-192/256 with $2^{31.5}$ brute force search.

Keywords: Differential fault analysis; Feistel structure; SPN structure; Block cipher; Camellia; Encryption procedure; Key schedule; S-box lookup; Fault width; Fault depth.

1 Introduction

The idea of fault attack was first suggested in 1997 by Boneh, DeMillo and Lipton[1], which makes use of the faults during the execution of a cryptographic algorithm. Under the idea, the attack was successfully exploited to break an RSA-CRT with both a correct and a faulty signature of the same message. Shortly after, Biham and Shamir proposed an attack on secret key cryptosystems called Differential Fault Analysis (DFA)[2], which combined the ideas of fault attack and differential attack. Since that, many research papers have been published using this cryptanalysis technique to successfully attack various cryptosystems, including ECC[3],3DES[4],AES[5-10],Camellia[11-12],ARIA[13],CLEFIA[14-15],RC4[16-17],Trivium[18-19] and so on.

Camellia is a 128-bit block cipher jointly developed by NTT and Mitsubishi Electric Corporation in 2000[20]. It is 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 is integrated into the OPENSSL-1.0.0-beta1.

In Jan 2009, the first DFA attack on Camellia is proposed by Yong-bin ZHOU et. al.[11], they inject a single byte fault into the r^{th} round left register to recover 1 K_r equivalent subkey byte, and after injecting single byte fault to the 18^{th},17^{th},16^{th},15^{th} round of Camellia left registers, obtain Camellia-128,192/256 key with 64 and 96 faulty ciphertexts under ideal conditions. In practical, it's not easy to inject 8 bytes left registers twice within 16 times, so more than 100 and 150 faulty ciphertexts are needed to recover Camellia-128,192/256 key. In Dec 2009, Xin-jie ZHAO proposes an improved fault attack[12] extending Yong-bin ZHOU's fault depth, injects a single byte fault into the r-1^{th} Camellia round left register to recover 5-6 bytes of K_r and 1 byte of $K_{r\text{-}1}$ and obtains Camellia-128,192/256 key with 16 and 24 faulty ciphertexts, thus greatly improves [11]'s efficiency.

In this paper, we analyze the basic DFA attack principle and summarize the DFA on block cipher with S-box into computing the S-box input and output differential problems, then present two further improved DFA attacks on Camellia. Our first type of attack broadens Yong-bin ZHOU's fault width by injecting multiple m byte($1 \le m \le 8$) faults into the r^{th} round left register to recover m bytes of K_r, if N equals 8, 8 and 12(16) faults are enough to recover Camellia-128, 192/256 key. The attack in [11] is a specialized case of our first attack when m=1, and compared with [11], our methods not only broaden the fault width, but also improve the fault analysis efficiency and attack practicalities by almost 8 times at the best case. Our second type of attack further extends Xin-jie ZHAO's fault depth, injects single byte fault into the r-2^{th} round left register or r-2^{th} round key to recover 8 bytes of K_r, 5-6 bytes of $K_{r\text{-}1}$ and 1 byte of $K_{r\text{-}2}$, so 4 and 6 faulty ciphertexts are enough to recover Camellia-128,192/256 key with 2^{22} and $2^{31.5}$ brute force search. Compared with [11] and [12], our methods not only enhance the fault depth, but also improve the fault analysis efficiency by 16 times and 4 times respectively, and decrease the faulty ciphertexts number. Besides, our second attack can be easily extended to DFA on Camellia key schedule case, while [11] can not.

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 discusses on the contradictions between traditional cipher design and implementation attacks, Section 8 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 state byte a becomes a^*, the differential value Δa ($\Delta a = a \oplus a^*$) can be either known or unknown, usually, the adversary can get the output differential value Δc. 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 output whitenings by Xored the last round key K_l to generate the ciphertexts C. As C is known, if a is obtained, K_l 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

If Δa is known, this case is usually related with Feistel structure block cipher. If one byte fault Δa is injected into $L_{r\text{-}1}$, due to the feature of Feistel structure, both Δa and Δc can be obtained after analyzing the cipher differential ΔC.

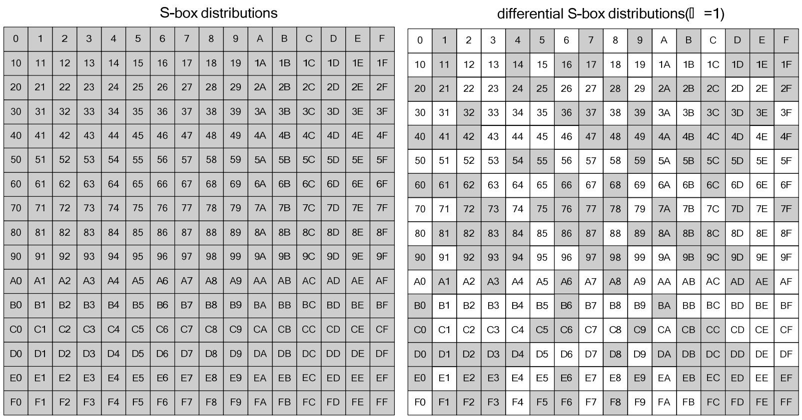


Fig. 1. Camellia S box and differential S box (Δ=1) sorted ascending.

If we input every possible value of a into formula (1), we can get limited candidates of a satisfying formula (1). Fig. 2 is the Camellia S-box and differential S-box (Δ=1) 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 candidate 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' ($S'[i]=S[i] \oplus S[i \oplus \Delta]$, Δ=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 possible candidate of a into formula (1), 2-4 candidates of a can be obtained, which means that we can also get 2-4 candidates for Camellia equivalent key.

2. Δa is unknown

If Δa is unknown, this case is usually related with SPN structure block cipher. Unlike Feistel 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 r^{th} S function, the faults are propagated into m differential faults. The S-box output differential is known, but the input differential is unknown. In order to recover a, Δa has to be guessed. 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 $\Delta a = n$, we can eliminate $\Delta a = n$, 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 more samples, a and the secret key can be obtained.

3 General Overview of DFA on Camellia

3.1 Basic assumptions and Notations

1. Assumptions:

(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 [21]. 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 user key is used during one successfully attack.

2. Notations:

A full description of the Camellia cipher is provided in [20], but below is the brief notations of Camellia utilized in this study.

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

(2) L_{r-1}, R_{r-1}: the 64-bit left and right halves 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 besides the faulty ciphertext.

3.2 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 corresponding faulty ciphertext C^*.
(3) Deduce one byte or several bytes of Camellia equivalent subkey using differential fault analysis technique.
(4) Repeat the above steps, until all 8 bytes of K_r are recovered.
(5) Proceed in the same way and attack the previous round, and deduce the equivalent subkeys K_{r-1}, K_{r-2}, K_{r-3}…., accordingly.
(6) Recover Camellia-128 key by analyzing K_{r-3}, K_{r-2}, K_{r-1},K_r and Camellia-192/256 key by analyze 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, several improved differential fault attacks 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

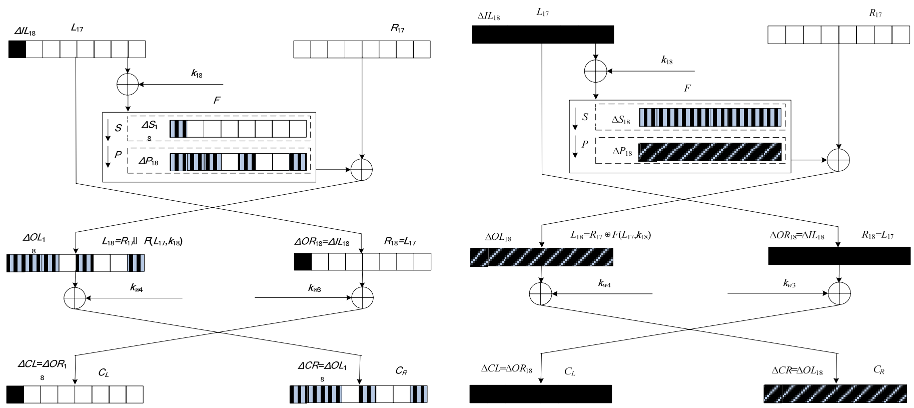
4.1 Yong-bin ZHOU's DFA attack: Inject Single byte Fault into the r^{th} round to recover one byte of K_r

Yong-bin ZHOU's attack [11] is a generic attack based on model of Section 2. It's main idea is to inject single byte fault on L_{r-1} and use equitation (1) to recover K_r. Specifically speaking, let's take recovering K_{18} as an example, the fault propagation process is depicted in Fig. 2 (a). The adversary first induces one byte fault ΔIL_{18}^0 to L_{17}^0, then after the S function, the input differential ΔIL_{18} is transferred into single byte fault ΔS_{18}^0, after the P function, 5 or 6 bytes of the output ΔP_{18} have the same fault as ΔS_{18}^0, after the final swap and exclusive OR of k_{w3} and k_{w4}, the ΔCL is equal to ΔIL_{18}

and also is the input S function differential value, the ΔCR is equal to ΔP_{18} and also is the output S function differential value, by applying DFA methods in Section 2, the adversary can recover $L_{17}{}^{0} \oplus k_{18}{}^{0}$, as $L_{17}{}^{0} \oplus k_{w3}{}^{0}{=}C_{L}{}^{0}$. Note that one byte fault can only recover one K_{18} byte from 256 to 2-4 candidates, and two times of the same 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 Inject Multi-byte Faults into the r^{th} round to recover Multi-bytes of K_r

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 ZHOU's attack by inject one single byte fault into L_{r-1}. Let's take injecting m ($1 \le m \le 8$)faults into L_{17} to recover m bytes of K_{18} as an example, the fault propagation process is depicted in Fig. 2 (b)(m=8).



(a) ZHOU's attack (b) This paper

Fig. 2. Fault propagation of attacking the Camellia 18th round to recover K_{18}.

Specific attacking procedures are as follows:

(1) Choose randomly 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^*.

The faulty propagate procedure is as follows: When m bytes fault ΔIL_{18} is injected into L_{17}, then after the 18th round S function, ΔS_{18} has m nonzero bytes, after the 18th round P function, full 8 bytes of fault ΔP_{18} were propagated, after the exclusive OR of R_{17}, the left output differential ΔOL_{18} is equal to ΔP_{18}, and the right output differential ΔOR_{18} is equal to ΔIL_{18}. After the 18th round, the output differential of C_L is equal to ΔIL_{18}, and the output differential of C_R is equal to ΔP_{18} and ΔOL_{18}.

(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) Deduce ΔS_{18}.

ΔP_{18} is equal to ΔC_R and ΔOL_{18}, and can be obtained directly from the differential of the correct and faulty ciphertext. As shown in equation (2), ΔS_{18} can be computed by the Camellia reverse P function.

$$\begin{cases} \Delta P_{18}{}^{0} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{6} \oplus \Delta S_{18}{}^{7}, \\ \Delta P_{18}{}^{1} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{6} \oplus \Delta S_{18}{}^{7}, \\ \Delta P_{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}{}^{5} \oplus \Delta S_{18}{}^{7}, \\ \Delta P_{183} = \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{6}, \\ \Delta P_{18}{}^{4} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{6} \oplus \Delta S_{18}{}^{7}, \\ \Delta P_{18}{}^{5} = \Delta S_{18}{}^{1} \oplus \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{6} \oplus \Delta S_{18}{}^{7}, \\ \Delta P_{18}{}^{6} = \Delta S_{18}{}^{2} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{7}, \\ \Delta P_{18}{}^{7} = \Delta S_{18}{}^{0} \oplus \Delta S_{18}{}^{3} \oplus \Delta S_{18}{}^{4} \oplus \Delta S_{18}{}^{5} \oplus \Delta S_{18}{}^{6} \end{cases} \Rightarrow \begin{cases} \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{cases} \qquad \textbf{(2)}$$

(5) Recover K_{18}.

From step (1)-(4), we can recover the m bytes input/output differential of the 18th round S function $\Delta IL_{18}/\Delta S_{18}$, 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 to the attacker, so m bytes of K_{18} ($K_{18}=k_{18} \oplus k_{w3}=L_{17} \oplus k_{18} \oplus C_L$) can be recovered. Repeat above steps to recover full 8 bytes of 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 $K_{r\text{-}2}$, $K_{r\text{-}3}$…., accordingly.

(7) Recover initial Camellia-128/192/256 key with methods in [12].

5 Improved DFA on Camellia By Extending Fault Depth

5.1 Xin-jie ZHAO's attack: inject Faults into the *r*-1th to recover K_r, $K_{r\text{-}1}$

Suppose the adversary has the ability of injecting single byte fault into the r-1th round left Camellia register $L_{r\text{-}2}$. Let's take injecting one byte fault into the 17th round left register L_{16} to recover 5-6 bytes of K_{18} and one byte of K_{17} as an example, the fault propagation process is depicted in Fig. 3 (a). Specific attacking procedures are as follows:

(1) Choose randomly 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 [12].

(4) Deduce ΔS_{18} and ΔIL_{17}^0.

ΔOL_{18} is known to the attacker by analyzing $C_R \oplus C_R^*$, and 8 ΔOL_{18}^i value is generated by ΔS_{18}^0, ΔS_{18}^1, ΔS_{18}^2, ΔS_{18}^4, ΔS_{18}^7 and ΔIL_{17}^0. All of this can generate 8 equations, it's quite simple to recover these 6 unknown differential value (ΔS_{18}^0, ΔS_{18}^1, ΔS_{18}^2, ΔS_{18}^4, ΔS_{18}^7 and ΔIL_{17}^0) by equation (3).

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

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

From step (4) we can recover 5 output differential bytes of the 18th S function, as the input differential is 5 equal faulty byte value ΔS_{17}^0. By applying the general DFA model of Section 2, it's easy to recover 5 different S function input value, 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.

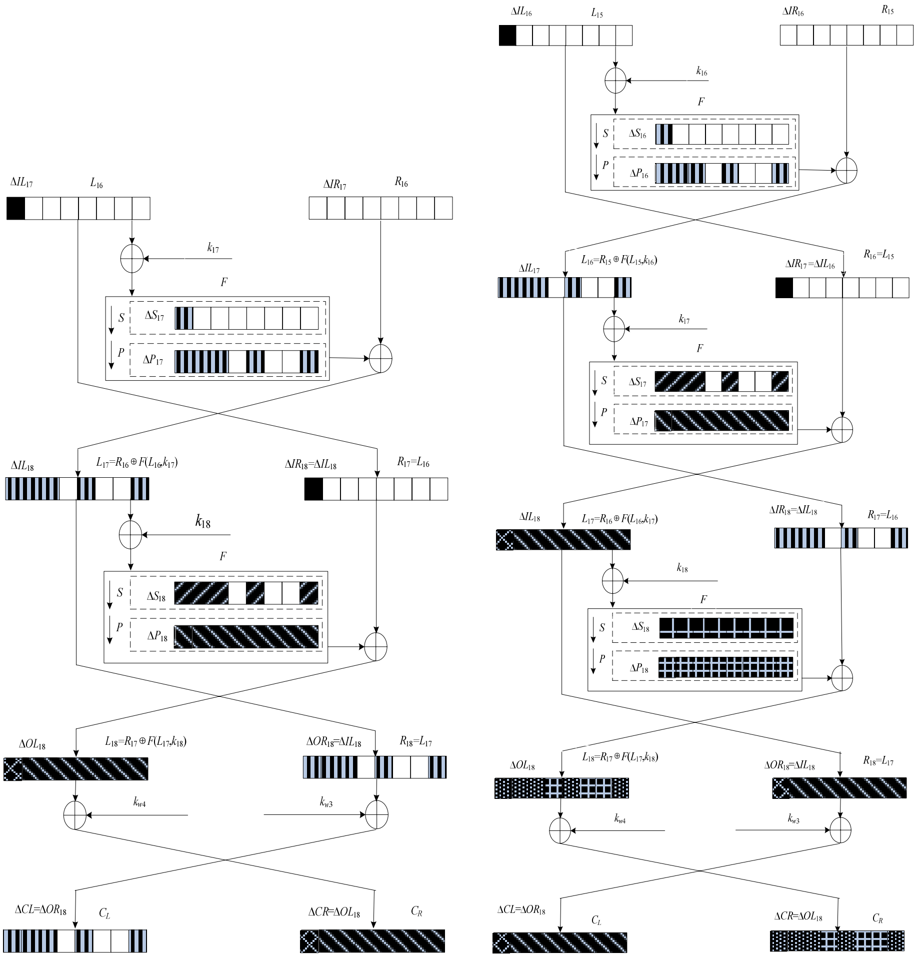
(6) Recover full 8 bytes of K_{18} and several bytes of K_{17}.

Repeat step (1)-(5) to recover full 8 bytes of K_{18}. Note that after K_{18} is recovered, as ΔIL_{17} is recovered, and the output 17th round S-box differential value 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.

5.2 Inject Faults into the r-2th Encryption round to recover K_r, K_{r-1}, K_{r-2}

Suppose the adversary has the ability of injecting single byte fault into the r-2th round left Camellia register L_{r-3}, just take injecting one byte fault into the 16th round left register L_{15} to recover full 8 bytes of K_{18}, 5-6 bytes of K_{17} and one byte of K_{16} as an example, the fault propagation process is depicted in Fig. 3 (b).



(a) Xin-jie ZHAO's attack (b) This paper

Fig. 3. Fault propagation of attacking the Camellia 17th and 16th round to recover K_{18}.

Specific attacking procedures are as follows:

(1) Choose randomly 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}.

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

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) (if L_{15} fault index is 0) above, ΔS_{17} can be obtained, if the adversary didn't known the accurate fault index, there are 8 possibilities, specific solving methods of solving the equations can be get from [12], finally, 8 candidates for ΔS_{17} can be obtained.

(5) Deduce the 17th round S-box lookup output 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 S-boxes (satisfy equation (5)), we can get one $\Delta IL_{17}{}^{0}$ candidate, else ε will be eliminated, after 255 iterations, the adversary can get limited $\Delta IL_{17}{}^{0}$ candidates.

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

(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 $\Delta S_{18.}$

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

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

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

ΔCR is known, if ΔIL_{17} is recovered, ΔS_{18} can be computed.

(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 the 18th round 8 S function input bytes $L_{17} \oplus k_{18}$, as $L_{17} \oplus k_{w3} = C_L$, C_L is known, K_{18} candidates can be recovered. 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} can be recovered, and the unique value of ΔIL^{17} can be obtained, as the 17th round S-box output differential ΔS_{17} is recovered in step (4), so by applying the general DFA model of Section 3.2, 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 $k_{17}{}^{i} \oplus kw_4{}^{i}$, which is also $K_{17}{}^{i}$(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 every input 16th round S-box single byte fault ΔIL_{16} is recovered, and the output 16th round S-box differential value ΔS_{16} (equals

ΔIL_{17}) can be 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.

5.3 Inject Faults into the Round Key k_{r-2} to recover K_r, K_{r-1}

The DFA method of Section 5.2 can be extended into Camellia key schedule fault attack. There is one thing different, injecting one byte fault into k_{r-2} can recover 8 bytes of K_r, 5-6 bytes of K_{r-1}, but can not recover any byte of K_{r-2}, as the input differential of the r-2th round S function is unknown.

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 (a). When we know the input S-box differential $\Delta IL_{18}{}^0$ and the out differential $\Delta S_{18}{}^0$, we can obtain a set of $L_{17}{}^0$ satisfying equation (8).

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

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

Table 1. 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=(2^{24}-256)	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 depicted in formula (9):

$$N = 16/m \tag{9}$$

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

3. Complexity analysis of attack in Section 5.2 and Section 5.3.

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 r^{th} round, 2 faults are enough to recover K_r and reduce the search space of $K_{r\text{-}1}$ from 2^{64} to about $2^{10.5}$ 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 the three fault indices in the r-2th round are not identical). As to Camellia-128, 2 faults in each of the 16th, 14th round, totally 4 faults are enough to reduce Camellia-128 key searching space from 2^{128} to 2^{22}. As to Camellia-192/256, 2 faults in each of the 16th, 14th, 12th round, totally 6 faults are enough to reduce Camellia-192/256key searching space to $2^{31.5}$.

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

6.2 Experimental Results and Comparisons

We have implemented simulations of the attacks given in this paper. The simulations are written in 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 verified 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} candidates number is depicted in Fig. 4. 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.

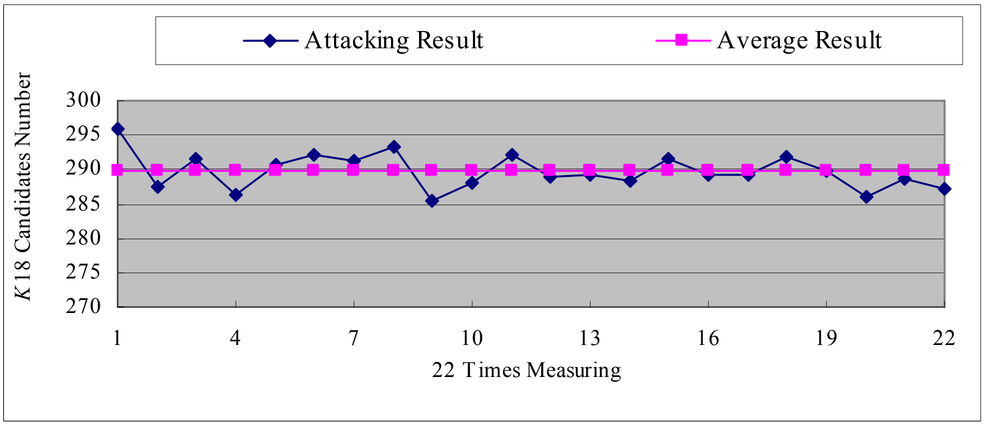


Fig. 4. Statistics of 22 sets for 2000 sample's average 8 byte faults in L_{17} and K_{18} candidates number

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

Table 2. Improvement of our attacks over previous Camellia DFA work.

Attack	Camellia	Fault Type	Fault Location	FL/FL^{-1}	Fault No
[11]	Camellia-128	Single byte in the 15th - 18th round	L_{14} - L_{17}	x /√	64
[11]	Camellia-192/256	Single byte in the 13th -18th round	L_{12} - L_{17}	x /√	96
Section 4.2	Camellia-128	Multi-bytes in the 15th - 18th round	L_{14} - L_{17}	x /√	8
Section 4.2	Camellia-192/256	Multi-bytes in the 13th -18th round	L_{12} - L_{17}	x /√	12
[12]	Camellia-128	Single byte in the 14th -17th round	L_{13} - L_{16}	x /√	16
[12]	Camellia-192/256	Single byte in the 12th -17th round	L_{11} - L_{16}	x	24
[12]	Camellia-192/256	Single byte in the 12th -17th round	L_{11} - L_{16}	√	32
[12]	Camellia-128	Single byte in the key schedule	k_{14} - k_{17}	x /√	16
[12]	Camellia-192/256	Single byte in the key schedule	k_{12} - k_{17}	x	24
Section 5.2	Camellia-128	Single byte in the 14th, 16th round	L_{13}, L_{15}	x /√	4
Section 5.2	Camellia-192/256	Single byte in the 12th, 14th, 16th round	L_{11}, L_{13}, L_{15}	x	6
Section 5.2	Camellia-192/256	Single byte in the 12th -17th round	L_{12}, L_{13}, L_{15}	√	16
Section 5.3	Camellia-128	Single byte in the key schedule	k_{14}, k_{16}	x /√	4
Section 5.3	Camellia-192/256	Single byte in the key schedule	k_{12}, k_{14}, k_{16}	x	6

It's clear to see that our DFA methods are much more effective than former attacks; our attacks have the following properties:

(1) Firstly, we broaden the fault depth of [11].

This is much more practical in the real attack scenarios. We find out the key recovery efficiency is increased with 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 [11]. Note that if the attacker can not accurate inject single byte fault into Camellia encryption procedure, the attack of Section 4.2 can be seen as the most powerful attack.

(2) Secondly, we enhance the fault depth of [11] and [12].

Instead of injecting 1 byte fault into $L_{r\text{-}1}$ to recover 1 byte of K_r in [11], our former attack in [12] injects 1 byte fault into $L_{r\text{-}1}$ to recover 5-6 bytes of K_r and 1 byte of $K_{r\text{-}1}$, and improved the efficiency of recovering K_r by 5-6 times. In Section 5.2, we further enhance the fault depth by injecting 1 byte fault into $L_{r\text{-}2}$ 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 [11]. 4 faults are enough to reduce Camellia-128 key searching space from 2^{128} to 2^{22}, 6 faults are enough to reduce Camellia-192/256 key searching space to $2^{31.5}$, which is almost 16 times more efficiency than [11].

(3) Thirdly, our DFA methods on Camellia encryption procedure in Section 5.2 can be easily adapted into DFA on Camellia key schedule case, while [11] can not. It's impossible for [11] to inject fault into k_r to recover K_r (the adversary can not get the input differential value of the r^{th} round S-box lookup), however, according to the analysis of the Section 5.3, it's quite easy to inject fault into $k_{r\text{-}1}$ and $k_{r\text{-}2}$ to recover K_r.

7 Discussions

In the cryptography design, cryptographists usually add more non-linear (S-box lookups) and complicated linear operations to prevent ciphers from linear and differential attack and it indeed works well on traditional mathematical analysis. However, when it comes to the cryptosystem implementation, it meets unprecedented challenges. As the non-linear part operations, S-box is leaking more information on secret key. Usually, the input of the S-box is related with one plaintext/ciphertext byte, one initial key or subkey byte, and the elements of the S-box is open to the public. Next, we take the S-box lookup as an example and discuss the relationships between it and implementation analysis.

(1) Cache based attack (CBA)

Cache hit and miss feature can affect the whole encryption time and the accessed Cache sets information of the cryptosystems, this can be utilized as timing driven and trace access driven Cache attacks. In timing driven attack, the adversary can use the whole encryption time to predict whether two times S-box lookup is Cache hit or miss, thus get the possible or impossible key byte candidates. In access driven attack, the adversary can use a spy process loading a $L1$ Cache size array to clear the Cache before the encryption, then trigger the cipher encrypt operations, after that, the spy process can gather the accessed Cache sets of the encryption process by measuring the time to reload each Cache block size array element. Combing the plaintext and ciphertext, the adversary can get the possible or impossible candidates for the encryption key. In trace driven attack, the adversary can gathered every S-box lookup Cache hit or miss sequence, combing plaintext and ciphertext to predict encrypt key. Note that it was just the frequent S-box lookup operations leading to Cache timing attacks.

(2) Differential side channel attack (DSCA)

In differential side channel attack, the attacker gets several power consumption or electromagnetic emanation curves. As the S-box dictionary is known to the attacker, the adversary first divides the key search space to several bytes, then tries every possible value of each byte to predict one or bits of the S-box lookup results and get the possible hamming weight or distance, combing the real power or electromagnetic curves which is affected by the hamming weight or distance. Then, the correct key byte can always have high coefficient than wrong key bytes, so the S-box lookups can also be used in DSCA to recover encryption key. In fact, beside S-box lookup operation, any operations in encryption with strong non-linear feature can be used in DSCA to recover encryption key.

(3) Template attack (TA)

In template attack, the attacker first construct a unique distinguisher for each key byte candidates from the measured signal curves by a controlled encrypt equipment, then, he tried to gather the leaked signal of the target encrypt equipment, using information-theoretic or signal theoretical methods to compute the coefficient of each key byte candidates, the most matched template candidate always relates with the correct key byte. It's just the non-linear feature of S-box lookup constructing the nature unique distinguisher for all key byte candidates and accelerating the key recovery efficiency.

(4) Differential fault attack (DFA)

From Section 4 and 5, we can get that the root of the DFA attack on block cipher with *S*-box lies in *S*-box itself. As long as the input differential and output differentia of S-box is known to the attacker, it's very easy to recover the input value of the *S*-box, which can be used for further analyzing and recovering of the encryption key. Current design of *S*-box is not perfect, the differential *S*-box cann't cover all the candidates from 0x00 to 0xff, which leaks information about the input differential once the adversary gets the output differential of SPN structure block ciphers, even if the differential S-box is prefect (cover all the candidates from 0x00 to 0xff), as the output differential and input differential value of the Feistel block ciphers is known to the adversary, it may become more easy to recover the S-box input value by just one fault, which makes DFA attack become more effective. Anyway, S-box makes DFA attacks on block cipher with S-box becomes possible.

From analysis above, we can safely come to the conclusion that there indeed exist great contradictions between traditional cipher design and implementation attacks, but how to solve this problem is still unknown and confused every cryptographist.

8 Conclusion

In this paper we present several improved DFA attacks 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 recover Camellia-128 very efficiently. Besides, our attack model can be adapted into most block ciphers with S-boxes, such as AES, ARIA, CLEFIA, SMS4 etc. Further more, all of the attacks described in this paper have been successfully put into experimental simulations on a personal computer and the experimental results effectively support the analysis and the arguments.

Acknowledgements

The authors would like to thank the anonymous reviewers for many helpful comments and suggestions. The research presented in this paper was supported 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: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg. (1997)
2. Biham, E., Shamir, A. Differential fault analysis of secret key cryptosystem. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg. (1997)
3. Biehl, I., Meyer, B., Muller, V. Differential fault analysis on elliptic curve cryptosystems. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 131–146. Springer, Heidelberg. (2000)
4. Hemme, L.: A differential fault attack against early rounds of (Triple-) DES. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 254–267. Springer, Heidelberg. (2004)
5. Blomer, J., Seifert, J.P.: Fault based cryptanalysis of the Advanced Encryption Standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg. (2003)
6. Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on AES. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer, Heidelberg. (2003)
7. Piret, G., Quisquater, J.J. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg. (2003)
8. Debdeep Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. In: B. Preneel (eds.) AFRICACRYPT 2009, LNCS 5580, pp. 421–434. (2009)
9. Michael Tunstall, Debdeep Mukhopadhyay. Differential Fault Analysis of the Advanced Encryption Standard using a single Fault. Cryptology ePrint Archive, http://eprint.iacr.org/2009/575.(2009)
10. Dhiman Saha, Debdeep Mukhopadhyay, Dipanwita RoyChowdhury. A Diagonal Fault Attack on the Advanced Encryption Standard. Cryptology ePrint Archive, http://eprint.iacr.org/2009/581.(2009)
11. 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)
12. Xin-jie ZHAO, Tao WANG. An Improved Differential Fault Attack on Camellia, Cryptology ePrint Archive, http://eprint.iacr.org/2009/585. (2009)
13. Wei LI, Da-wu GU, Juan-ru LI. Differential fault analysis on the ARIA algorithm. Information Sciences. Elsevier Inc. pp.3727－3737. (2008)

14. Hua CHEN, Wen-ling WU, and Deng-guo FENG. Differential Fault Analysis on CLEFIA. In S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 284–295. Springer Heidelberg. (2007)
15. Junko Takahashi and ToshinoriFukunaga. Improved Differential Fault Analysis on CLEFIA. Proceedings of the 2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC, IEEE Computer Society, pp 25-34. (2008)
16. Hoch, J.J., Shamir, A.: Fault analysis of stream ciphers. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 240–253. Springer, Heidelberg. (2004)
17. Biham, E., Granboulan, L., Nguyn, P.Q.: Impossible fault analysis of RC4 and differential fault analysis of RC4. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 359–367. Springer, Heidelberg. (2005)
18. M. Hojsik and B. Rudolf. “Floating fault analysis of Trivium,” In: D.R. Chowdhury, V. Rijmen, and A. Das (eds.) INDOCRYPT 2008. LNCS, Heidelberg,Springer,2008,vol. 5365, pp. 239–250. (2008)
19. HU Yupu, GAO Juntao and Liu Qing. Hard Fault Analysis of Trivium. Cryptology ePrint Archive, http://eprint.iacr.org/2009/333. (2009)
20. K. Aoki, T. Ichikawa, M. Kansa, M. Matsui, S. Moriai, Nakajima, and T. Tokita, “Specification of Camellia - a 128-bit Block Cipher”, http://www.cosic.esat.kuleuven.be/nessie/workshop/submissions. (2000)
21. C.Giraud, H.Thiebeauld, “A survey on fault attacks”. Proceeding of 6th International Conference on Smart Card Research and Advanced Applications(CARDIS’O4), Toulouse,France, pp. 22—27. (2004)