Differential Fault Analysis on the AES Key Schedule

Junko TAKAHASHI and Toshinori FUKUNAGA
NTT Information Sharing Platform Laboratories,
Nippon Telegraph and Telephone Corporation,
{takahashi.junko, fukunaga.toshinori}@lab.ntt.co.jp

Abstract

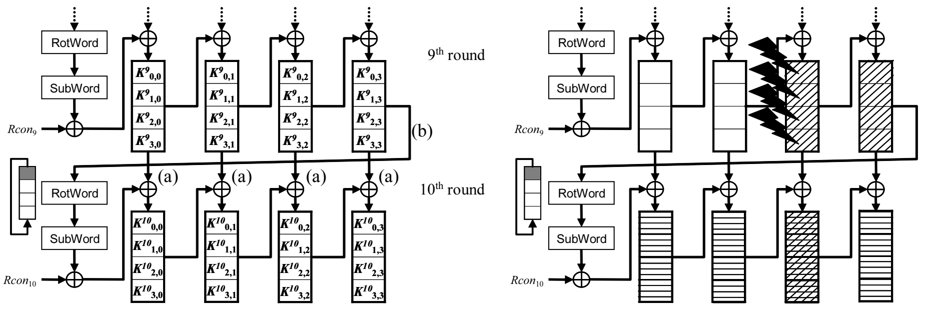
This letter proposes a differential fault analysis on the AES key schedule and shows how an entire 128-bit AES key can be retrieved. In the workshop at FDTC 2007, we presented the DFA mechanism on the AES key schedule and proposed general attack rules. Using our proposed rules, we showed an efficient attack that can retrieve 80 bits of the 128-bit key. Recently, we have found a new attack that can obtain an additional 8 bits compared with our previous attack. As a result, we present most efficient attack for retrieving 88 bits of the 128-bit key using approximately two pairs of correct and faulty ciphertexts.

1 Introduction

Fault analysis is one of the threat attacks for retrieving secret information from secure devices such as smart cards. This attack is executed by inducing faults into the secure device while it is computing the cryptographic algorithm. A differential fault analysis (DFA) on DES was proposed by Biham and Shamir [1]. Their attack obtains keys by analyzing the difference between correct and faulty ciphertexts calculated for the same plaintext. Recently, some papers have proposed a DFA on AES [2, 3, 4] and on the AES key schedule [5, 6, 7].

In our presentation in the workshop at FDTC 2007 [8], we generalized the attack approach to the DFA on the AES key schedule and proposed some attack rules. We showed how to retrieve an 128-bit key without solving complicated simultaneous equations. Using our proposed rules, we showed a more efficient attack than previous ones. Our attack can retrieve 80 bits of the 128-bit key using approximately two pairs of correct and faulty ciphertexts.

In this letter, we propose an even more efficient attack than previous ones including our attack presented in FDTC 2007. This new attack can retrieve an additional 8 bits compared with our previous attack. The result shows that 88 bits of the 128-bit key can be retrieved using two pairs of correct and faulty ciphertexts.



RotWord
SubWord
$Rcon_9$
$K^9_{0,0}$ $K^9_{1,0}$ $K^9_{2,0}$ $K^9_{3,0}$
$K^9_{0,1}$ $K^9_{1,1}$ $K^9_{2,1}$ $K^9_{3,1}$
$K^9_{0,2}$ $K^9_{1,2}$ $K^9_{2,2}$ $K^9_{3,2}$
$K^9_{0,3}$ $K^9_{1,3}$ $K^9_{2,3}$ $K^9_{3,3}$
9th round
(b)
(a)
10th round
$Rcon_{10}$
$K^{10}_{0,0}$ $K^{10}_{1,0}$ $K^{10}_{2,0}$ $K^{10}_{3,0}$
$K^{10}_{0,1}$ $K^{10}_{1,1}$ $K^{10}_{2,1}$ $K^{10}_{3,1}$
$K^{10}_{0,2}$ $K^{10}_{1,2}$ $K^{10}_{2,2}$ $K^{10}_{3,2}$
$K^{10}_{0,3}$ $K^{10}_{1,3}$ $K^{10}_{2,3}$ $K^{10}_{3,3}$

Figure 1: AES key scheduling process: without faults (left) and with faults (right).

In Section 2, we review the DFA mechanism and proposed rules presented in FDTC 2007 (for details, see [8]). Section 3 describes our new attack.

2 DFA on the AES Key Schedule

2.1 DFA mechanism

In this section, we describe only attack essentials of the DFA on the AES key schedule (for details, see [8]). In our attack, we assume that faults injected into 32 bits of one column randomly disturb the intermediate states of the key schedule in the 9th round. This assumption is feasible especially for cryptographic devices with 32-bit (or 16-bit) CPU (or coprocessor). We also assume that such a distribution of the intermediate states occurs just once; that is, faults are induced only one time. Therefore, fault values are $\epsilon^9_{a,b} = \cdots = \epsilon^9_{a,3}$, where a and b are the row and column numbers of the fault-injected byte. For example, when faults are injected into 32 bits of the 3rd column, they propagate to the same row in the 9th round and propagate to the 10th round through path (a) and path (b) in Fig. 1. The specifications of the AES key schedule algorithm are given in [9].

From here on, we use the following notation.

K^l: The lth round key of correct operation, which consists of 16 bytes: a 4 × 4 matrix

$\widetilde{K}^l$: The lth round key of faulty operation, which consists of 16 bytes: a 4 × 4 matrix

$K^l_{i,j}(\widetilde{K}^l_{i,j})$: (i, j) byte of $K^l(\widetilde{K}^l)$ where i and j are the row and column numbers $(i, j = 0 \dots 3)$

$\epsilon^l_{i,j}$: The faulty value, which is the difference between $K^l_{i,j}$ and $\widetilde{K}^l_{i,j}$; that is, $\epsilon^l_{i,j} = K^l_{i,j} \oplus \widetilde{K}^l_{i,j} (i, j = 0 \dots 3)$

$\oplus$: The bitwise exclusive-OR (XOR) operation

The main approach of this attack is that the state calculated by the faulty output must be equal to the state calculated by the correct output before the addition of the faulty 9th round key. An intermediate state m calculated by the correct output before the addition of the correct 9th round key can be written as

$$m = K^9 \oplus InvSubBytes[InvShiftRows[K^{10} \oplus (correct\ output\ bytes)]]. \quad (1)$$

Similarly, an intermediate state m' calculated by the faulty output before the addition of the faulty 9th round key can be written as

$$m' = \widetilde{K}^9 \oplus InvSubBytes[InvShiftRows[\widetilde{K}^{10} \oplus (faulty\ output\ bytes)]]. \quad (2)$$

Since these states must be equal, the following equation is satisfied

$$\begin{aligned} K^9 \oplus & InvSubBytes[InvShiftRows[K^{10} \oplus (correct\ output\ bytes)]] \\ &= \widetilde{K}^9 \oplus InvSubBytes[InvShiftRows[\widetilde{K}^{10} \oplus (faulty\ output\ bytes)]]. \quad (3) \end{aligned}$$

For simplicity, we define $y \doteq (InvShiftRows\ [(correct\ output\ bytes)])$ and $\tilde{y} \doteq (InvShiftRows\ [(faulty\ output\ bytes)])$ where $y_{i,j}(\widetilde{y}_{i,j})$ is the (i,j) byte of $y(\widetilde{y})$. Furthermore, we define $\epsilon^9_{i,j}$ as $\epsilon_{i,j}$ and K^9 as K. We can express all bytes of K^{10} using K [8]. Therefore, each byte of eq. (3) can be written as follows:

$$K_{i,j} \oplus \widetilde{K}_{i,j} \oplus S^{-1}[Q_{i,j} \oplus S[K_{i+1(\mathrm{mod}4),3}] \oplus y_{i,j}] \oplus S^{-1}[\widetilde{Q}_{i,j} \oplus S[\widetilde{K}_{i+1(\mathrm{mod}4),3}] \oplus \widetilde{y}_{i,j}] = 0. \quad (4)$$

where $S[\ ]$ is the S-box function and $S^{-1}[\ ]$ is the inverse S-box function. The notation $Q_{i,j}$ is the (i,j) byte of Q defined as follows:

$$Q = \begin{pmatrix} Rcon_{10} & Rcon_{10} & Rcon_{10} & Rcon_{10} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\oplus \begin{pmatrix} K_{0,0} & K_{0,0} & K_{0,0} & K_{0,0} \\ K_{1,0} & K_{1,0} & K_{1,0} & K_{1,0} \\ K_{2,0} & K_{2,0} & K_{2,0} & K_{2,0} \\ K_{3,0} & K_{3,0} & K_{3,0} & K_{3,0} \end{pmatrix}$$

$$\oplus \begin{pmatrix} 0 & K_{0,1} & K_{0,1} & K_{0,1} \\ K_{1,1} & 0 & K_{1,1} & K_{1,1} \\ K_{2,1} & K_{2,1} & 0 & K_{2,1} \\ K_{3,1} & K_{3,1} & K_{3,1} & 0 \end{pmatrix}$$

$$\oplus \begin{pmatrix} 0 & 0 & K_{0,2} & K_{0,2} \\ K_{1,2} & 0 & 0 & K_{1,2} \\ K_{2,2} & K_{2,2} & 0 & 0 \\ 0 & K_{3,2} & K_{3,2} & 0 \end{pmatrix}$$

$$\oplus \begin{pmatrix} 0 & 0 & 0 & K_{0,3} \\ K_{1,3} & 0 & 0 & 0 \\ 0 & K_{2,3} & 0 & 0 \\ 0 & 0 & K_{3,3} & 0 \end{pmatrix}. \tag{5}$$

Furthermore, $\widetilde{Q}_{i,j}$ is the (i,j) byte of $\widetilde{Q}$. We show an example of $\widetilde{Q}$ for the case where faults are injected into 32 bits of the 3rd column; that is, $(i,2)$ $(i = 0 \ldots 3)$:

$$\widetilde{Q} = Q \oplus \begin{pmatrix} 0 & 0 & \epsilon_{0,2} & 0 \\ 0 & 0 & 0 & \epsilon_{1,2} \\ \epsilon_{2,2} & 0 & 0 & 0 \\ 0 & \epsilon_{3,2} & 0 & 0 \end{pmatrix}. \tag{6}$$

As is shown in [8], each byte of eq. (4) can be classified into 8 patterns, which are the simple equation of the faulty value: $\epsilon_{i,j} = y_{i,j} \oplus \widetilde{y}_{i,j}$, $\epsilon^{l}_{i,j} = 0$ $(l = 9, 10)$ or the formula of types A–F as follows:

type A

$$S[K_{i+1(\mathrm{mod}4),3}] \oplus S[\widetilde{K}_{i+1(\mathrm{mod}4),3}] \oplus y_{i,j} \oplus \widetilde{y}_{i,j} = 0 \tag{7}$$

type B

$$S[K_{i+1(\mathrm{mod}4),3}] \oplus S[\widetilde{K}_{i+1(\mathrm{mod}4),3}] \oplus y_{i,j} \oplus \widetilde{y}_{i,j} \oplus \epsilon_{i,j} = 0 \tag{8}$$

type C

$$\begin{aligned} S^{-1}[Q_{i,j} \oplus & S[K_{i+1(\mathrm{mod}4),3}] \oplus y_{i,j}] \\ & \oplus S^{-1}[Q_{i,j} \oplus S[K_{i+1(\mathrm{mod}4),3}] \oplus \widetilde{y}_{i,j}] \oplus \epsilon_{i,j} = 0 \end{aligned} \tag{9}$$

A	A	F	D
A	A	D	F
B	A	D	D
A	B	D	D

Figure 2: The type of each byte of eq. (4) when faults are injected into 32 bits of the 3rd column.

type D

$$S^{-1}[Q_{i,j} \oplus S[K_{i+1(\mathrm{mod}4),3}] \oplus y_{i,j}] \\ \oplus S^{-1}[Q_{i,j} \oplus S[\widetilde{K}_{i+1(\mathrm{mod}4),3}] \oplus \widetilde{y}_{i,j}] \oplus \epsilon_{i,j} = 0 \tag{10}$$

type E

$$S^{-1}[Q_{i,j} \oplus S[K_{i+1(\mathrm{mod}4),3}] \oplus y_{i,j}] \\ \oplus S^{-1}[Q_{i,j} \oplus S[K_{i+1(\mathrm{mod}4),3}] \oplus \widetilde{y}_{i,j} \oplus \epsilon_{i,j}] \oplus \epsilon_{i,j} = 0 \tag{11}$$

type F

$$S^{-1}[Q_{i,j} \oplus S[K_{i+1(\mathrm{mod}4),3}] \oplus y_{i,j}] \\ \oplus S^{-1}[Q_{i,j} \oplus S[\widetilde{K}_{i+1(\mathrm{mod}4),3}] \oplus \widetilde{y}_{i,j} \oplus \epsilon_{i,j}] \oplus \epsilon_{i,j} = 0 \tag{12}$$

For example, when faults are injected into 32 bits of the 3rd column, each byte of eq. (4) can be described as shown in Fig. 2 (for other cases, see [8]).

2.2 Proposed rules

On the basis of the classification of eq. (4) and the position of each type in the matrix of eq. (4), we previously proposed the following attack rules for retrieving the key.

Rule 1. When a type A byte and a type B byte are in the same row i, we can obtain the faulty value ϵ in row i from one pair of correct and faulty ciphertexts. When the coordinates of the type A byte are (i, a) and those of the type B byte are (i, b), the XOR between eqs. (7) and (8) is given by

$$\epsilon_{i,s} = y_{i,a} \oplus \widetilde{y}_{i,a} \oplus y_{i,b} \oplus \widetilde{y}_{i,b}, \tag{13}$$

where s is the column number of the fault-injected byte in row i. Since $y_{i,a}$, $y_{i,b}$, $\widetilde{y}_{i,a}$, and $\widetilde{y}_{i,b}$ are known values, we can obtain the faulty value $\epsilon_{i,s}$.

Rule 2. If $\epsilon_{i,j}$ is known and there is a type A byte in row $i-1 (\bmod 4)$, we can obtain $K_{i,3}$ in the same row with $\epsilon_{i,j}$ from approximately two pairs of correct and faulty ciphertexts. On the other hand, if $K_{i,3}$ is known and there is a type A byte in row $i-1 (\bmod 4)$, we can obtain $\epsilon_{i,j}$ in the same row with $K_{i,3}$ from one pair of correct and faulty ciphertexts. In fact, if we know $\epsilon_{i,j}$, we can calculate $K_{i,3}$ by substituting $\epsilon_{i,j}$ into the equation for the type A byte:

$$S[K_{i,3}] \oplus S[K_{i,3} \oplus \epsilon_{i,j}] \oplus y_{i-1(\bmod 4),j} \oplus \widetilde{y}_{i-1(\bmod 4),j} = 0. \tag{14}$$

Since we know $\epsilon_{i,j}$, we can solve eq. (14) for $K_{i,3}$:

$$K_{i,3} = \{x \mid S[x] \oplus S[x \oplus \epsilon_{i,j}] \oplus y_{i-1(\bmod 4),j} \oplus \widetilde{y}_{i-1(\bmod 4),j} = 0\}. \tag{15}$$

Using eq. (15), we can determine the correct key with approximately two pairs of correct and faulty ciphertexts (for details, see [5]).

Rule 3. When there are a type A byte and a type D byte (or type F byte) in the same row and $\epsilon_{i,j}$ is known, we can obtain $Q_{i,j} \oplus S[K_{i+1(\bmod 4),3}]$ from approximately two pairs of correct and faulty ciphertexts. On the other hand, when there are a type A byte and a type D byte (or type F byte) in the same row and $Q_{i,j} \oplus S[K_{i+1(\bmod 4),3}]$ is known, we can obtain $\epsilon_{i,j}$ from one pair of correct and faulty ciphertexts (in the case of a type F byte, from approximately two pairs of correct and faulty ciphertexts).

In fact, when the coordinates of the type A byte are (i, a), the type D byte are (i, d) and the type F byte is (i, f), we get the following equations from eqs. (7) and (10) and eqs. (7) and (12)

$$\begin{aligned} S^{-1}[Q_{i,d} \oplus S[K_{i+1(\bmod 4),3}] \oplus y_{i,a} \oplus \widetilde{y}_{i,a} \oplus \widetilde{y}_{i,d}] \oplus \epsilon_{i,d} \\ \oplus S^{-1}[Q_{i,d} \oplus S[K_{i+1(\bmod 4),3}] \oplus y_{i,d}] = 0, \end{aligned}$$

$$\begin{aligned} S^{-1}[Q_{i,f} \oplus S[K_{i+1(\bmod 4),3}] \oplus y_{i,a} \oplus \widetilde{y}_{i,a} \oplus \widetilde{y}_{i,f} \oplus \epsilon_{i,f}] \\ \oplus \epsilon_{i,f} \oplus S^{-1}[Q_{i,f} \oplus S[K_{i+1(\bmod 4),3}] \oplus y_{i,f}] = 0. \end{aligned}$$

Since we know all values except the unknown value ($\epsilon_{i,j}$ or $Q_{i,j} \oplus S[K_{i+1(\bmod 4),3}]$), we can solve the above equations using approximately two pairs of correct and faulty ciphertexts according to rule 2. If we know $K_{i+1(\bmod 4),3}$, we can obtain $Q_{i,j}$ which is the key value itself or the XOR between some key values.

Rule 4. If three of four values – $Q_{i,j}$ in a type D (or type F) byte, $\epsilon_{i,j}$, $\epsilon_{i+1(\bmod 4),j}$ and $K_{i+1(\bmod 4),3}$ – are known, we can obtain the unknown value from approximately two pairs of correct and faulty ciphertexts. This is possible by solving eq. (10) or (12), similar to rule 2.

Rule 5. When there is a pair of known values $Q_{i,j} \oplus \alpha$ and $Q_{i,k} \oplus \alpha$ in the same row (α: any constant value), we can obtain the key value itself or the XOR between the key values by calculating the difference between them. In fact, when we compare the column number j with k (in the case of $j < i$ (or $k < i$), we set j (or k) as $j + 4$ (or $k + 4$) and when we compare $j + 4$ with k (or compare $k + 4$ with j)), we set the smaller one as a and the bigger one as b. We can then calculate the combination of $K_{i,j}$ as follows:

$$\bigoplus_{x=\{(a-i) \bmod 4\}+1}^{(b-i) \bmod 4} K_{i,x} = Q_{i,a} \oplus Q_{i,b}. \tag{16}$$

Rule 6. If there is a type B byte in row i and two of three values – $\epsilon_{i,j}$, $\epsilon_{i+1 (\mathrm{mod}\ 4),j}$, and $K_{i+1 (\mathrm{mod}\ 4),3}$ – are known, we can obtain the unknown value by solving eq. (8). We deduce $\epsilon_{i,j}$ or $\epsilon_{i+1 (\mathrm{mod}\ 4),j}$ using one pair of correct and faulty ciphertexts and we also deduce $K_{i+1 (\mathrm{mod}\ 4),3}$ using approximately two pairs of correct and faulty ciphertexts, similar to rule 2.

Rule 7. When there are two type D (or type F) bytes on (i,j) and (i,k), and $Q_{i,j}$, $Q_{i,k}$, and $K_{i+1 (\mathrm{mod}\ 4),3}$ are known values, we can calculate $\epsilon_{i,j}$ and $\epsilon_{i+1 (\mathrm{mod}\ 4),j}$ using approximately three pairs of correct and faulty ciphertexts. We can obtain them by solving the simultaneous equations of eq. (10) or (12).

3 Attack Expressed by Our Rules

In this section, we show how to retrieve the key using our rules. Here, we consider that faults are injected into 32 bits of the 3rd column (for other cases, see [8]). The corrupted values are described as $\widetilde{K}_{i,2} = K_{i,2} \oplus \epsilon_{i,2}$ and $\widetilde{K}_{i,3} = K_{i,3} \oplus \epsilon_{i,3}$ $(i = 0 \ldots 3)$, where $\epsilon_{i,2} = \epsilon_{i,3}$. Each byte takes the type shown the matrix in Fig. 2.

This matrix shows that there are a type A byte and a type B byte in (2,1) and (2,0). We then apply rule 1 and obtain $\epsilon_{2,2}$. Similarly, since there are a type A byte and a type B byte in (3,0) and (3,1), we apply rule 1 and obtain $\epsilon_{3,2}$. Since we know $\epsilon_{i,2}$ $(i = 2, 3)$ and there is a type A byte above each byte of $\epsilon_{i,2}$ $(i = 2, 3)$, we apply rule 2 and obtain $K_{2,3}$ and $K_{3,3}$. Since we know $\epsilon_{2,2}$ in the type D byte (2,2) and there is a type A byte in (2,1) in the same row of $\epsilon_{2,2}$, we apply rule 3 and obtain $K_{2,0} \oplus S[K_{3,3}]$. Similarly, since we know the faulty value $\epsilon_{2,2}$ in the type D byte (2,3), and there is a type A byte in (2,1) in the same row of $\epsilon_{2,2}$, we apply rule 3 and obtain $K_{2,0} \oplus K_{2,1} \oplus S[K_{3,3}]$. Since we know $K_{3,3}$, we obtain $K_{2,0}$ and $K_{2,1}$ by rule 5. Similarly, since we know $\epsilon_{3,2}$ in the type D byte and there is a type A byte in the same row of $\epsilon_{3,2}$, we obtain $K_{3,0} \oplus K_{3,1} \oplus K_{3,2} \oplus K_{3,3} \oplus S[K_{0,3}]$ and $K_{3,0} \oplus S[K_{0,3}]$ by rule 3. Since we know $K_{3,3}$, we obtain $K_{3,0} \oplus K_{3,1} \oplus K_{3,2} \oplus S[K_{0,3}]$ by rule 5. Since we obtain $K_{3,0} \oplus S[K_{0,3}]$, we also obtain $K_{3,1} \oplus K_{3,2}$ by rule 5.

Next, if we guess $K_{0,3}$, we obtain $\epsilon_{0,2}$ in the type A byte in (3,0) by rule 2. Then, since we know $\epsilon_{0,2}$ and there is the type A byte in (0,0) or (0,1), we obtain $K_{0,0} \oplus K_{0,1} \oplus K_{0,2} \oplus S[K_{1,3}]$ in the type F byte in (0,2) by rule 3. We substitute $K_{0,3}$, $\epsilon_{0,2}$ and $K_{0,0} \oplus K_{0,1} \oplus K_{0,2} \oplus S[K_{1,3}]$ into the equation of the (0,3) byte in order to verify the candidates. If they satisfy the equation of (0,3), the candidates are correct and we can obtain $K_{0,3}$ and $K_{0,0} \oplus K_{0,1} \oplus K_{0,2} \oplus S[K_{1,3}]$ and $\epsilon_{0,2}$. Since we know $K_{0,3}$ and $K_{3,0} \oplus S[K_{0,3}]$, we obtain $K_{3,0}$ by rule 5.

Furthermore, if we guess $K_{1,3}$, we obtain $\epsilon_{1,2}$ in the type A byte in (0,0) or (0,1) by rule 2. Similarly, if we guess $K_{1,0} \oplus K_{1,1}$, we obtain $\epsilon_{1,2}$ in the type D byte in (1,2) by rule 3. Since $\epsilon_{1,2}$ obtained from the above two ways must be equal, we keep the sets of candidates ($K_{1,3}$, $K_{1,0} \oplus K_{1,1}$) that gave the same $\epsilon_{1,2}$. In fact, two sets of candidates ($K_{1,3}$, $K_{1,0} \oplus K_{1,1}$) remain in this step. In order to verify which set is correct, we substitute the candidates ($K_{1,3}$, $K_{1,0} \oplus K_{1,1}$) and $\epsilon_{1,2}$ obtained from them into the equation of the type F byte in (1,3). We can obtain $K_{1,2} \oplus S[K_{2,3}]$ by rule 3 only when we substitute the correct set of ($K_{1,3}$, $K_{1,0} \oplus K_{1,1}$) and $\epsilon_{1,2}$. Since we know $K_{2,3}$, we obtain $K_{1,2}$ by rule 5.

Table 1 shows the above procedure for obtaining the key. It lists the applied rules, using byte types and known or guessed values, and the retrieved information.

As a result, we obtain 88 bits of the key and all 32 bits of the faulty value. We obtain 64 bits of the key directly ($K_{0,3}$, $K_{1,2}$, $K_{1,3}$, $K_{2,0}$, $K_{2,1}$, $K_{2,3}$, $K_{3,0}$, $K_{3,3}$) and 24-bit values of XOR between some key values ($K_{0,0} \oplus K_{0,1} \oplus K_{0,2}$, $K_{1,0} \oplus K_{1,1}$, $K_{3,1} \oplus K_{3,2}$). We have two ways to obtain an entire 128-bit key. One is to execute a 40-bit brute-force search, which is a feasible computation. The other is to change the fault injection point. After injection into the 3$^{\text{rd}}$ column, faults are injected into the 32 bits of the 2$^{\text{nd}}$ and 1$^{\text{st}}$ columns successively. In this case, we use rules 4, 6, and 7, which were not used in this section [8].

4 Conclusion

We proposed a new attack for obtaining an additional 8 bits compared with our previous attack presented in FDTC 2007. The results show that it can obtain 88 bits of the 128-bit AES key using approximately two pairs of correct and faulty ciphertexts.

A comparison with previous attacks is shown in Table 2. Figure 3 shows the relation between the number of required pairs of correct and faulty ciphertexts and the retrieved key bits. These results show that we can obtain the 128-bit AES key using approximately two pairs of correct and faulty ciphertexts with a 40-bit brute-force search, four pairs of them with a 16-bit brute-force search, seven pairs of them without brute-force search. This attack is more efficient than all previous ones.

Table 1: Procedure for retrieving keys.

$\sharp$	Applied rules	Using type byte and known or guessed values $\ll A \gg$ is a guessed value	Retrieved information
1	rule 1	type A in (2,1) type B in (2,0)	$\epsilon_{2,2}$
2	rule 1	type A in (3,0) type B in (3,1)	$\epsilon_{3,2}$
3	rule 2	type A in {(1,0) or (1,1)} $\epsilon_{2,2}$	$K_{2,3}$
4	rule 2	type A in {(2,0) or (2,1)} $\epsilon_{3,2}$	$K_{3,3}$
5	rule 3	type A in (2,1) type D in (2,2) $\epsilon_{2,2}$	$K_{2,0} \oplus S[K_{3,3}]$
6	rule 3	type A in (2,1) type D in (2,3) $\epsilon_{2,2}$	$K_{2,0} \oplus K_{2,1} \oplus S[K_{3,3}]$
7	rule 5	$K_{3,3}$	$K_{2,0}, K_{2,1}$
8	rule 3	type A in (3,0) type D in (3,2) $\epsilon_{3,2}$	$K_{3,0} \oplus K_{3,1} \oplus K_{3,2} \oplus K_{3,3} \oplus S[K_{0,3}]$
9	rule 3	type A in (3,0) type D in (3,3) $\epsilon_{3,2}$	$K_{3,0} \oplus S[K_{0,3}]$
10	rule 5	$K_{3,3}$, $K_{3,0} \oplus S[K_{0,3}]$	$K_{3,1}$, $K_{3,2}$
11	rule 2	type A in (3,0) $\ll K_{0,3} \gg$	$\ll \epsilon_{0,2} \gg$
	rule 3	type A in {(0,0) or (0,1)} type F in (0,2) $\ll \epsilon_{0,2} \gg$	$\ll K_{0,0} \oplus K_{0,1} \oplus K_{0,2} \oplus S[K_{1,3}] \gg$
	rule 3	type D in (0,3) $\ll K_{0,3} \gg$ $\ll K_{0,0} \oplus K_{0,1} \oplus K_{0,2} \oplus S[K_{1,3}] \gg$ $\ll \epsilon_{0,2} \gg$	(when evaluated equation satisfies) $K_{0,3}$ $K_{0,0} \oplus K_{0,1} \oplus K_{0,2} \oplus S[K_{1,3}]$ $\epsilon_{0,2}$
12	rule 5	$K_{0,3}$, $K_{3,0} \oplus S[K_{0,3}]$	$K_{3,0}$
13	rule 2	type A in {(0,0) or (0,1)} $\ll K_{1,3} \gg$	$\ll \epsilon_{1,2} \gg$
	rule 3	type A in {(1,0) or (1,1)} type D in (1,2) $\ll K_{1,0} \oplus K_{1,1} \gg$	$\ll \epsilon_{1,2} \gg$
	rule 3	type A in {(1,0) or (1,1)} type F in (1,3) $\ll K_{1,0} \oplus K_{1,1} \gg$ $\ll \epsilon_{1,2} \gg$	(when evaluated equation satisfies) $K_{1,0} \oplus K_{1,1}$, $K_{1,3}$ $K_{1,2} \oplus S[K_{2,3}]$ $\epsilon_{1,2}$
14	rule 5	$K_{2,3}$, $K_{1,2} \oplus S[K_{2,3}]$	$K_{1,2}$

Table 2: The relation between number of fault injection point and key information (required number of ciphertexts pairs).

Number of fault injection point	Key information (number of pairs)		
	Peacham's attack	Our attack (FDTC)	Our new attack
1	48 bits (2)	80 bits (2)	88 bits (2)
2	88 bits (4)	112 bits (4)	112 bits (4)
3	112 bits (6)	128 bits (7)	128 bits (7)
4	128 bits (12)	–	–

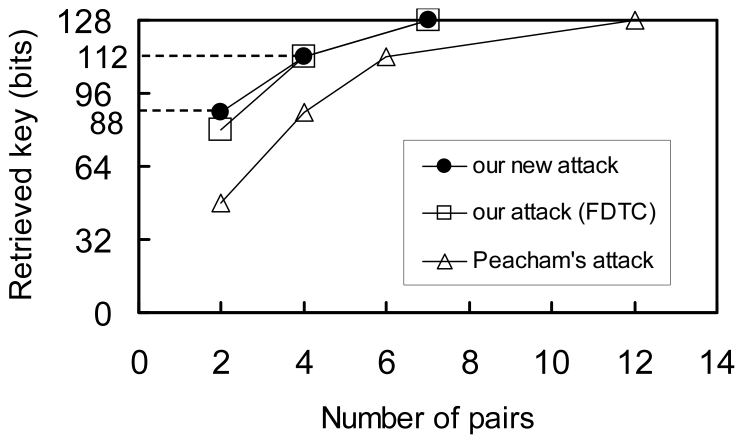


Figure 3: Comparison with previous attacks (number of required pairs of correct and faulty ciphertexts versus retrieved key (bits)).

References

[1] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," *Technion - Computer Science Department - Technical Report CS0901.revised - 1997.*

[2] P. Dusart, G. Letourneux and O. Vivolo, "Differential Fault Analysis on A.E.S.," in J. Zhou, M. Yung, Y. Han, editor, *Applied Cryptography and Network Security – ACNS 2003*, vol. 2846 of *Lecture Notes in Computer Science*, pp. 293-306, 2003.

[3] G. Piret and J. J. Quisquater, "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD," in C. D. Walter *et al.* , editor, *Cryptographic Hardware and Embedded Systems –*

CHES 2003, vol. 2779 of *Lecture Notes in Computer Science*, pp. 77-88, 2003.

[4] A. Moradi, M. T. M. Shalmani and M. Salmasizadeh, “A Generalized Method of Differential Fault Attack Against AES Cryptosystem,” in L. Goubin and M. Matsui, editor, *Cryptographic Hardware and Embedded Systems – CHES 2006*, vol. 4249 of *Lecture Notes in Computer Science*, pp. 91-100, 2006.

[5] C.-N. Chen and S.-M. Yen, “Differential Fault Analysis on AES Key Schedule and Some Countermeasures,” *Australasian Conference on Information Security and Privacy 2003 (ACISP 2003)*, vol. 2727 of *Lecture Notes in Computer Science*, pp. 118-129, Springer, 2003.

[6] C. Giraud, “DFA on AES,” in H. Dobbertin, V. Rijmen and A. Sowa, editors, *Advanced Encryption Standard (AES): 4th International Conference, AES 2004*, vol. 3373 of *Lecture Notes in Computer Science*, pp. 27-41, Springer-Verlag, 2005.

[7] D. Peacham and B. Thomas, “A DFA attack against the AES key schedule,” SiVenture White Paper 001, 26 October 2006. Available at http://www.siventure.com/pdfs/AES_KeySchedule_ DFA_whitepaper.pdf

[8] J. Takahashi, T. Fukunaga and K. Yamakoshi, “DFA mechanism on the AES key schedule,” in proceedings of the *Fourth International Workshop, FDTC 2007, IEEE-Computer Society*, pp. 62-72.

[9] National Institute of Standards and Technology, *Advanced Encryption Standard*, **NIST FIPS PUB 197**, 2001.