Improved Differential Fault Analysis on CLEFIA

Junko Takahashi and Toshinori Fukunaga
NTT Information Sharing Platform Laboratories
Nippon Telegraph and Telephone Corporation
3-9-1 Midori-cho, Musashino-shi, Tokyo, 180-8585, JAPAN
{takahashi.junko, fukunaga.toshinori}@lab.ntt.co.jp

Abstract

We propose a more efficient differential fault analysis (DFA) attack on CLEFIA, the 128-bit blockcipher developed by Sony Corporation in 2007. In the previous study, the most efficient DFA attack on CLEFIA with a 128-bit key uses approximately 18 pairs of correct and faulty ciphertexts. We develop a new attack method and show that only 2 pairs of correct and faulty ciphertexts are needed to retrieve the 128-bit key. The proposed attack uses a characteristic of the CLEFIA algorithm, a four-branch generalized Feistel structure with four 32-bit data lines. The simulation results of the proposed attack show that it takes less than 1 minute for 74.1% of a total simulation times, and less than 1 hour for 98.1% when using a PC.

1. Introduction

Some attacks have been recently proposed that attempt to recover the secret key embedded in a secure device such as a smart card. Fault Analysis (FA) is one type of such attacks that recovers the secret key embedded in a secure device by injecting the faults during its computation of a cryptographic algorithm.

FA was first proposed by 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. 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. Until now, many researchers proposed DFA on symmetric-key cryptography such as Triple-DES, AES, KHAZAD and IDEA [3, 4, 5, 6, 7, 8]. They improved the attack efficiency compared to previous works.

In 2007, the Sony Corporation proposed the 128-bit blockcipher CLEFIA [9]. CLEFIA was designed to have a generalized Feistel structure with four 32-bit data lines to achieve a small implementation and high speed. Some known attacks against a blockcipher were evaluated with respect to CLEFIA [10, 11, 12]. Recently, Chen *et al.* proposed the first DFA on CLEFIA using approximately 18 pairs of correct and faulty ciphertexts to recover a 128-bit key and 54 pairs to recover 192 and 256-bit keys [13]. After one-byte random fault injection into some rounds, they simply applied the method of DFA for recovering the DES key to CLEFIA. This method is repeating the fault injection at the same location and solving simultaneous equations until the round key is uniquely determined. In their attack, many pairs of correct and faulty ciphertexts are needed to recover the original key compared to the attack against the other blockchiper.

In this paper, we propose a more efficient DFA attack on CLEFIA compared to the previous DFA attack [13]. The proposed attack requires 2 pairs of correct and faulty ciphertexts to obtain an entire 128-bit key with feasible calculation time. We take advantage of the characteristic structure of CLEFIA, which consists of a generalized Feistel structure with four 32-bit data lines. We propose a new method for obtaining the round keys, which is deducing the candidates for the round keys. We conduct a simulation on a computer to estimate the calculation time. The results show that we can obtain the 128-bit key less than 1 minute for 74.1% of a total simulation times and less than 1 hour for 98.1%.

The reminder of the paper is organized as follows. Notations are defined in Section 2. In Section 3, we briefly review the CLEFIA algorithm. We describe the previous study in Section 4. The proposed attack method and the DFA attack on CLEFIA are described in Sections 5 and 6. The simulation results are shown in Section 7. Finally, we present our conclusions in Section 8.

978-0-7695-3314-8/08 $25.00 © 2008 IEEE
DOI 10.1109/FDTC.2008.14

2. Notations

This section describes the mathematical notations, conventions, and symbols used throughout this paper.

$a_{(n)}$:	n denotes the bit length of a
$a \mid b$:	Concatenation
$a \leftarrow b$:	Updating a value of a by a value of b
$a \oplus b$:	Bitwise exclusive-OR. Addition in GF(2^n)
$\langle a \rangle$:	A set of a
$\langle a, b \rangle$:	A set comprising a combination of a and b
$\lvert\langle a \rangle\rvert$:	Size of $\langle a \rangle$

3. CLEFIA algorithm

We briefly review the CLEFIA encryption algorithm employing a 128-bit key. The decryption, other cases of 192 and 256-bit keys are given in the specification of CLEFIA[14].

Structure The overall structure of the CLEFIA encryption with a 128-bit key is shown in Fig. 1. CLEFIA uses a four-branch generalized Feistel network and the bit width of each data path is 32 bits. The number of rounds depends on the length of the key and it is 18 for the 128-bit key. Each round has two kinds of F-functions, F_0 and F_1.

F-functions The construction of the F-functions is shown in Fig. 2. Both F-functions uses two kinds of nonlinear eight-bit S-boxes, S_0 and S_1 in different order. Each F-function uses different diffusion matrices, M_0 and M_1.

Key scheduling Let K the key and L be an intermediate key, the key scheduling part consists of the following two steps,

1. Generating L from K.
   The intermediate key L is generating by applying 4-branch 12-round generalized Feistel network which takes twenty-four 32-bits constant values as round keys, and K as an input.
2. Expanding K and L (Generating whitening keys and round keys).
   Then, K and L are used to generate the whitening key and the round keys. The DoubleSwap function illustrated in Fig.3 is used at this point.

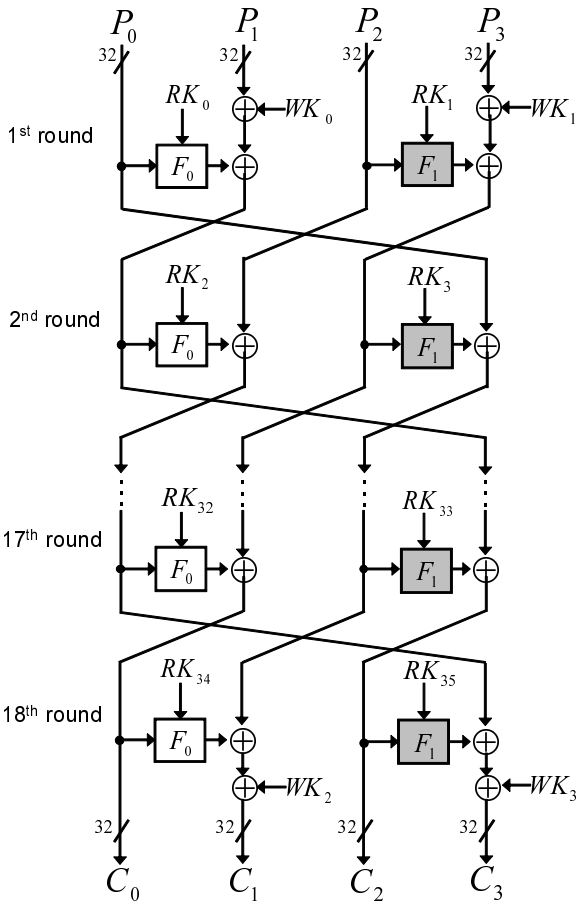

Figure 1. Structure of CLEFIA encryption algorithm with a 128-bit key

4. Previous study

The first DFA attack on CLEFIA was proposed at ICICS 2007 [13]. This attack employs the method, which is repeatedly injecting faults and solving simultaneous equations for the S-boxes until a round key is uniquely determined. The original key can be retrieved using some determinate round keys. This attack does not require a brute-force search.

Here, we show the method of retrieving the round keys used in [13]. We can retrieve 8 bits of a round key, $k_{(8)}$, from two sets of two inputs of the F-function $x_{a(8)}, x_{b(8)}$ and the output difference of the S-box $\delta z_{ab(8)}$, that is, ($x_{a(8)}$, $x_{b(8)}$, $\delta z_{ab(8)}$) and ($x_{c(8)}$, $x_{d(8)}$, $\delta z_{cd(8)}$) by employing the following algorithm.

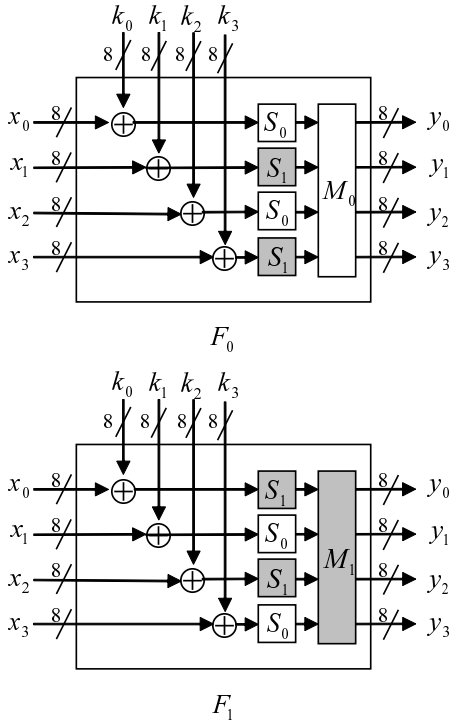

Figure 2. F-functions

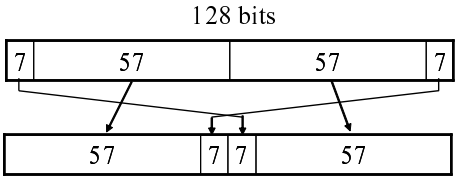

Figure 3. DoubleSwap Function

Algorithm 1: Deduce $k_{(8)}$

Input: $x_{a(8)}, x_{b(8)}, \delta z_{ab(8)}$
$x_{c(8)}, x_{d(8)}, \delta z_{cd(8)}$
Output: $k_{(8)}$

Solve the following simultaneous equations,
$S[k_{(8)} \oplus x_{a(8)}] \oplus S[k_{(8)} \oplus x_{b(8)}] \oplus \delta z_{ab(8)}$=0
$S[k_{(8)} \oplus x_{c(8)}] \oplus S[k_{(8)} \oplus x_{d(8)}] \oplus \delta z_{cd(8)}$=0

if $k_{(8)}$ is uniquely determined
return $k_{(8)}$
else
Error

A S-box, $S[\ ]$, should be appropriately chosen from S_0 or S_1. The basic attack procedure employing Algorithm 1 is described as follows.

1. Calculate the correct ciphertext.
2. 32 bits of the input of F_0-function (F_1-function) are corrupted by fault injection more than once, and some faulty ciphertexts are obtained.
3. Calculate $\Delta Z_{ab(32)}$ from the output difference of the F-function $\Delta Y_{ab(32)}$, that is, $\Delta Z_{ab(32)} = M^{-1}(\Delta Y_{ab(32)})$, which uses the appropriate $M^{-1}(\ )$.
4. Apply Algorithm 1 to each S-box of the F-function until 8 bits of a round key are uniquely determined. After 32 bits of a round key, $RK_{(32)}$, are uniquely determined, proceed the next step.
5. Repeat the fault injection at the previous rounds and apply Algorithm 1 until 32 bits of a round key are uniquely determined.
6. Repeat injecting the faults and applying Algorithm 1 until all round keys, needed to obtain the original key, can be retrieved.

In [13], the authors insist that an average of 3 pairs of correct and faulty ciphertexts required to determine uniquely a round key. Fault injections into a total of three rounds are required to retrieve the 128-bit key, nine rounds are required to retrieve the 192 and 256-bit keys. Therefore, they showed that 18 pairs on average are required to retrieve the 128-bit key, and 54 pairs on average are required to retrieve the 192 and 256-bit keys.

5. Proposed attack method

In this section, we describe idea of proposed attack method, CLEFIA S-box analysis and proposed method for retrieving the round keys.

5.1. Idea of proposed attack method

Reducing the number of fault injection points

According to [13], in order to obtain six round keys to retrieve the 128-bit key, the attacker must inject faults into a total of six locations of the F-function inputs from 16^{th} round to 18^{th} round.

However, we find that a fault corrupts the intermediate values of the fault-injection round and the subsequent rounds. For example, when a fault corrupts the input of the F_0-function in the $(r-2)^{\text{th}}$ round, it also corrupts the inputs of the F_0-functions in the $(r-1)^{\text{th}}$ and r^{th} rounds. Therefore, the faulty ciphertext obtained by a fault injection can be used to deduce the round

keys for a maximum of three F-functions. This is a maximum number of the F-functions which can solve simultaneous equations for retrieving the round keys.

Proposed method for retrieving the round keys

According to [13], they employed a method that deduce 8 bits of a round key and it is repeated until they are uniquely determined by solving simultaneous equations for the S-boxes. The original key can be retrieved from the determinate round keys.

We propose a method that deduces the candidates for 8 bits of round keys by solving the equation for a S-box. The original key can be retrieved from the candidate for the round keys. Then, a brute-force search is needed to evaluate whether or not the candidates for the round keys are correct.

5.2. CLEFIA S-box analysis

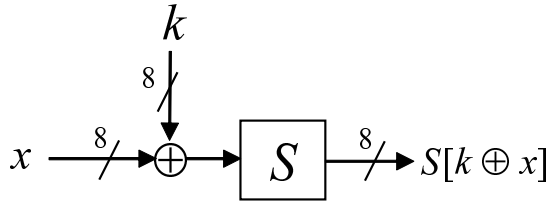

Figure 4. One-byte S-box model

We describe an analysis of S-box using the proposed method described in Section 5.3.

Let us consider the simple one byte S-box model shown in Fig. 4. When we know a pair of inputs $x_{a(8)}$ and $x_{b(8)}$, and know output difference $\delta_{(8)}$, we can obtain a set of unknown key candidates, $\langle k_{(8)} \rangle$, by solving the following equation.

$$S[k_{(8)} \oplus x_{a(8)}] \oplus S[k_{(8)} \oplus x_{b(8)}] \oplus \delta_{(8)} = 0$$

The size of key candidate $|\langle k_{(8)} \rangle|$ depends on $x_{a(8)}$, $x_{b(8)}$, and $\delta_{(8)}$, and the structure of the S-box.

In CLEFIA, two kinds of S-boxes (S_0/S_1) are used in the F-functions. We examined $|\langle k_{(8)} \rangle|$ of each S_0 and S_1 for all combinations of $x_{a(8)}$, $x_{b(8)}$ and $\delta_{(8)}$. The results are shown in Table 1. In the table, the following symbols are used.

N :	Number of combinations $(x_{a(8)}, x_{b(8)}, \delta_{(8)})$
P :	Probability for all combinations
$P_{\|\langle k \rangle\| \neq 0}$:	Probability under $\|\langle k_{(8)} \rangle\| \neq 0$
$E(\|\langle k \rangle\|)$:	Expected value of $\|\langle k_{(8)} \rangle\|$ under $\|\langle k_{(8)} \rangle\| \neq 0$

Table 1. Statistics of the S-boxes

for S_0

$\|\langle k \rangle\|$	N	P	$P_{\|\langle k \rangle\| \neq 0}$	$E(\|\langle k \rangle\|)$
0	10,245,376	0.611	-	-
2	4,992,256	0.298	0.764	1.529
4	1,289,472	0.077	0.197	0.790
6	217,088	0.013	0.033	0.199
8	30,464	0.002	0.005	0.037
10	2,304	0.000	0.000	0.004
256	256	0.000	0.000	0.008
Total	16,777,216 $(= 2^{24})$	1	1	2.257 $(= 2^{1.36})$

for S_1

$\|\langle k \rangle\|$	N	P	$P_{\|\langle k \rangle\| \neq 0}$	$E(\|\langle k \rangle\|)$
0	8,486,400	0.506	-	-
2	8,225,280	0.490	0.992	1.984
4	65,280	0.004	0.008	0.031
256	256	0.000	0.000	0.010
Total	16,777,216 $(= 2^{24})$	1	1	2.024 $(= 2^{1.02})$

The results suggest that there is no key candidate in 61.1% of the combinations of $(x_{a(8)}, x_{b(8)}, \delta_{(8)})$ for S_0, and in 50.6% of those for S_1. When there is a key candidate, its expected size is $2^{1.36}$ for S_0 and $2^{1.02}$ for S_1.

5.3. Proposed method for retrieving keys

We propose an algorithm for retrieving a round key, $RK_{(32)}$, used in each F-function from its known (or guessed) input and output.

From the construction of the F-function and the results given in Section 5.2, we propose the following algorithm for deducing the candidates for a round key, $\langle RK_{(32)} \rangle$, from two inputs, $X_{a(32)}, X_{b(32)}$, and the output difference of F-function, $\Delta Y_{ab(32)}$.

Algorithm 2: Deduce the candidates for $RK_{(32)}$

Input: $X_{a(32)} = \{x_{a0(8)}|x_{a1(8)}|x_{a2(8)}|x_{a3(8)}\}$
$X_{b(32)} = \{x_{b0(8)}|x_{b1(8)}|x_{b2(8)}|x_{b3(8)}\}$
$\Delta Y_{ab(32)}$
Output: $\langle RK_{(32)}\rangle$

$f(x_a, x_b, \delta, k) = S[k \oplus x_a] \oplus S[k \oplus x_b] \oplus \delta$
$\{\delta_{0(8)}|\delta_{1(8)}|\delta_{2(8)}|\delta_{3(8)}\} \leftarrow M^{-1}(\Delta Y_{ab(32)})$

for $i = 0$ to 3 do
  $\langle k_i\rangle \leftarrow$ A set of solution $f(x_{ai}, x_{bi}, \delta_i, k_i) = 0$ for k_i

if all $\langle k_i\rangle$ has any solution
  $\langle RK_{(32)}\rangle \leftarrow \{\langle k_0\rangle|\langle k_1\rangle|\langle k_2\rangle|\langle k_3\rangle\}$
else
  $\langle RK_{(32)}\rangle \leftarrow \phi$

return $\langle RK_{(32)}\rangle$

Depending on the application of this algorithm to F_0 or F_1, $S[\,]$ should be appropriately chosen from S_0 or S_1. Similarly, $M^{-1}[\,]$ should be appropriately chosen from M_0^{-1} or M_1^{-1}. Here, M_0^{-1} (M_1^{-1}) is the inverse matrix of diffusion matrix M_0 (M_1). Since M_0 (M_1) is linear function, $M_0^{-1}(M_1^{-1})$ is easily obtained. In fact, $M_0^{-1} = M_0$ ($M_1^{-1} = M_1$).

From the results in Section 5.2, we can calculate the probability whether or not a round key candidate exists and the expected size of a round key candidate for all combinations of $X_{a(32)}$, $X_{b(32)}$, and $\Delta Y_{ab(32)}$.

- The probability of $\langle RK_{(32)}\rangle \neq \phi$ is

$$(1 - 0.611)^2 \cdot (1 - 0.506)^2 = 0.037.$$

- The probability of $\langle RK_{(32)}\rangle = \phi$ is

$$1 - 0.037 = 0.963.$$

- The expected value of $|\langle RK_{(32)}\rangle|$ when a round key candidate exists is

$$2.257^2 \cdot 2.024^2 = 27.02 = 2^{4.76}.$$

5.4. Characteristic of Algorithm 2

From the results in Section 5.3, we find the following characteristics when Algorithm 2 is applied to the F-functions.

When we know (or guess) a combination of $X_{a(32)}$, $X_{b(32)}$, and $\Delta Y_{ab(32)}$ of the F-function, we can obtain the following information concerning a set of round key candidates, $\langle RK_{(32)}\rangle$, and a combination of $X_{a(32)}$, $X_{b(32)}$, and $\Delta Y_{ab(32)}$ from the results in Section 5.3.

- When we apply Algorithm 2 to certain values of ($X_{a(32)}$, $X_{b(32)}$, $\Delta Y_{ab(32)}$), we can deduce $\langle RK_{(32)}\rangle$, the expected size of which is $2^{4.76}$.
- When we apply Algorithm 2 to the guessed values of ($X_{a(32)}$, $X_{b(32)}$, $\Delta Y_{ab(32)}$),
  - with a 3.7% probability, we can deduce $\langle RK_{(32)}\rangle$, the expected size of which is $2^{4.76}$ for the guessed values of ($X_{a(32)}$, $X_{b(32)}$, $\Delta Y_{ab(32)}$).
  - with a 96.3% probability, we know that the guessed values of ($X_{a(32)}$, $X_{b(32)}$, $\Delta Y_{ab(32)}$) are incorrect. When a combination of values of ($X_{a(32)}$, $X_{b(32)}$, $\Delta Y_{ab(32)}$) are calculated from the other round key candidates, the candidates of the other round keys are incorrect. This means that we can decrease the size of the other round key candidates.

6. Proposed DFA attack on CLEFIA

In this section, we describe the basic attack procedure, attack assumptions and the detail attack procedure using the proposed method.

6.1. Basic attack procedure

The basic attack procedure is described as follows.

- Inject faults and apply Algorithm 2 to two F-functions in the last round and obtain its round-key candidates. Then, the space of the two round-key candidates is $2^{9.51}\{= 2^{4.76} \cdot 2^{4.76}\}$.
- Apply Algorithm 2 to two F-functions in the penultimate round and obtain its round-key candidates using the round-key candidates of the last round key. In this case, most elements of a set of input data of Algorithm 2 are incorrect. This causes a decrease in the size of a set of round-key candidates in the last round. Then, the space of two F-functions is $2^{9.51}\{= (2^{4.76} \cdot (2^{4.76} \cdot 0.037))^2\}$.
- Repeat the application Algorithm 2 to the F-functions of the third to the last round. Then, the space of two F-functions is $2^{19.02}\{= 2^{9.51} \cdot 2^{9.51}\}$.

6.2. Attack assumptions

The attack assumptions are described as follows.

- The attack is against CLEFIA with a 128-bit key
- The attacker can obtain a pair of correct and faulty ciphertexts calculated from the same plaintext and key.

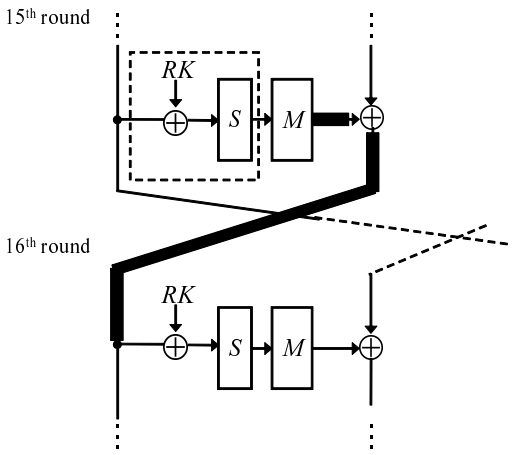

Figure 5. Fault injection region

- The attacker does not need to know the value of the plaintext.
- The attacker does not know the value of the faults. He knows the fault injection area.
- A total of four bytes of inputs in the 16^{th} round are randomly corrupted during one encryption calculation.

  To satisfy the above conditions, the attacker can choose either fault injection point.

  - any bit in any byte is corrupted before the diffusion matrix M, that is, in a dot-line region in the 15^{th} round as shown in Fig. 5
  - a total of four bytes are corrupted after the diffusion matrix M, that is, in the heavy line shown in the 15^{th} round or 16^{th} round in Fig. 5.

Considering the general DFA attack on the target device such as a smart card, the above assumptions are more realistic.

6.3. Detailed attack procedure

An attack procedure consists of the following three main parts.

- Obtain correct and faulty ciphertexts (Step 0)
- Deduce sets of the round-key candidates (from Step 1a to Step 3b)
- Recover the original key $K_{(128)}$ (from Step 4 to Step 6)

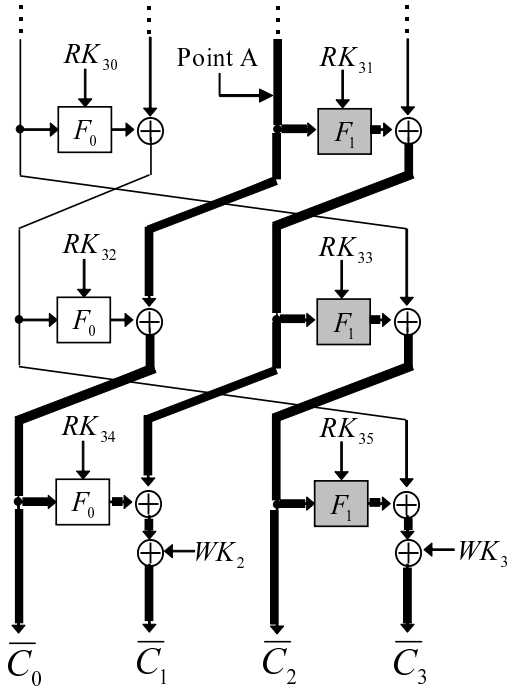

Figure 6. Fault propagation when 32-bit intermediate values at point A in the 16^{th} round are corrupted

From Step 1a to Step 3b, the order of Step a and Step b can be interchanged.

When a total of four bytes are corrupted at point A (point B) in the 16^{th} round, faults are propagated, as shown in Fig. 6 (Fig. 7).

Here, we describe each step in details.

Step 0. Obtain correct and faulty ciphertexts Randomly select a plaintext, and obtain one correct ciphertext, $C_{(128)}$, and two faulty ciphertexts, $\overline{C}_{(128)}$, $\overline{\overline{C}}_{(128)}$, where

$$C_{(128)} = \{C_{0(32)}|C_{1(32)}|C_{2(32)}|C_{3(32)}\},$$
$$\overline{C}_{(128)} = \{\overline{C}_{0(32)}|\overline{C}_{1(32)}|\overline{C}_{2(32)}|\overline{C}_{3(32)}\}, \text{ and}$$
$$\overline{\overline{C}}_{(128)} = \{\overline{\overline{C}}_{0(32)}|\overline{\overline{C}}_{1(32)}|\overline{\overline{C}}_{2(32)}|\overline{\overline{C}}_{3(32)}\}.$$

Faulty ciphertext $\overline{C}_{(128)}$ is caused by corruption at point A, and $\overline{\overline{C}}_{(128)}$ is caused by corruption at point B.

Step 1a. Deduce $\langle \boldsymbol{RK_{35}} \rangle$ Since two inputs of the F_1-function and the output difference of the F_1-function can be described by correct and faulty ciphertexts and we know them, we can obtain $\langle RK_{35} \rangle$ by applying Algorithm 1 to the

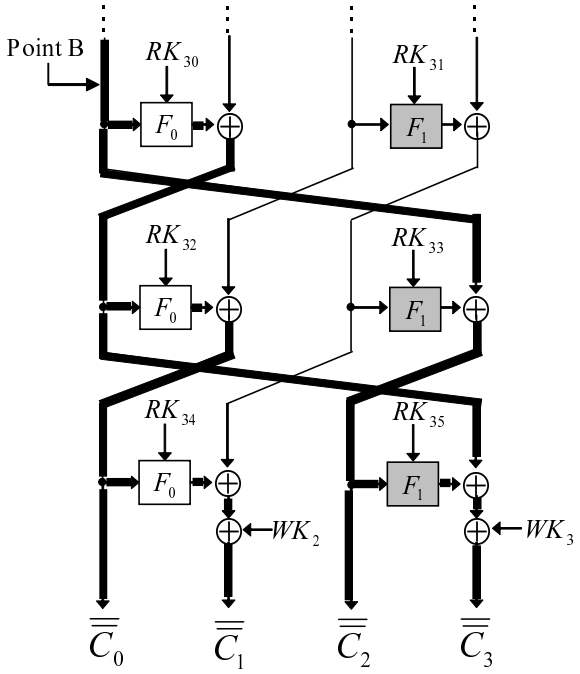

Figure 7. Fault propagation when 32-bit intermediate values at point B in the 16^{th} round are corrupted

F_1-function in the 18^{th} round. When we apply Algorithm 1, we use the following values:

$$X_a = C_2,$$
$$X_b = \overline{C}_2, \text{ and}$$
$$\Delta Y_{ab} = C_3 \oplus \overline{C}_3.$$

Step 1b. Deduce $\langle \boldsymbol{RK_{34}} \rangle$ As Step 1a, we apply Algorithm 1 to the F_0-function in the 18^{th} round. Then, we can obtain $\langle RK_{34} \rangle$ using the following values:

$$X_a = C_0,$$
$$X_b = \overline{\overline{C}}_0, \text{ and}$$
$$\Delta Y_{ab} = C_1 \oplus \overline{\overline{C}}_1.$$

Step 2a. Deduce $\langle \boldsymbol{RK_{33} \oplus WK_2, RK_{34}} \rangle$ As Step 1, we apply Algorithm 1 to the F_1-function in the 17^{th} round for all $rk_{34} \in \langle RK_{34} \rangle$ deduced in Step 1b where rk_{34} is the candidate for RK_{34}. Then, we can obtain $\langle RK_{33} \oplus WK_2, RK_{34} \rangle$ using the following values:

$$X_a = F_0(C_0, rk_{34}) \oplus C_1,$$
$$X_b = F_0\left(\overline{C}_0, rk_{34}\right) \oplus \overline{C}_1, \text{ and}$$
$$\Delta Y_{ab} = C_2 \oplus \overline{C}_2.$$

Step 2b. Deduce $\langle \boldsymbol{RK_{32} \oplus WK_3, RK_{35}} \rangle$ We apply Algorithm 1 to the F_0-function in the 17^{th} round for all $rk_{35} \in \langle RK_{35} \rangle$ deduced in Step 1a. Then, we can obtain $\langle RK_{32} \oplus WK_3, RK_{35} \rangle$ using the following values:

$$X_a = F_1\left(C_2, rk_{35}\right) \oplus C_3,$$
$$X_b = F_1\left(\overline{\overline{C}}_2, rk_{35}\right) \oplus \overline{\overline{C}}_3, \text{ and}$$
$$\Delta Y_{ab} = C_0 \oplus \overline{\overline{C}}_0.$$

Step 3a. Deduce $\langle \boldsymbol{RK_{31}, RK_{32} \oplus WK_3, RK_{34}, RK_{35}} \rangle$ As Step 1, we apply Algorithm 1 to the F_1-function in the 16^{th} round for all $rk_{32} \oplus wk_3$, $rk_{35} \in \langle RK_{32} \oplus WK_3, RK_{35} \rangle$ and $rk_{34} \in \langle RK_{34} \rangle$ deduced in Steps 1b and 2b. Then, we can obtain $\langle RK_{31}, RK_{32} \oplus WK_3, RK_{34}, RK_{35} \rangle$ using the following values:

$$X_a = F_0\left(F_1(C_2, rk_{35}) \oplus C_3, rk_{32} \oplus wk_3\right) \oplus C_0,$$
$$X_b = F_0\left(F_1\left(\overline{C}_2, rk_{35}\right) \oplus \overline{C}_3, rk_{32} \oplus wk_3\right) \oplus \overline{C}_0, \text{ and}$$
$$\Delta Y_{ab} = F_0(C_0, rk_{34}) \oplus F_0\left(\overline{C}_0, rk_{34}\right) \oplus C_1 \oplus \overline{C}_1.$$

Step 3b. Deduce $\langle \boldsymbol{RK_{30}, RK_{33} \oplus WK_2, RK_{34}, RK_{35}} \rangle$ We apply Algorithm 1 to the F_0-function in the 16^{th} round for all $rk_{33} \oplus wk_2$, $rk_{34} \in \langle\ RK_{33} \oplus WK_2, RK_{34} \rangle$ and $rk_{35} \in \langle RK_{35} \rangle$ deduced in Steps 1a and 2a. Then, we can obtain $\langle RK_{30}, RK_{33} \oplus WK_2, RK_{34}, RK_{35} \rangle$ using the following values:

$$X_a = F_1\left(F_0(C_0, rk_{34}) \oplus C_1, rk_{33} \oplus wk_2\right) \oplus C_2,$$
$$X_b = F_1\left(F_0\left(\overline{\overline{C}}_0, rk_{34}\right) \oplus \overline{\overline{C}}_1, rk_{33} \oplus wk_2\right) \oplus \overline{\overline{C}}_2, \text{ and}$$
$$\Delta Y_{ab} = F_1(C_2, rk_{35}) \oplus F_1\left(\overline{\overline{C}}_2, rk_{35}\right) \oplus C_3 \oplus \overline{\overline{C}}_3.$$

Step 4. Deduce $\langle \boldsymbol{RK_{30}, RK_{31}, RK_{32} \oplus WK_3, RK_{33} \oplus WK_2, RK_{34}, RK_{35}} \rangle$ From Step 1a to Step 3b, we can obtain two sets of round-key candidates:

$$\langle RK_{30}, RK_{33} \oplus WK_2, RK_{34}, RK_{35} \rangle,$$
$$\langle RK_{31}, RK_{32} \oplus WK_3, RK_{34}, RK_{35} \rangle.$$

From the above two sets, we obtain a set of $\langle RK_{30}, RK_{31}, RK_{32} \oplus WK_3, RK_{33} \oplus WK_2, RK_{34}, RK_{35} \rangle$ by eliminating sets in which rk_{34} or rk_{35} are not equal in the two sets.

Step 5. Deduce $\langle RK_{32}, RK_{33}, RK_{34}, RK_{35}, WK_2, WK_3\rangle$ Applying the inverse function of the DoubleSwap function, we can calculate $\langle RK_{32}, RK_{33}, RK_{34}, RK_{35}, WK_2, WK_3\rangle$.

Step 6. Recover secret key $K_{(128)}$ For all rk_{32}, rk_{33}, rk_{34}, rk_{35}, wk_2, $wk_3 \in \langle RK_{32}, RK_{33}, RK_{34}, RK_{35}, WK_2, WK_3\rangle$, we calculate the key candidates $k_{(128)} = \{k_{0(32)}|k_{1(32)}|k_{2(32)}|k_{3(32)}\}$ by applying the inverse function of the DoubleSwap function and $GFN_{4,12}^{-1}$ to rk_{32}, rk_{33}, rk_{34}, and rk_{35}. If $(k_{2(32)}, k_{3(32)})$ of $k_{(128)}$ are equal to (wk_2, wk_3), the candidate $k_{(128)}$ is correct and equal to the original key $K_{(128)}$.

6.4. Size of the round-key candidates

We evaluate the expected value of the size of the round-key candidates, $|\langle RK_{(32)}\rangle|$, at each step from Step 1a to Step 5.

Step 1a. or Step 1b. The expected value of $|\langle RK_{35}\rangle|$ or $|\langle RK_{34}\rangle|$ is $2^{4.76}$.

Step 2a. or Step 2b. At Step 2a or Step 2b, the expected value of $|\langle RK_{35}\rangle|$ or $|\langle RK_{34}\rangle|$ can be redeuced. Then, the expected value of $|\langle RK_{33} \oplus WK_2, RK_{34}\rangle|$ or $|\langle RK_{32} \oplus WK_3, RK_{35}\rangle|$ is $2^{4.76}\{= 2^{4.76} \cdot (0.037 \cdot 2^{4.76})\}$.

Step 3a. or Step 3b. The expected value of $|\langle RK_{31}, RK_{32} \oplus WK_3, RK_{34}, RK_{35}\rangle|$ or $|\langle RK_{30}, RK_{33} \oplus WK_2, RK_{34}, RK_{35}\rangle|$ is $2^{9.51}\{= 2^{4.76} \cdot 2^{4.76}\}$.

Step 4. Since each expected value at Step 3a or Step 3b is $2^{9.51}$, the expected value of $|\langle RK_{30}, RK_{31}, RK_{32} \oplus WK_3, RK_{33} \oplus WK_2, RK_{34}, RK_{35}\rangle|$ is $2^{19.02}(= 2^{9.51} \cdot 2^{9.51})$.

Step 5. Since the expected value of $|\langle RK_{30}, RK_{31}, RK_{32} \oplus WK_3, RK_{33} \oplus WK_2, RK_{34}, RK_{35}\rangle|$ is $2^{19.02}$, the expected value of $|\langle RK_{32}, RK_{33}, RK_{34}, RK_{35}, WK_2, WK_3\rangle|$ is also $2^{19.02}$.

Table 2 gives the expected value of $|\langle RK_{(32)}\rangle|$ at the end of each attack step. At the last step, the expected value of the space of the round-key candidates is $2^{19.02}$. Therefore, we need the 19.02-bit brute-force search to obtain the correct key.

7. Simulation results

In order to verify the proposed attack and evaluate the size of the round-key candidates, we implement the attack in C code and execute it on a Xeon 3.0 GHz PC. In the simulation, we use the plaintext and the key randomly chosen. The histograms of the brute-force search space and calculation time are shown in Fig.8 and Fig.9 for 2,000 samples. As is shown in Fig.8, the highest frequency is achieved by the 20-bit brute-force search. This result agrees with the theoretical results that a 19.02 brute-force search is required. As is shown Fig.9, the total calculation time for retrieving the key is 1 minute for 74.1% of a total simulation times and less than 1 hour for 98.1%.

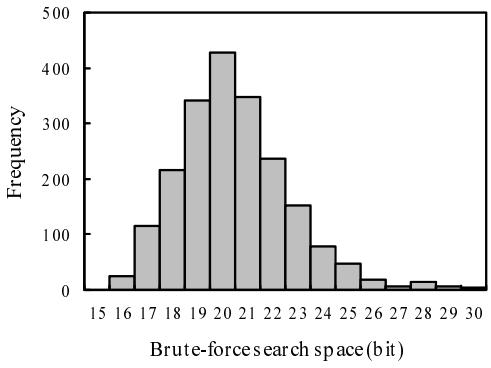

Figure 8. Histogram of the brute-force search space of 2,000 samples.

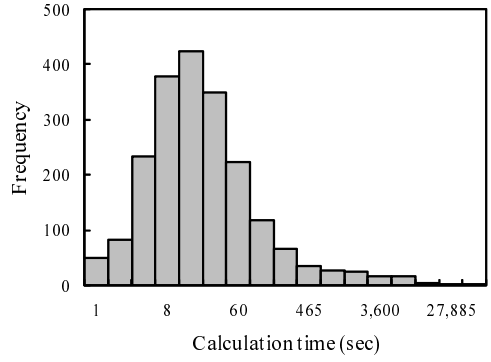

Figure 9. Histogram of the calculation time of 2,000 samples.

8. Conclusions

We propose the most efficient DFA attack on CLEFIA. The proposed attack uses a characteristic of an algorithm of CLEFIA, which is a four-branch generalized Feistel struc-

Table 2. Expected value of $|\langle RK_{(32)} \rangle|$ at end of each attack step

Step	RK_{30}	$RK_{33} \oplus WK_2$	RK_{34}	RK_{35}	$RK_{32} \oplus WK_3$	RK_{31}
1a				$2^{4.76}$		
1b			$2^{4.76}$			
2a		$2^{4.76}$				
2b				$2^{4.76}$		
3a			$2^{9.51}$			
3b		$2^{9.51}$				
4	$2^{19.02}$					

Table 3. Comparison between proposed and previous attacks (for 128-bit key)

	Proposed attack	Chen's attack [13]
Fault model	Random fault model	Random fault model
Method for retrieving key	Retrieving round-key candidates	Retrieving determinate round key
Number of pairs	2^*	$18 - 20^{**}$
Number of fault injection points	2	6
Fault injection round	16^{th}	16^{th}, 17^{th}, 18^{th}
Calculation time†	< 1 min (for 74.1% of all cases) < 1 hour (for 98.1% of all cases)	< 1 sec (for all cases)

* 1 correct and 2 faulty ciphertexts
** 1 correct and 18 faulty ciphertexts
† on Xeon 3.0 GHz PC

ture with four 32-bit data lines. We propose the new method which is deducing the candidates for the round keys and retrieving the original key from the candidates for the round key. As a result, we can reduce the required number of correct and faulty ciphertexts. We show that the entire 128-bit key is obtained using only 2 pairs of correct and faulty ciphertexts. We implement our attack to verify it and evaluate the calculation time. The simulation result shows that it takes less than 1 minute for 74.1% of a total simulation times and less than 1 hour for 98.1%.

A comparison between the proposed and the previous attacks [13] is shown in Table 3. The proposed attack reduces the number of pairs of correct and faulty ciphertexts and fault injection points required to obtain the 128-bit key.

The proposed attack can be extended to 192-bit and 256-bit keys. This topic is for future work.

References

[1] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," EUROCRYPT'97, LNCS 1233, pp.37–51, 1997.

[2] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystem," CRYPTO 1997, LNCS 1294, pp. 513–525, 1997.

[3] L. Hemme, "A Differential Fault Attack Against Early Rounds of (Triple-)DES," CHES 2004, LNCS 3156, pp. 254–267, 2004.

[4] C. Giraud, "DFA on AES," AES 2004, LNCS 3373, pp. 27–41, 2005.

[5] A. Moradi, M. T. M. Shalmani and M. Salmasizadeh, "A Generalized Method of Differential Fault Attack Against AES Cryptosystem," CHES 2006, LNCS 4249, pp. 91–100, 2006.

[6] G. Piret and J.-J. Quisquater, "A Differential Fault Attack Technique against SPN Structure, with Application to the AES and KHAZAD," CHES 2003, LNCS 2779, pp. 77–88, 2003.

[7] J. Takahashi, T. Fukunaga and K. Yamakoshi, "DFA Mechanism on the AES Key Schedule," FDTC 2007, *IEEE-Computer Society*, pp. 62–72, 2007.

[8] C. Clavier, B. Gierlichs, and I. Verbauwhede, “Fault Analysis Study of IDEA”, CT-RSA 2008, LNCS 4964, pp. 274–287, 2008.

[9] T. Shirai, K. Shibutani, T. Akishita, S. Moriai and T. Iwata, “The 128-Bit Blockcipher CLEFIA (Extended Abstract),” FSE 2007, LNCS 4593, pp. 181–195, 2007.

[10] Sony Corporation, “The 128-bit Blockcipher CLEFIA Security and Performance Evaluations (Revision 1.0, June 1, 2007.),” http://www.sony.net/Products/clefia/.

[11] Y. Tsunoo, E. Tsujihara, M. Shigeri, T. Saito, T. Suzuki and H. Kubo, “Impossible Differential Cryptanalysis of CLEFIA,” FSE2008, 2008.

[12] W. Wang and X. Wang, “Improved Impossible Differential Cryptanalysis of CLEFIA,” IACR ePrint archive: Report 2007/466, 2007.

[13] H. Chen, W. Wu and D. Feng, “Differential Fault Analysis on CLEFIA,” ICICS 2007, LNCS 4861, pp. 284–295, 2007.

[14] Sony Corporation, “The 128-bit Blockcipher CLEFIA Algorithm Specification (Revision 1.0, June 1, 2007),” http://www.sony.net/Products/clefia/.