Differential Fault Analysis on CLEFIA

Hua Chen, Wenling Wu, and Dengguo Feng

State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, P.R. China
{chenhua,wwl,feng}@is.iscas.ac.cn

Abstract. CLEFIA is a new 128-bit block cipher proposed by SONY corporation recently. The fundamental structure of CLEFIA is a generalized Feistel structure consisting of 4 data lines. In this paper, the strength of CLEFIA against the differential fault attack is explored. Our attack adopts the byte-oriented model of random faults. Through inducing randomly one byte fault in one round, four bytes of faults can be simultaneously obtained in the next round, which can efficiently reduce the total induce times in the attack. After attacking the last several rounds' encryptions, the original secret key can be recovered based on some analysis of the key schedule. The data complexity analysis and experiments show that only about 18 faulty ciphertexts are needed to recover the entire 128-bit secret key and about 54 faulty ciphertexts for 192/256-bit keys.

Keywords: Block Cipher, Generalized Feistel Structure, Differential Fault Attack.

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 the presentation of DFA, many research papers have been published on using this cryptanalysis technique to successfully attack various cryptosystems, including ECC, 3DES, AES, RC4, and so on[3][4][5][6][7][8][9][10][11].

The block cipher CLEFIA was proposed by SONY corporation recently[12]. It is a 128-bit block cipher which supports 128-bit, 192-bit and 256-bit keys. The fundamental structure of CLEFIA is a generalized Feistel structure consisting of 4 data lines. There are two 32-bit F-functions per one round, which respectively use two different S-boxes and two different diffusion matrices. The key scheduling part shares the generalized Feistel structure with the data processing part.

S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 284–295, 2007.

The number of rounds is 18, 22, and 26 for 128-bit, 192-bit, and 256-bit keys, respectively.

In [12], the strength of CLEFIA against some well-known attacks were examined by the designers, including differential cryptanalysis, linear cryptanalysis, impossible differential cryptanalysis, related-key cryptanalysis and so on. However, the differential fault attack was not mentioned.

In this paper, an efficient differential fault attack against CLEFIA is presented. The attack adopts the byte-oriented model of random faults. In the attack, four bytes of faults can be simultaneously obtained in one round by inducing randomly one byte fault in the last round, which can efficiently reduce the total induce times. After obtaining the subkeys in the last several rounds, and through some analysis of the key schedule, the whole secret key can be determined with only 18 faulty ciphertexts on average for 128-bit key and 54 faulty ciphertexts on average for 192 or 256-bit key. The experimental results also verify the facts.

This paper is organized as follows. In Section 2, the basic description of CLEFIA is presented. Then the basic idea of our attack is given in Section 3. Section 4 provides the detailed attacking procedure for different key sizes, the data complexity analysis of our attack and the experimental results through the computer simulation. Finally, the conclusion remarks are presented in section 5.

2 Description of CLEFIA

In this section, the basic description of CLEFIA is presented. Due to the page limitation, only $GFN_{d,r}$, F-functions, encryption function and key scheduling are introduced. The lacking of introducing the other parts of CLEFIA will not affect the description of our attack.

In the following description of CLEFIA, let a_b represent the bit length of a is b, $|$ represent concatenation and ${}^t a$ represent the transposition of a vector a.

Description of $GFN_{d,r}$. CLEFIA uses a 4-branch and an 8-branch Type-2 generalized Feistel network[13]. Denote d-branch r-round generalized Feistel network as $GFN_{d,r}$. In CLEFIA, $GFN_{d,r}$ employs two different 32-bit F-functions F_0 and F_1 whose input/output are defined as follows.

$$F_0, F_1 = \begin{cases} \{0,1\}^{32}, \{0,1\}^{32} \to \{0,1\}^{32} \\ RK_{(32)}, x_{(32)} \mapsto y_{(32)} \end{cases}$$

For d 32-bit input X_i and output Y_i $(0 \le i < d)$, and $dr/2$ 32-bit round keys RK_i $(0 \le i < dr/2)$, $GFN_{d,r}(d = 4, 8)$ are defined as follows.

$$GFN_{4,r} = \begin{cases} \{\{0,1\}^{32}\}^{2r}, \{\{0,1\}^{32}\}^4 \to \{\{0,1\}^{32}\}^4 \\ RK_{0(32)}, ..., RK_{2r-1(32)}, x_{0(32)}, ..., x_{3(32)} \mapsto y_{0(32)}, ..., y_{3(32)} \end{cases}$$

$$GFN_{8,r} = \begin{cases} \{\{0,1\}^{32}\}^{4r}, \{\{0,1\}^{32}\}^8 \to \{\{0,1\}^{32}\}^8 \\ RK_{0(32)}, ..., RK_{4r-1(32)}, x_{0(32)}, ..., x_{7(32)} \mapsto y_{0(32)}, ..., y_{7(32)} \end{cases}$$

The detailed description of $GFN_{4,r}$ is as follows.

Step 1. $T_0|T_1|T_2|T_3 \leftarrow X_0|X_1|X_2|X_3$
Step 2. For $i = 0$ to $r - 1$ do the following:
  Step 2.1 $T_1 = T_1 \oplus F_0(RK_{2i}, T_0)$, $T_3 = T_3 \oplus F_1(RK_{2i+1}, T_2)$
  Step 2.2 $T_0|T_1|T_2|T_3 \leftarrow T_1|T_2|T_3|T_0$
Step 3. $Y_0|Y_1|Y_2|Y_3 \leftarrow T_3|T_0|T_1|T_2$

The description of $GFN_{8,r}$ is similar to $GFN_{4,r}$, and not introduced here.

The inverse function $GFN^{-1}_{d,r}$ are realized by changing the order of RK_i and the direction of word rotation at Step 2.2 and Step 3 of $GFN_{4,r}$.

F-Functions. F-functions $F_0 : (RK_{(32)}, x_{(32)}) \mapsto y_{(32)}$ can be described as follows:

Step 1. $T \leftarrow RK \oplus x$
Step 2. Let $T \leftarrow T_0|T_1|T_2|T_3$, $T_i \in \{0,1\}^8$
  $T_0 = S_0(T_0)$, $T_1 = S_1(T_1)$
  $T_2 = S_0(T_2)$, $T_3 = S_1(T_3)$
Step 3. Let $y \leftarrow y_0|y_1|y_2|y_3$, $y_i \in \{0,1\}^8$
  ${}^t(y_0, y_1, y_2, y_3) = M_0 \, {}^t(T_0, T_1, T_2, T_3)$

F_0 uses two different 8×8 S-boxes S_0 and S_1. S_0 is constructed by combining 4×4 small S-boxes. S_1 is constructed with the inverse transform plus affine operation in finite field. The diffusion matrix M_0 is a 4×4 Hadamard-type matrix. Figure 1 depicts the F-function F_0.

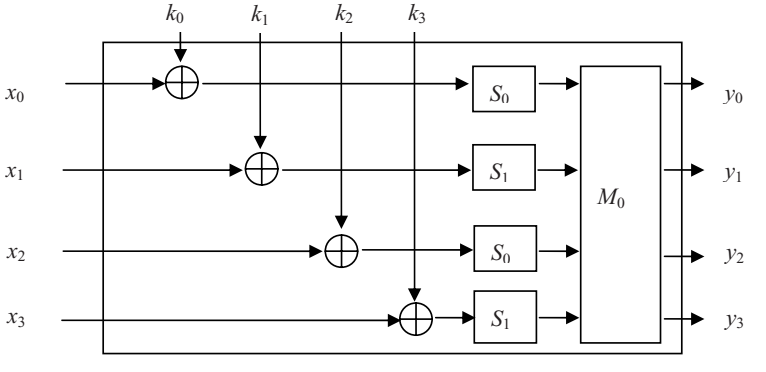

Fig. 1. *F-function* F_0

$F_1 : (RK_{(32)}, x_{(32)}) \rightarrow y_{(32)}$ is similar to F_0 except that, the order of S-boxes is S_1, S_0, S_1, S_0 and M_0 is substituted by M_1. M_1 is also a 4×4 Hadamard-type matrix.

For both M_0 and M_1, the multiplications between matrices and vectors are performed in $GF(2^8)$ defined by the lexicographically first primitive polynomial

$z^8 + z^4 + z^3 + z^2 + 1$. M_0^{-1} and M_1^{-1} respectively represents the inverse matrix of M_0 and M_1.

2.1 Encryption Function

The encryption function of CLEFIA is denoted as ENC_r. Let $P, C \in \{0,1\}^{128}$ be a plaintext and a ciphertext, which can be divided into $P = P_0|P_1|P_2|P_3$ and $C = C_0|C_1|C_2|C_3$, $P_i, C_i \in \{0,1\}^{32}, 0 \leq i \leq 3$. Let $WK_0, WK_1, WK_2, WK_3 \in \{0,1\}^{32}$ be whitening keys and $RK_i \in \{0,1\}^{32} (0 \leq i < 2r)$ be round keys provided by the key scheduling. Then, r round encryption function ENC_r can be described as follows.

$$ENC_r : \begin{cases} \{\{0,1\}^{32}\}^4, \{\{0,1\}^{32}\}^{2r}, \{\{0,1\}^{32}\}^4 \rightarrow \{\{0,1\}^{32}\}^4 \\ WK_{0(32)}, ..., WK_{3(32)}, RK_{0(32)}, ..., RK_{2r-1(32)}, P_{0(32)}, ..., P_{3(32)} \\ \mapsto C_{0(32)}, ..., C_{3(32)} \end{cases}$$

The detailed description is as follows.

Step 1. $T_0|T_1|T_2|T_3 \leftarrow P_0|(P_1 \oplus WK_0)|P_2|(P_3 \oplus WK_1)$
Step 2. $T_0|T_1|T_2|T_3 \leftarrow GFN_{4,r}(RK_0, ..., RK_{2r-1}, T_0, ..., T_3)$
Step 3. $C_0|C_1|C_2|C_3 \leftarrow T_0|(T_1 \oplus WK_2)|T_2|(T_3 \oplus WK_3)$

Figure 2 depicts the encryption function ENC_r.

2.2 Key Scheduling

The key scheduling generates whitening keys $WK_i (0 \leq i < 4)$, and round keys $RK_j (0 \leq j < 2r)$.

Let K be a k-bit key, where k is 128, 192 or 256. The key scheduling is divided into the following two sub-parts.

(1) Generating an intermediate key L from K.
(2) Expanding K and L to generate WK_i and RK_j.

The key scheduling is explained according to the sub-parts.

For the 128-bit key scheduling, the 128-bit intermediate key L is generated by applying $GFN_{4,12}$ which takes twenty-four 32-bit constant values $CON_i^{128}, 0 \leq i < 24$ as round keys and $K = K_0|K_1|K_2|K_3$ as an input. Then K and L are used to generate $WK_i (0 \leq i < 4)$ and $RK_j (0 \leq j < 36)$ in the following steps.

Step 1. $L \leftarrow GFN_{4,12}(CON_0^{(128)}, ..., CON_{23}^{(128)}, K_0, ..., K_3)$
Step 2. $WK_0|WK_1|WK_2|WK_3 \leftarrow K$
Step 3. For $i = 0$ to 8 do the following:
$T \leftarrow L \oplus (CON_{24+4i}^{(128)}|CON_{24+4i+1}^{(128)}|CON_{24+4i+2}^{(128)}|CON_{24+4i+3}^{(128)})$
$L = \sum(L)$
if i is odd. $T = T \oplus K$
$RK_{4i}|RK_{4i+1}|RK_{4i+2}|RK_{4i+3} \leftarrow T$

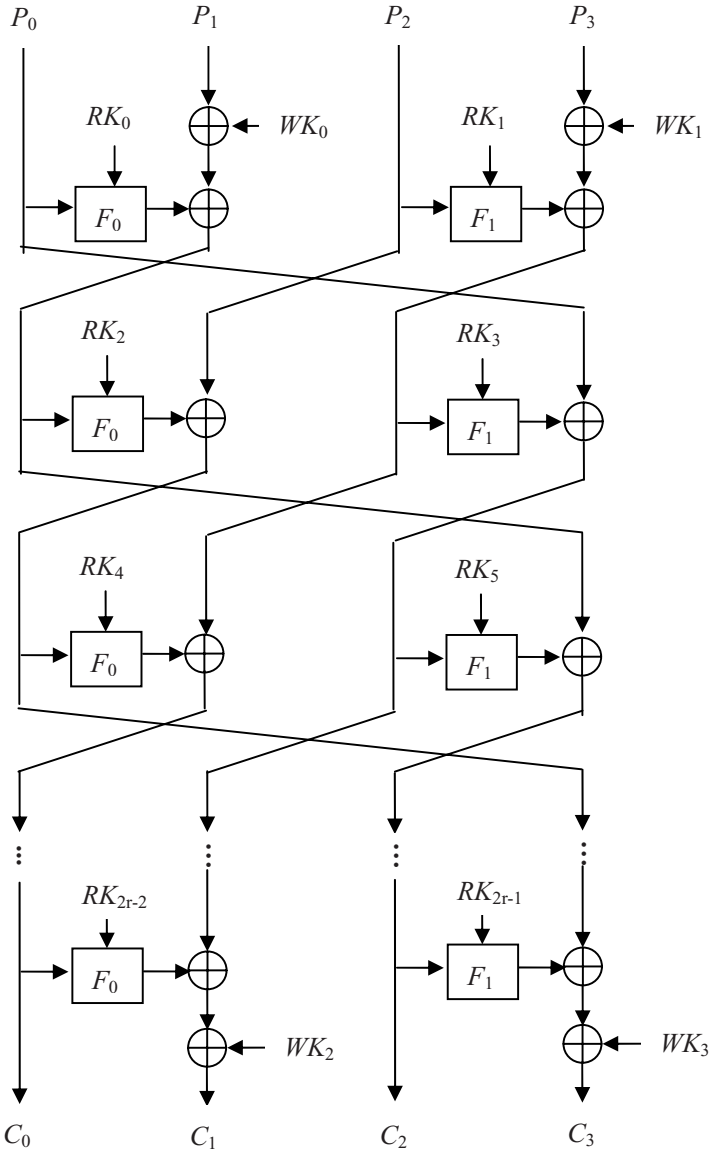

Fig. 2. ENC_r

The *DoubleSwap* function $\sum : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ is defined as follows:

$X_{128} \mapsto Y_{128}$
$Y = X[7-63]X[121-127]X[0-6]X[64-120]$

Figure 3 depicts the $\sum$ function.

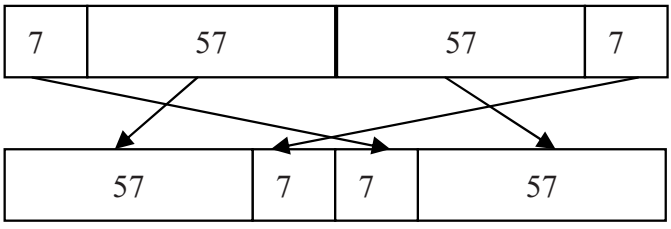

Fig. 3. *Function* $\sum$

The following steps show the 192-bit/256-bit key scheduling and the value of k is respectively set as 192 and 256.

Step 1. Set $k = 192$ or $k = 256$
Step 2. If $k = 192$: $K_L \leftarrow K_0|K_1|K_2|K_3, K_R \leftarrow K_4|K_5|\bar{K}_0|\bar{K}_1$
else if $k = 256$:$K_L \leftarrow K_0|K_1|K_2|K_3, K_R \leftarrow K_4|K_5|K_6|K_7$
Step 3. Let $K_L = K_{L0}|K_{L1}|K_{L2}|K_{L3}$, $K_R = K_{R0}|K_{R1}|K_{R2}|K_{R3}$
$L_L|L_R \leftarrow GFN_{8,10}(CON_0^{(k)}, ..., CON_{39}^{(k)}, K_{L0}, ...K_{L3}, K_{R0}, ..., K_{R3})$
Step 4. $WK_0|WK_1|WK_2|WK_3 \leftarrow K_L \oplus K_R$
Step 5. For $i = 0$ to 10 (if $k = 192$) or 12 (if $k = 256$) do the following:

If $(i \bmod 4) = 0$ or 1:
$T \leftarrow L_L \oplus (CON_{40+4i}^{(k)}|CON_{40+4i+1}^{(k)}|CON_{40+4i+2}^{(k)}|CON_{40+4i+3}^{(k)})$
$L_L = \sum(L_L)$
if i is odd, $T = T \oplus K_R$
else:
$T \leftarrow L_R \oplus (CON_{40+4i}^{(k)}|CON_{40+4i+1}^{(k)}|CON_{40+4i+2}^{(k)}|CON_{40+4i+3}^{(k)})$
$L_R = \sum(L_R)$
if i is odd, $T = T \oplus K_L$
$RK_{4i}|RK_{4i+1}|RK_{4i+2}|RK_{4i+3} \leftarrow T$

3 Basic Idea of Our Attack

3.1 Fault Model and Basic Assumptions

The byte-oriented model of random faults is adopted in our attack and the basic assumptions are as follows.

(1) Only one byte fault can be induced into the register storing the intermediate results. The adversary knows neither the location of the fault nor its concrete value.

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

3.2 Basic Idea of Our Attack

Let r be the round number of CLEFIA algorithm. The basic idea for our differential fault attack on CLEFIA is as follows:

(1) choose randomly a plaintext and obtain the corresponding right ciphertext.

(2) Disturb another encryption of the plaintext until one random byte fault is successfully induced into T_0 of the $(r-1)$-th round, which causes four bytes faults into T_0 of the r-th round, and obtain the corresponding faulty ciphertext. Calculate the candidate values of all the bytes of RK_{2r-2} using differential analysis technique. Repeat the induce procedure until all the bytes of RK_{2r-2} are recovered; Similarly, RK_{2r-1} can be recovered by inducing fault into T_2 of $(r-1)$-th round.

(3) Disturb another encryption of the plaintext until one random byte fault is successfully induced into T_0 of the $(r-2)$-th round, which causes four bytes faults into T_0 of the $(r-1)$-th round, and obtain the corresponding faulty ciphertext. Calculate the candidate values of all the bytes of $RK_{2r-4} \oplus WK_3$ using differential analysis technique. Repeat the induce procedure until all the bytes of $RK_{2r-4} \oplus WK_3$ are recovered; Similarly, $RK_{2r-3} \oplus WK_2$ can be recovered by inducing fault into T_2 of $(r-2)$-th round.

(4) Disturb another encryption of the plaintext until one random byte fault is successfully induced into T_0 of the $(r-3)$-th round, which causes four bytes faults into T_0 of the $(r-2)$-th round, and obtain the corresponding faulty ciphertext. Calculate the candidate values of all the bytes of RK_{2r-6} using differential analysis technique. Repeat the induce procedure until all the bytes of RK_{2r-6} are recovered; Similarly, RK_{2r-5} can be recovered by inducing fault into T_2 of $(r-3)$-th round.

(5) If the key size is 128, jump to step (6); else continue inducing faults according to the similar procedures as step (2)-step (4) until $RK_{2r-8} \oplus WK_2$, $RK_{2r-7} \oplus WK_3$, RK_{2r-10}, RK_{2r-9}, $RK_{2r-12} \oplus WK_3$, $RK_{2r-11} \oplus WK_2$, RK_{2r-14}, RK_{2r-13}, $RK_{2r-16} \oplus WK_2$, $RK_{2r-15} \oplus WK_3$, RK_{2r-18} and RK_{2r-17} are all recovered.

(6) Based on the recovered round keys, analyze the key scheduling of CLEFIA, and deduce the whole secret key K.

4 DFA on CLEFIA

4.1 Notations and Symbols

In order to clearly illustrate the following attacking procedure, some notations and symbols are to be defined.

Firstly, $X_i^j = (x_{i,0}^j, x_{i,1}^j, x_{i,2}^j, x_{i,3}^j)$ and $Y_i^j = (y_{i,0}^j, y_{i,1}^j, y_{i,2}^j, y_{i,3}^j)$, $j \in \{0,1\}$, are respectively defined as the input and output of the S-boxes in F_j of the i-th round. Y_i^j is also the input of M_j of the i-th round. $Z_i^j = (z_{i,0}^j, z_{i,1}^j, z_{i,2}^j, z_{i,3}^j)$, $j \in \{0,1\}$, is defined as the output of M_j of the i-th round.

$\Delta X_i^j = (\Delta x_{i,0}^j, \Delta x_{i,1}^j, \Delta x_{i,2}^j, \Delta x_{i,3}^j)$ and $\Delta Y_i^j = (\Delta y_{i,0}^j, \Delta y_{i,1}^j, \Delta y_{i,2}^j, \Delta y_{i,3}^j)$, $j \in \{0,1\}$, are respectively defined as the input and output differences of the

S-boxes in F_j of the i-th round. $\Delta Z_i^j = (\Delta z_{i,0}^j, \Delta z_{i,1}^j, \Delta z_{i,2}^j, \Delta z_{i,3}^j)$, $j \in \{0,1\}$, is defined as the output difference of M_j of the i-th round.

Then define $IN^j(a,b) = \{x \in GF(2^8) | S_j(x) \oplus S_j(x \oplus a) = b\}$, $a \neq 0, b \in GF(2^8)$, $j \in \{0,1\}$.

For the 128-bit key scheduling, denote $L^i = (L_0^i, L_1^i, L_2^i, L_3^i)$ as the initial value of L and $T^i = (T_0^i, T_1^i, T_2^i, T_3^i)$ as the final value of T in the i-th iteration in Step 3. For the 192/256-bit key scheduling, denote $L_L^i = (L_{L0}^i, L_{L1}^i, L_{L2}^i, L_{L3}^i)$ as the initial value of L_L, $L_R^i = (L_{R0}^i, L_{R1}^i, L_{R2}^i, L_{R3}^i)$ as the initial value of L_R and $T^i = (T_0^i, T_1^i, T_2^i, T_3^i)$ as the final value of T in the i-th iteration in Step 5.

Finally, let $C_{i,j} (0 \leq i, j \leq 3)$ represent the j-th byte of C_i.

4.2 Attacking Procedure with 128-Bit Key

The attacking procedure with 128-bit key is as follows.

(1) Select randomly a plaintext P, and obtain the right ciphertext C under the secret key $K = (K_0, K_1, K_2, K_3)$.

(2) Attack the 18-th round encryption and recover RK_{34} and RK_{35}.

a) Induce one byte random fault into T_0 of the 17-th round and obtain the corresponding faulty ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*)$. So $\Delta X_{18}^0 = (C_{0,0} \oplus C_{0,0}^*, C_{0,1} \oplus C_{0,1}^*, C_{0,2} \oplus C_{0,2}^*, C_{0,3} \oplus C_{0,3}^*)$, $\Delta Z_{18}^0 = (C_{1,0} \oplus C_{1,0}^*, C_{1,1} \oplus C_{1,1}^*, C_{1,2} \oplus C_{1,2}^*, C_{1,3} \oplus C_{1,3}^*)$, $\Delta Y_{18}^0 = M_0^{-1}(\Delta Z_{18}^0)$.

b) Therefore, $x_{18,i}^0 \in IN^j(\Delta x_{18,i}^0, \Delta y_{18,i}^0), 0 \leq i \leq 3$, $j = i \bmod 2$. Because $x_{18,i}^0 = C_{0,i} \oplus RK_{34,i}$, $RK_{34,i} \in (C_{0,i} \oplus IN^j(\Delta x_{18,i}^0, \Delta y_{18,i}^0))$.

c) Repeat the procedure of a) and b) until RK_{34} can be uniquely determined.

d) Through the similar procedure of a)-c), RK_{35} can be recovered by inducing one byte random fault into T_2 of the 17-th round.

(3) Attack the 17-th round encryption and recover $RK_{32} \oplus WK_3$ and $RK_{33} \oplus WK_2$.

a) Induce one byte random fault into T_0 of the 16-th round and obtain the corresponding faulty ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*)$. It is easy to deduce $\Delta X_{17}^0 = C_3 \oplus C_3^* \oplus \Delta Z_{18}^1 = C_3 \oplus C_3^* \oplus F_1(C_2 \oplus RK_{35}) \oplus F_1(C_2^* \oplus RK_{35})$. $\Delta Z_{17}^0 = (C_{0,0} \oplus C_{0,0}^*, C_{0,1} \oplus C_{0,1}^*, C_{0,2} \oplus C_{0,2}^*, C_{0,3} \oplus C_{0,3}^*)$, $\Delta Y_{17}^0 = M_0^{-1}(\Delta Z_{17}^0)$.

b) Therefore, $x_{17,i}^0 \in IN^j(\Delta x_{17,i}^0, \Delta y_{17,i}^0), 0 \leq i \leq 3$, $j = i \ mod\ 2$. Because $x_{17,i}^0 = C_{3,i} \oplus WK_3 \oplus z_{18,i}^1 \oplus RK_{32,i}$, $RK_{32,i} \oplus WK_{3,i} \in (C_{3,i} \oplus z_{18,i}^1 \oplus IN^j(\Delta x_{17,i}^0, \Delta y_{17,i}^0))$.

c) Repeat the procedure of a) and b) until $RK_{32} \oplus WK_3$ can be uniquely determined.

d) Through the similar procedure of a)- c), $RK_{33} \oplus WK_2$ can be recovered by inducing one byte random fault into T_2 of the 16-th round.

(4) Attack the 16-th round encryption and recover RK_{30} and RK_{31}.

a) Induce one byte random fault into T_0 of the 15-th round and obtain the corresponding faulty ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*)$.

b) $\Delta X_{16}^0 = C_2 \oplus C_2^* \oplus \Delta Z_{17}^1 = C_2 \oplus C_2^* \oplus F_1(RK_{33}, X_{17}^1 \oplus RK_{33}) \oplus F_1(RK_{33}, X_{17}^1 \oplus RK_{33} \oplus \Delta X_{17}^1)$. To compute $F_1(RK_{33}, X_{17}^1 \oplus RK_{33})$, it is no need to know the value of RK_{33} because $RK_{33} \oplus (X_{17}^1 \oplus RK_{33}) = X_{17}^1$. Since X_{17}^1 has been

known in step (3), $F_1(RK_{33}, X_{17}^1 \oplus RK_{33})$ can be calculated out. Similarly, to compute $F_1(RK_{33}, X_{17}^1 \oplus RK_{33} \oplus \Delta X_{17}^1)$, $X_{17}^1 \oplus \Delta X_{17}^1$ should be known. It is easy to deduce $\Delta X_{17}^1 = C_1 \oplus C_1^* \oplus \Delta Z_{18}^0 = C_1 \oplus C_1^* \oplus F_0(RK_{34}, C_0) \oplus F_0(RK_{34}, C_0^*)$, so ΔX_{16}^0 can be calculated out.

c) $\Delta Z_{16}^0 = \Delta X_{17}^0 = C_3 \oplus C_3^* \oplus \Delta Z_{18}^1 = C_3 \oplus C_3^* \oplus F_1(RK_{35}, C_2) \oplus F_1(RK_{35}, C_2^*)$. So $\Delta Y_{16}^0 = M_0^{-1}(\Delta Z_{16}^0)$.

d) Therefore, $x_{16,i}^0 \in IN^j(\Delta x_{16,i}^0, \Delta y_{16,i}^0), 0 \leq i \leq 3, j = imod2$. Because $x_{16,i}^0 = C_{2,i} \oplus F_1(RK_{33}, X_{17}^1 \oplus RK_{33}) \oplus RK_{30,i}$, $RK_{30,i} \in (C_{2,i} \oplus F_1(RK_{33}, X_{17}^1 \oplus RK_{33}) \oplus IN^j(\Delta x_{16,i}^0, \Delta y_{16,i}^0))$.

e) Repeat the procedure of a)- d) until RK_{30} can be uniquely determined.

f) Through the similar procedure of a)- e), RK_{31} can be recovered by inducing one byte random fault into T_2 of the 15-th round.

(5) Analyze the 128-bit key scheduling and recover K.

a) In step 3 of the 128-bit key scheduling, RK_{32}, RK_{33}, RK_{34} and RK_{35} are generated when $i = 8$. As RK_{34} and RK_{35} have been recovered, $T_2^8 = RK_{34}$ and $T_3^8 = RK_{35}$ are also known. As i is not odd, $L_2^8 = T_2^8 \oplus CON_{58}^{(128)}$, $L_3^8 = T_3^8 \oplus CON_{59}^{(128)}$.

b) Through the inverse transformation of $\sum$ and L_2^7 can be calculated out. As $i = 7$ is odd, $T_2^7 = L_2^7 \oplus CON_{54}^{(128)} \oplus K_2$. Because $T_2^7 = RK_{30}$ is known, K_2 can also be calculated out. Therefore, $WK_2 = K_2$ is obtained. As $RK_{33} \oplus WK_2$ has been recovered, RK_{33} is also obtained. So $T_1^8 = RK_{33}$ is also recovered.

c) Apply the inverse transformation of $\sum$, L_3^7 can be calculated out. As $T_3^7 = L_3^7 \oplus CON_{55}^{(128)} \oplus K_3$ and $T_3^7 = RK_{31}$ has been known, K_3 can be obtained. Therefore, $WK_3 = K_2$ is obtained. As $RK_{32} \oplus WK_3$ has been recovered, RK_{32} is also obtained. So $T_0^8 = RK_{32}$ is recovered.

d) As all the bytes of T^8 have been obtained, L^8 can be easily calculated out. Repeat the inverse transformation of $\sum$ until L^0 is deduced. So $K = GFN_{4,12}^{-1}(CON_0^{(128)}, ..., CON_{23}^{(128)}, L^0)$ is recovered.

4.3 Attacking Procedure with 192/256-Bit Keys

The attacking procedure with 192-bit key and 256-bit key is very similar. Due to the page limitation, only the attacking procedure with 192-bit key is presented as follows.

(1) Attack respectively the 22-nd, 21-st, 20-th, 19-th, 18-th, 17-th, ..., and 14-th round encryption, recover RK_{42}, RK_{43}, $RK_{40} \oplus WK_3$, $RK_{41} \oplus WK_2$, RK_{38}, RK_{39}, $RK_{36} \oplus WK_2$, $RK_{37} \oplus WK_3$, RK_{34}, RK_{35}, $RK_{32} \oplus WK_3$, $RK_{33} \oplus WK_2$, RK_{30}, RK_{31}, $RK_{28} \oplus WK_2$, $RK_{29} \oplus WK_3$, RK_{26} and RK_{27}. Here, the similar methods in the 128-bit attacking procedure are adopted to recover the above subkey values.

(2) Analyze the 192-bit key scheduling and recover K.

a) In step 5 of the 192-bit key scheduling, RK_{26} and RK_{27} are generated when $i = 6$. As RK_{26} and RK_{27} have been recovered, $T_2^6 = RK_{26}$ and $T_3^6 = RK_{27}$ are also known. As i is not odd, $L_{R2}^6 = T_2^6 \oplus CON_{66}^{(192)}$, $L_{R3}^6 = T_3^6 \oplus CON_{67}^{(192)}$.

b)Through the transformation of $\sum$, L_{R3}^7 can be calculated out. As $i = 7$ is odd, $T_3^7 = L_{R3}^7 \oplus CON_{71}^{(192)} \oplus K_{L3}$. Because $T_3^7 = RK_{31}$ is recovered, K_{L3} can be calculated out.

c) RK_{34} and RK_{35} are generated when $i = 8$. Since RK_{34} and RK_{35} have been recovered, $T_2^8 = RK_{34}$ and $T_3^8 = RK_{35}$ are also known. As i is not odd, $L_{L2}^8 = T_2^8 \oplus CON_{74}^{(192)}$, $L_{L3}^8 = T_3^8 \oplus CON_{75}^{(192)}$.

d) Through the transformation of $\sum$, L_{L3}^9 can be calculated out. As $i = 9$ is odd, $T_3^9 = L_{L3}^9 \oplus CON_{79}^{(192)} \oplus K_{R3}$. Because $T_3^9 = RK_{39}$ has been recovered, K_{R3} can be calculated out. As $WK_3 = K_{L3} \oplus K_{R3}$, WK_3 is recovered. Since $RK_{40} \oplus WK_3$, $RK_{37} \oplus WK_3$, $RK_{32} \oplus WK_3$ and $RK_{29} \oplus WK_3$ have been recovered, RK_{40}, RK_{37}, RK_{32} and RK_{29} can be obtained.

e) RK_{42} and RK_{43} are generated when $i = 10$. As RK_{42} and RK_{43} have been recovered, $T_2^{10} = RK_{42}$ and $T_3^{10} = RK_{43}$ are also known. $L_{R2}^{10} = T_2^{10} \oplus CON_{82}^{(192)}$, $L_{R3}^{10} = T_3^{10} \oplus CON_{83}^{(192)}$. Through the inverse transformation of $\sum$, L_{R2}^7 can be calculated out. $T_2^7 = L_{R2}^7 \oplus CON_{70}^{(192)} \oplus K_{L2}$. As $T_2^7 = RK_{30}$ has been recovered, K_{L2} can be calculated out.

f) As RK_{32} is known, $T_0^8 = RK_{32}$ is also obtained. $L_{L0}^8 = T_0^8 \oplus CON_{72}^{(192)}$. Through the transformation of $\sum$, L_{L2}^9 can be calculated out. $T_2^9 = L_{L2}^9 \oplus CON_{78}^{(192)} \oplus K_{R2}$. Because $T_2^9 = RK_{38}$ is recovered, K_{R2} can be calculated out.

g) As $WK_2 = K_{L2} \oplus K_{R2}$, WK_2 is recovered. Since $RK_{41} \oplus WK_2$, $RK_{36} \oplus WK_2$, $RK_{33} \oplus WK_2$ and $RK_{28} \oplus WK_2$ have been obtained, RK_{41}, RK_{36}, RK_{33} and RK_{28} can be recovered. Thus

$$L_R^{10} = (RK_{40}|RK_{41}|RK_{42}|RK_{43}) \oplus (CON_{80}^{(192)}|CON_{81}^{(192)}|CON_{82}^{(192)}|CON_{83}^{(192)}),$$

$$L_L^8 = (RK_{32}|RK_{33}|RK_{34}|RK_{35}) \oplus (CON_{72}^{(192)}|CON_{73}^{(192)}|CON_{74}^{(192)}|CON_{75}^{(192)})$$

can be obtained.

h) Repeat the inverse transformation of $\sum$ until L_L^0 and L_R^0 are deduced. So $K_L|K_R = GFN_{8,10}^{-1}(CON_0^{(192)}, ..., CON_{39}^{(192)}, L_L^0|L_R^0)$ is recovered.

4.4 Data Complexity Analysis

In CLEFIA, two different S-boxes S_0 and S_1 are adopted. S_0 is generated by combining 4×4 small S-boxes. S_1 is constructed with the inverse operation plus affine transform in finite field. For the non-empty $IN^0(a, b)(a \neq 0, b \in GF(2^8))$ of S_0, the propagation of (a, b)s satisfying $|IN^0(a, b)| \leq 4$ is 96.2%. For the case of $|IN^0(a, b)| = 2$, 2 faulty ciphertexts should be generated to recover the input of S_0. For the case of $|IN^0(a, b)| = 4$, about 4 faulty ciphertexts should be generated to recover the input of S_0. So about 3 faulty ciphertexts on average are needed to recover the input of S_0. For the non-empty $IN^1(a, b)(a \neq 0, b \in GF(2^8))$ of S_1, the propagation of (a, b)s satisfying $|IN^1(a, b)| = 2$ is 99.2%. So about 2 faulty ciphertexts on average are needed to recover the input of S_1.

In the attacking procedure in section 4.2 and 4.3, if the i-th round is to be attacked, the $(i-1)$-th round will be randomly induced one byte fault, which can

cause four bytes faults to simultaneously happen in the i-th round. So in order to recover a subkey or the sum of a subkey and a post-whitening subkey, only about 3 faulty ciphertexts on average are needed. As one round encryption is composed of two F function and two subkeys are used, about 6 faulty ciphertexts on average should be obtained. For 128-bit key, three round encryptions are to be attacked, so about 18 faulty ciphertexts on average are required. For 192/256-bit keys, nine round encryptions are to be attacked, so about 54 faulty ciphertexts on average are required.

4.5 Computer Simulation

Our attack method has been successfully implemented through the computer simulation. The programming language is Visual C++ 6.0 and the operation system is Windows XP. The attack experiments are repeated ten times on CLEFIA with 128-bit, 192-bit and 256-bit key respectively.

Table 1 gives the induce number of DFA on CLEFIA. Apparently, in most cases, the induce number is 18 for 128-bit key and 54 for 192/256-bit keys, which verify the former data complexity analysis results.

Table 1. Experimental Results of DFA on CLEFIA

the i-th experiment	128-bit	192-bit	256-bit
1	19	54	54
2	19	54	55
3	18	54	54
4	18	54	54
5	18	55	54
6	20	54	54
7	18	54	54
8	18	55	54
9	18	54	55
10	18	54	55

5 Conclusion

In this paper, the differential fault analysis on CLEFIA is explored. The byte-oriented model of random faults is adopted in our attack. Four bytes of faults can be simultaneously obtained in one round by inducing randomly one byte fault in the last round, which can efficiently reduce the total induce times in the attack. After obtaining the subkeys in the last several rounds, and through some analysis of the key schedule, only 18 faulty ciphertexts on average are needed to recover the whole value of secret key for 128-bit key and 54 faulty ciphertexts on average for 192/256-bit keys. The experimental results through the computer simulation also verify the theoretical complexity analysis results.

Acknowledgment

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. 60603013, 60503014 and 90604036) and 863 Project (Grant No. 2007AA01Z470).

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. 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)
6. 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)
7. Chen, C.N., Yen, S.M.: Differential fault analysis on AES key schedule and some countermeasures. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 118–129. Springer, Heidelberg (2003)
8. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2004. LNCS, vol. 3373, pp. 27–41. Springer, Heidelberg (2005)
9. 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)
10. 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)
11. 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)
12. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit Blockcipher CLEFIA. In: Biryukov, A. (ed.) FSE 2007, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
13. Zheng, Y., Matsumoto, T., Imai, H.: On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 461–480. Springer, Heidelberg (1990)