Differential Fault Analysis for Round-Reduced AES by Fault Injection

JeaHoon Park, SangJae Moon, DooHo Choi, YouSung Kang, and JaeCheol Ha

This paper presents a practical differential fault analysis method for the faulty Advanced Encryption Standard (AES) with a reduced round by means of a semi-invasive fault injection. To verify our proposal, we implement the AES software on the ATmega128 microcontroller as recommended in the standard document FIPS 197. We reduce the number of rounds using a laser beam injection in the experiment. To deduce the initial round key, we perform an exhaustive search for possible key bytes associated with faulty ciphertexts. Based on the simulation result, our proposal extracts the AES 128-bit secret key in less than 10 hours with 10 pairs of plaintext and faulty ciphertext.

Keywords: AES, fault attack, differential fault attack.

Manuscript received Aug. 11, 2010; revised Oct. 14, 2010; accepted Nov. 18, 2010.

This work was supported by SCARF project which is the IT R&D program of MKE/KEIT, Rep. of Korea (KI002066, Development of the Technology of Side Channel Attack Countermeasure Primitives and Security Validation).

JeaHoon Park (phone: +82 53 940 8817, email: jenoon65@ee.knu.ac.kr) is with the Graduate School of Electrical Engineering and Computer Science, Kyungpook National University, Daegu, Rep. of Korea.

SangJae Moon (email: sjmoon@ee.knu.ac.kr) is with the School of Electronics Engineering, Kyungpook National University, Daegu, Rep. of Korea.

DooHo Choi (email: dhchoi @etri.re.kr) and YouSung Kang (email: youskang@etri.re.kr) are with the Software Research Laboratory, ETRI, Daejeon, Rep. of Korea.

JaeCheol Ha (corresponding author, email: jcha@hoseo.edu) is with the Department of Information Security, Hoseo University, Asan, Choongnam, Rep. of Korea.

doi:10.4218/etrij.11.0110.0478

I. Introduction

Many hardware implementations of cryptographic applications without countermeasures against physical attacks suffer from various physical attacks, such as side-channel attacks and fault attacks. Among these physical attacks, the differential fault analysis (DFA) attack extracts the secret key by analyzing the differences between correct and faulty ciphertexts resulting from fault injection during execution of an algorithm. In 1996, Biham and Shamir first introduced the DFA attack on an implementation of the data encryption standard (DES) with errors induced by fault injection [1]. Subsequently, many researchers have mounted DFAs on symmetric key encryption algorithms, such as the triple-DES [2], Advanced Encryption Standard (AES) [3]-[11], CLEFIA [12], [13], and ARIA [14]. In particular, AES has been considered the main target of DFAs because of its popularity, security, and usefulness.

Previous DFAs for the AES algorithm can be roughly classified into three types according to the assumption of the fault model. In the first type of DFA, the intermediate state values of the encryption process can be corrupted by fault injection [3]-[6]. They use a fault propagation property in which the injected fault before the *MixColumns* operation is diffused by the *MixColumns* transformation; one erroneous byte affects 4 output bytes of *MixColumns*. As an example, Piret and Quisquater's DFA needs only two faulty ciphertexts to extract the 128-bit AES master secret key [5]. The second type of DFA is based on an assumption that a fault can be injected during the key expansion process of the AES [7]-[10]. A fault injected during the key expansion process passes through the three transformations of the key expansion and also affects the encryption process after the AddRoundKey transformation. Thus, modification of the fault injected in the key expansion and diffusion during the encryption process is analyzed. Recently, a DFA by Kim and Quisquater based on this assumption deduces the 128-bit master secret key using only two faulty ciphertexts [10]. The third type of DFA does not assume data corruption but does assume an instruction fault. The DFA by Choukri and Tunstall reduces the number of AES rounds to single round by inputting a voltage glitch on the chip's source power to induce an instruction fault. However, their implementation of AES on the target device is a variant structure. The implementation of the structure is not the one recommended in the standard documentation FIPS 197. If a chip developer implements the recommended AES of FIPS 197, then the DFA by Choukri and Tunstall does not work.

This paper proposes a practical DFA for a faulty AES with a reduced round by means of a semi-invasive fault injection. The effectiveness of the DFA method was verified by a practical experiment. The AES recommended in FIPS 197 was implemented on the target ATmega128 microcontroller. We reduced the number of the AES rounds by using a laser beam injection on the target chip surface after decapsulating. This is a new experimental result of a semi-invasive fault injection for the recommended AES. After obtaining 10 ciphertexts from the reduced-round AES, we successfully extracted the 192-bit and 256-bit secret key of AES as well as 128-bit secret key.

II. Related Works

1. AES Algorithm

The AES specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data [15]. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. The AES algorithm is capable of using cryptographic keys of 128 bits, 192 bits, or 256 bits to encrypt and decrypt a data block of 128 bits. We deal with the 128-bit AES due to its widespread usage. The number of rounds in the 128-bit AES is 10, and the intermediate 16-byte values of the encryption process are called the state, which is usually represented by a 4×4 matrix for 128-bit data.

A. AES Encryption Process

Each round of the 128-bit AES encryption is composed of the following 4 transformation functions:

- SubBytes is a nonlinear byte substitution. We denote the SubBytes function as **SB**.
- ShiftRows is a cyclic shift operation applied to each row using a byte with different offsets. We denote the ShiftRows function as **SR**.
- MixColumns is a linear transformation function applied to each column. We denote the MixColumns function as **MC**.
- AddRoundKey is a bitwise XOR operation applied to each 128-bit round key.

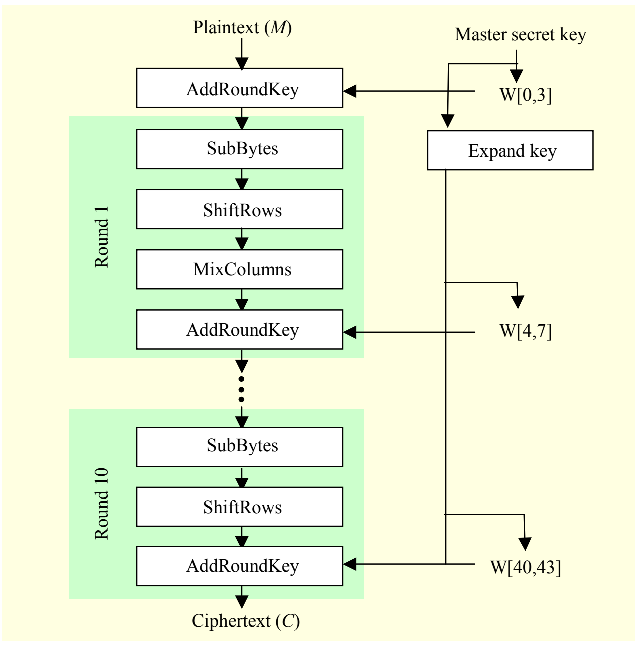


Fig. 1. AES encryption process.

An initial AddRoundKey is required before the first round, and the MixColumns transformation in the last (10th) round is removed. Figure 1 shows the entire AES encryption process.

B. AES Key Expansion Process

The 128-bit AES algorithm takes the master secret key, SK, and performs a key expansion routine to generate 10 round keys. Each expanded round key consists of a linear array of 4-byte words, denoted as $W[i]$. There are three transformation functions in the key expansion process as follows:

- RotWord is a function that takes a word $[a_0, a_1, a_2, a_3]$ as an input, performs a cyclic permutation, and returns the word $[a_1, a_2, a_3, a_0]$.
- SubWord is a function that takes a word composed of 4 bytes and applies the S-box to each byte.
- Rcon[i] is a round constant word given by $[x^{i-1},\{00\},\{00\},\{00\}]$, with x^{i-1} representing powers of x (x is denoted as $\{02\}$) in the field GF(2^8). Note that i starts at 1, not 0.

Figure 2 shows the AES key expansion process. RK_0 is the initial round key identical to the master secret key and RK_i is the i-th round key generated by the key expansion process.

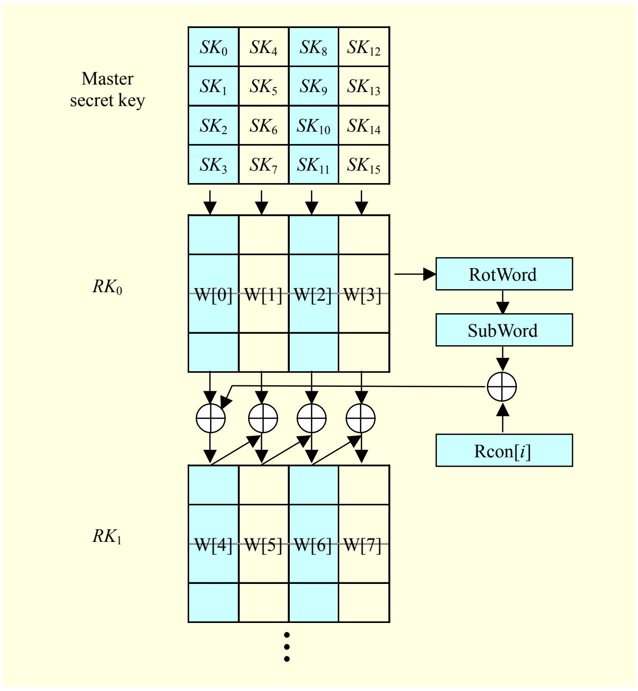


Fig. 2. AES key expansion process.

2. Previous DFA for Round-Reduced AES

In 2005, Choukri and Tunstall presented a DFA on AES by reducing the number of rounds using a fault injection on a Silvercard, which is a smart card based on PIC16F877 [11]. In their experiment, the fault injection method involves a glitch in the power supplied to the smart card. The simplest case of reducing the number of rounds to one was chosen to facilitate subsequent cryptanalysis. They implemented a slightly different AES algorithm from the one described in the AES standard document. In their AES implementation, the last round with a conditional branch to omit MixColumns is included into a "for" loop procedure for executing the AES rounds. Thus, their AES implementation has been reduced to one round by a fault and is composed of the following 5 sequential functions:

- AddRoundKey(); //the initial round key addition,
- SubBytes(); //the first round SB,
- ShiftRows(); //the first round SR,
- MixColumns(); //the first round MC,
- AddRoundKey(); //the first round key addition.

The above round-reduced AES contains only two AddRoundKey transformation functions: the initial one and the first one. If an attacker inputs two plaintexts, M_1 and M_2, respectively, then he can obtain two faulty ciphertexts, C_1' and C_2'. To extract the initial round key RK_0, an attacker checks the following equation per byte:

$$\mathrm{SB}(M_1 \oplus RK_0) \oplus \mathrm{SB}(M_2 \oplus RK_0) = \mathrm{MC}^{-1}(C_1' \oplus C_2'). \quad (1)$$

The ShiftRows function is not taken into account, as it is a byte-wise permutation. For the first round, the AddRoundKey is ignored as the effect of this function is removed by XORing the two faulty ciphertexts.

However, Choukri and Tunstall deliberately implemented a 128-bit AES, of which the last round is included in the "for" loop procedure, unlike the pseudocode in the FIPS 197 document, to facilitate their DFA on the target. If a chip developer implements AES as recommended in the standard document, then their attack will not work.

III. DFA Proposal for Round-Reduced AES

We assume that an attacker can input plaintext and get the corresponding ciphertext during the AES encryption process. Furthermore, he can also reduce the number of rounds by injecting a fault into the target device. A DFA is usually composed of two sequential phases: the physical attack phase to obtain faulty ciphertexts and the computational analysis phase to extract the secret key using the faulty ciphertexts obtained from the physical attack phase. A detailed description of the physical attack phase is also given in subsection IV.1, and the computer simulation results will be shown in subsection IV.2.

1. Proposed DFA Method

We propose a practical DFA method using round-reduced AES by fault injection. Our work deals with AES software implemented in a cryptographic device and uses the pseudocode described in the standard document FIPS 197. Table 1 shows several of the notations used in our paper.

Figure 3 shows the pseudocode of the AES encryption algorithm described in the standard document [15]. The important feature of the pseudocode is that the last round of AES was not included in the "for" loop procedure. If an adversary performs a fault attack on the target device implemented using the pseudocode in Fig. 3, the simplest case of reducing the number of rounds is two rounds: the first and last round. Since there are three AddRoundKey transformation functions, the analysis method by Choukri and Tunstall can no longer be applied. Thus, we develop a new cryptanalysis method of the two-round AES.

Figure 4 shows our faulty encryption process by reducing the number of rounds.

In order to deduce the initial round key, we use the pairs of plaintext and the faulty output of the round-reduced AES in Fig. 4. Based on an exhaustive search for key candidates, we

Table 1. Notations used in AES algorithm.

$M_{i,j}$	The j-th byte of the i-th plaintext
C_i'	The i-th faulty ciphertext
C_i	The intermediate state of the C_i'
SK	The master secret key
$RK_{i,j}$	The j-th byte of the i-th round key
$W[i]$	The i-th word of key expansion result
$S_{i,j}$	The i-th row and j-th column of the state

```
state = M
AddRoundKey(state, &w[0])
for $i$ = 1 step 1 to 9
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, &w[$i$*4])
end for
SubBytes(state)
ShiftRows(state)
AddRoundKey(state, &w[40])
```

Fig. 3. Pseudocode for 128-bit AES encryption.

```
state = M
AddRoundKey(state, &w[0])      //the initial round key addition
SubBytes(state)                //the first round SB
ShiftRows(state)               //the first round SR
MixColumns(state)              //the first round MC
AddRoundKey(state, &w[4])      //the first round key addition
SubBytes(state)                //the last round SB
ShiftRows(state)               //the last round SR
AddRoundKey(state, &w[40])     //the last round key addition
C'= state
```

Fig. 4. Pseudocode for round-reduced AES encryption.

find several bytes of the initial round key used for the key expansion and encryption process of the target byte. A similar exhaustive analysis is repeated with the other faulty ciphertext pairs in order to reduce the key candidates.

We input a plaintext M_1 and obtain a faulty ciphertext C_1' that passed through just 2 rounds: the first and last rounds. Also, we obtain the other faulty ciphertext C_2' corresponding to plaintext M_2 by the same round reduction process. Thus, C_1' and C_2' are described by

$$C_1' = \mathrm{SR}(\mathrm{SB}(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(M_1 \oplus RK_0))) \oplus RK_1)) \oplus RK_{10}, \quad (2)$$

$$C_2' = \mathrm{SR}(\mathrm{SB}(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(M_2 \oplus RK_0))) \oplus RK_1)) \oplus RK_{10}. \quad (3)$$

The result of XORing C_1' with C_2' is as

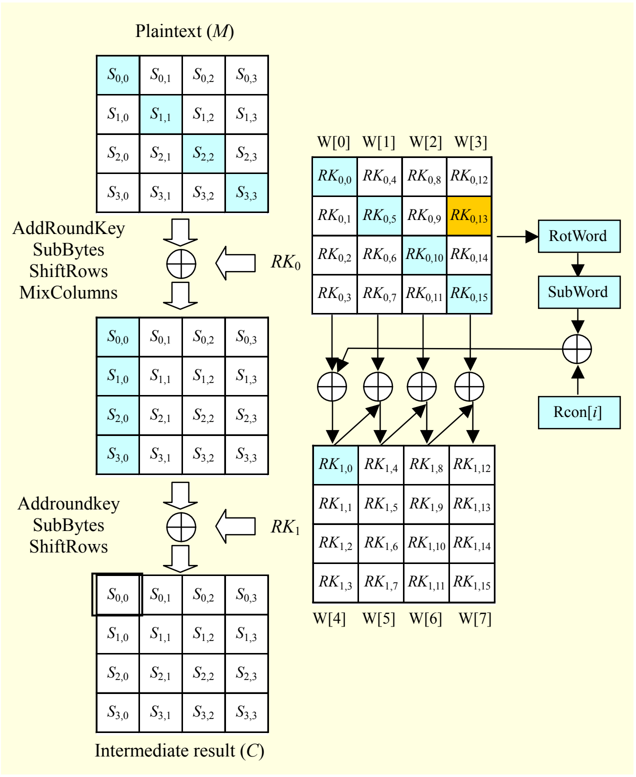


Fig. 5. First round key bytes to compute state $S_{0,0}$ of intermediate result C.

$$\begin{aligned} C_1' \oplus C_2' &= \mathrm{SR}(\mathrm{SB}(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(M_1 \oplus RK_0))) \oplus RK_1)) \\ &\quad \oplus \mathrm{SR}(\mathrm{SB}(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(M_2 \oplus RK_0))) \oplus RK_1)) \\ &= C_1 \oplus C_2, \end{aligned} \quad (4)$$

where C_1 and C_2 are the respective intermediate values before AddRoundKey of the last round. As shown in (4), RK_{10} can be eliminated, and we know M_1, M_2, C_1', and C_2'. We can generate $C_1 \oplus C_2$ using an exhaustive search for possible RK_0 and RK_1 values. Since RK_1 is generated from RK_0, there is no reason to select RK_1 candidates. Thus, we only need to guess the possible key bytes of RK_0. To reduce the range of the exhaustive search, we sequentially recover several bytes of RK_0.

By focusing on state $S_{0,0}$ of intermediate result (C) in Fig. 5, we only need to consider 5 bytes of RK_0. $S_{i,j}$ represents a single byte of the state array where $0 \le i, j < 4$. Figure 5 explains why this is the case. As shown in Fig. 5, to compute $S_{0,0}$ of C, we need to guess 4 bytes of RK_0, that is, $RK_{0,0}$, $RK_{0,5}$, $RK_{0,10}$, and $RK_{0,15}$, and one byte of RK_1, that is, $RK_{1,0}$. $RK_{1,0}$ comes from $RK_{0,0}$, and $RK_{0,13}$ comes from the key expansion process. Thus, we can compute $S_{0,0}$ of C by guessing 5 bytes of RK_0, that is, $RK_{0,0}$, $RK_{0,5}$, $RK_{0,10}$, $RK_{0,13}$, and $RK_{0,15}$. Because there are many possible candidates for the 5 bytes of RK_0, we repeat this analysis with the other plaintext and faulty ciphertext pairs.

Step 1. Analysis process with target $S_{0,0}$ of C.

The key bytes related to $S_{0,0}$ are $RK_{0,0}$, $RK_{0,5}$, $RK_{0,10}$, $RK_{0,15}$, and $RK_{1,0}$. In this case, we can utilize $RK_{0,0}$ and $RK_{0,13}$ instead of $RK_{1,0}$. The following steps are repeated until we find 5 unique secret bytes, that is, $RK_{0,0}$, $RK_{0,5}$, $RK_{0,10}$, $RK_{0,13}$, and $RK_{0,15}$, in the candidate list L using several plaintext and faulty ciphertext pairs.

Step 1.1) Obtain two faulty ciphertexts, $C_1'$and C_2', from the round-reduced AES with arbitrary plaintexts M_1 and M_2, respectively, using the master secret key. Initialize key candidate list L with 2^{40} possible values of 5 bytes of RK_0, that is, $RK_{0,0}$, $RK_{0,5}$, $RK_{0,10}$, $RK_{0,13}$, and $RK_{0,15}$.

Step 1.2) To recover the 5 bytes of RK_0, compute the following equations with the values in candidate list L.

$$\begin{aligned}S^1_{0,0} &= \mathrm{SB}(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(M_{1,j} \oplus RK_{0,j}))) \oplus RK_{1,0}) \\ &= \mathrm{SB}(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(M_{1,j} \oplus RK_{0,j}))) \oplus f_0(RK_{0,0}, RK_{0,13})),\end{aligned} \tag{5}$$

$$\begin{aligned}S^2_{0,0} &= \mathrm{SB}(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(M_{2,j} \oplus RK_{0,j}))) \oplus RK_{1,0}) \\ &= \mathrm{SB}(\mathrm{MC}(\mathrm{SR}(\mathrm{SB}(M_{2,j} \oplus RK_{0,j}))) \oplus f_0(RK_{0,0}, RK_{0,13})),\end{aligned} \tag{6}$$

where SR(SB($M_{i,j}$⊕$RK_{0,j}$)) is the four bytes input of the MixColumns for i=1, 2 and j = 0, 5, 10, 15, and $f_0(a, b)$ is part of the key expansion function for generating $RK_{1,0}$.

Step 1.3) Via the checking process, we update key candidate list L using the values satisfying (7).

$$S^{1'}_{0,0} \oplus S^{2'}_{0,0} = S^1_{0,0} \oplus S^2_{0,0}, \tag{7}$$

where $S^{n'}_{0,0}$ is the (0, 0) state of faulty ciphertext C_n' for n = 1, 2.

Step 1.4) Repeat step 1.2 to step 1.3 using the other plaintext and faulty ciphertext pairs. When there are only 5 byte values in the list L, these values represent the true 5 bytes of RK_0.

Step 2. Analysis process with target $S_{0,1}$ of C.

The key bytes related with $S_{0,1}$ are $RK_{0,4}$, $RK_{0,9}$, $RK_{0,14}$, $RK_{0,3}$, and $RK_{1,4}$. In this case, we can utilize $RK_{1,0}$ and $RK_{0,4}$ instead of $RK_{1,4}$. Since $RK_{1,0}$ was already found in the step 1 analysis, we perform similar processing with step 1.2 to step 1.4 until we find 4 unique secret bytes, $RK_{0,3}$, $RK_{0,4}$, $RK_{0,9}$, and $RK_{0,14}$ in candidate list L with several plaintext and faulty ciphertext pairs.

Step 3. Analysis process with target $S_{0,2}$ of C.

The key bytes related with $S_{0,2}$ are $RK_{0,8}$, $RK_{0,13}$, $RK_{0,2}$, $RK_{0,7}$, and $RK_{1,8}$. In this case, we can utilize $RK_{1,4}$ and $RK_{0,8}$ instead of $RK_{1,8}$. Since $RK_{0,13}$ was already found in step 1 and $RK_{1,4}$ in step 2, we perform similar processing with step 1.2 to step 1.4 until we find the 3 unique secret bytes, $RK_{0,2}$, $RK_{0,7}$, and $RK_{0,8}$ in candidate list L with several plaintext and faulty ciphertext pairs.

Step 4. Analysis process with target $S_{0,3}$ of C.

The key bytes related with $S_{0,3}$ are $RK_{0,1}$, $RK_{0,6}$, $RK_{0,11}$, $RK_{0,12}$, and $RK_{1,12}$. In this case, we can utilize $RK_{1,8}$ and $RK_{0,12}$ instead of $RK_{1,12}$. Since $RK_{1,8}$ was already found in step 3, we perform similar processing with step 1.2 to step 1.4 until we find 4 unique secret bytes, $RK_{0,1}$, $RK_{0,6}$, $RK_{0,11}$, and $RK_{0,12}$ in candidate list L with several plaintext and faulty ciphertext pairs.

Finally, we can deduce all 16 bytes of the master secret key using 10 pairs of plaintext and faulty ciphertext, in detail, there are 5 bytes in step 1, 4 bytes in step 2, 3 bytes in step 3, and 4 bytes in step 4.

2. Number of Plaintext and Faulty Ciphertext Pairs

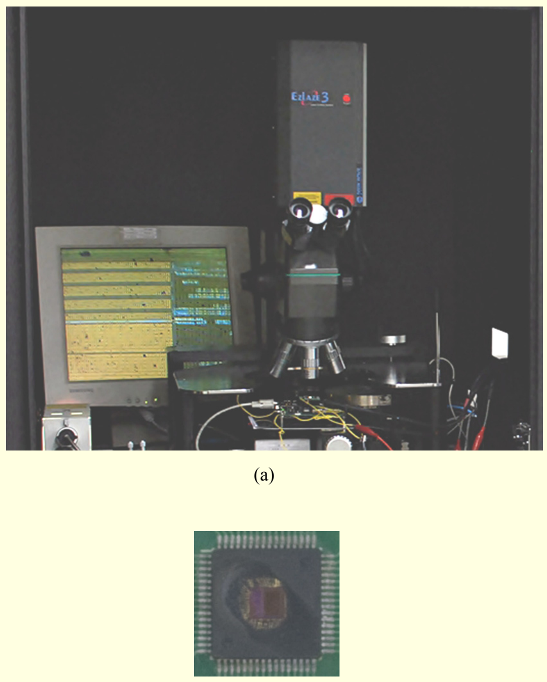
We simply analyze how many faulty ciphertexts are needed for finding the 128-bit initial round key, RK_0. Our proposal performs an exhaustive search for the key bytes related with the one byte faulty ciphertext. After performing analysis to find candidate key bytes with only two pairs of plaintext and faulty ciphertext, an attacker can reduce the number of 5 bytes candidates of the initial round key. At each step, the attacker deals with at most 5 bytes of RK_0 at once when checking the computed byte with the one target byte in the XORed result of two faulty ciphertexts. Thus, the number of key candidates in L is reduced from 2^{40} to 2^{32} (= $2^{40}/2^8$) after performing step 1.1 to step 1.3 with two plaintext and faulty ciphertext pairs. The number of candidates in L is reduced to 2^{24} after repeating the next iteration with the other pairs. This kind of analysis is expressed by

$$\frac{2^l}{(2^8)^r} = 2^{l-8r}, \tag{8}$$

where l is the bit length of the key bytes searched at once, and r is the number of iterations from step 1.2 to step 1.3. In the worst case, the attacker needs to guess 5 bytes at once, namely, l is at most 40. Thus, he must perform this analysis 5 times with 10 pairs of plaintext and faulty ciphertext. The faulty ciphertexts used in step 1 can be reused for the other steps. Thus, the attacker can extract the 128-bit master secret key of the AES with just 10 pairs of plaintext and faulty ciphertext.

3. Extension to 192-bit and 256-bit AES

For the cases of 192-bit and 256-bit AES, we can apply our DFA in a similar manner of the method for the 128-bit AES. The number of key bytes of RK_0 to guess at once is at most 5, and an exhaustive search of these 5 key bytes consumes most of the key search time. For example, for the 192-bit AES, the key bytes related to the state array $S_{0,0}$ are $RK_{0,0}$, $RK_{0,5}$, $RK_{0,10}$, $RK_{0,15}$, and $RK_{1,0}$. The key search steps are repeated until we



(a)

(b)

Fig. 6. Experimental setup: (a) laser for fault injection and (b) decapsulated target chip.

find 5 unique secret bytes, $RK_{0,0}$, $RK_{0,5}$, $RK_{0,10}$, $RK_{0,15}$, and $RK_{1,0}$ in the candidate list L using several plaintext and faulty ciphertext pairs. To extract the master key with 10 pairs of plaintext and faulty ciphertext for the 192-bit AES, we need three exhaustive searches for 5 bytes, and one search each for 4 bytes, 3 bytes, and 2 bytes. Thus, we can assume that the search time needed to extract the 192-bit (24-byte) secret key is about 3 times as long as that of the 128-bit AES.

For the 256-bit AES having a 32-byte key, we need four exhaustive searches for 5 bytes and four searches for 3 bytes. With 10 pairs of plaintext and faulty ciphertext, the search time will be about 4 times as long as that of the 128-bit AES.

Our cryptanalysis method of 128-bit AES can be extended to the 192-bit and 256-bit AES cases while we retain the basic analysis concept. However, since the cryptanalysis method proposed by Choukri and Tunstall only treats the one-round AES, their attack cannot be applied to the 192-bit and 256-bit AES cases.

IV. Experiment of Proposed DFA and Key Search Simulation

1. Fault Injection Experiment

To implement the physical attack phase, we performed a real experiment using a laser. Firstly, we implemented AES software on the ATmega128 microcontroller [16]. Our implementation follows the pseudocode recommended in the AES standard document FIPS 197. Then, we modified the number of rounds by injection of a fault at the "for" loop procedure in Fig. 3. The tool used for fault injection is the EzLaze 3 Laser, which can target a laser beam on the surface of the decapsulated target chip [17]. After several trial and error runs, we were able to obtain 10 faulty ciphertexts from the round-reduced AES in Fig. 4. Figure 6 shows our experimental setup.

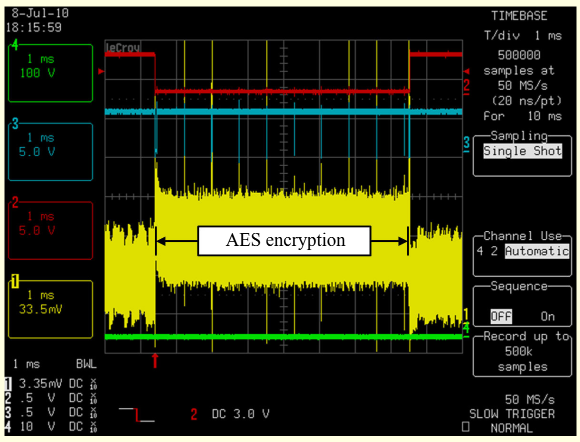


Fig. 7. Captured power signal of 10 round AES encryption.

Compared with non-invasive fault injection methods such as the voltage glitches described in [11], the laser beam injection method is more expensive to implement. In spite of the high cost, the laser beam injection method provides many advantages to an attacker. The combination of the laser and an optical microscope enable to inject a laser beam to desired locations, such as RAM, CPU, EEPROM, and BUS. The fault injection tool can inject a laser beam with a 3 ns or 4 ns laser pulse, and we can control the time of a laser beam injection in steps of 10 ns.

In order to reduce the number of rounds, we targeted a laser beam on the decapsulated target chip as shown in Fig. 6(b). In the experiment, we observed the power signal using a digital oscilloscope, and controlled several I/O signals to distinguish the specific operations. Figure 7 shows the captured power signal of the 10 round AES encryption process.

As shown in Fig. 7, the third signal is the power consumption signal of the entire AES encryption process. It takes 7.513636 ms, as shown in the low state of the first I/O signal. We can distinguish each round by the second I/O signal of the Fig. 7. To skip the other rounds of the AES process after execution of the first round, we injected a laser fault at the time when the "for" loop procedure determines the next iteration after the first round. We assume that this fault can cause an abrupt increase of the counter index value i in Fig. 3. Another assumption is that a fault may influence the comparison

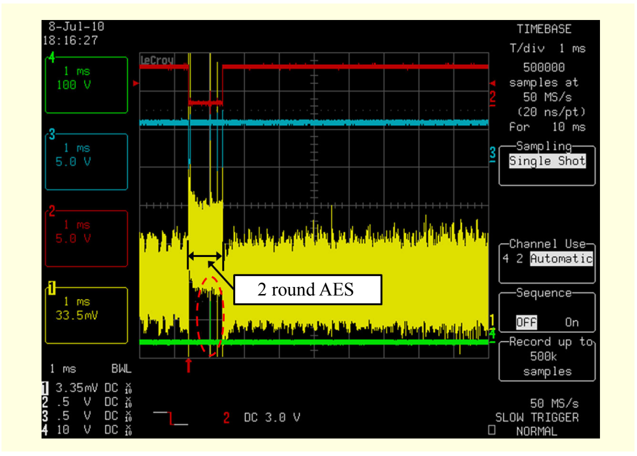
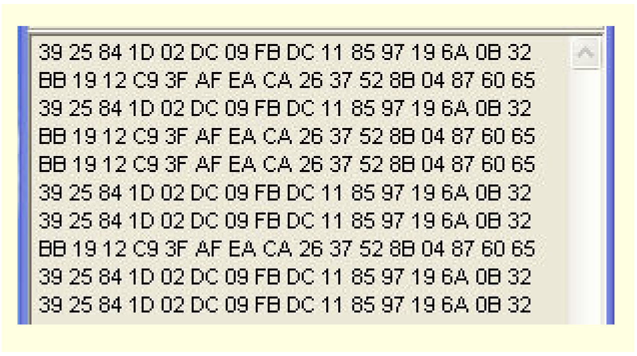


Fig. 8. Power signal of round-reduced AES encryption.



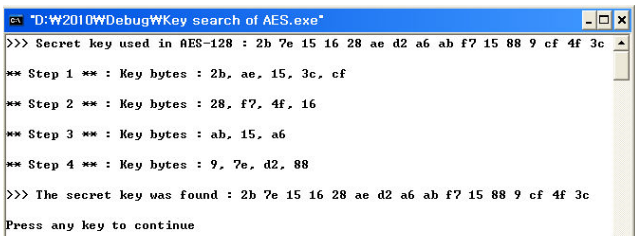
```
39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B 32
BB 19 12 C9 3F AF EA CA 26 37 52 8B 04 87 60 65
39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B 32
BB 19 12 C9 3F AF EA CA 26 37 52 8B 04 87 60 65
BB 19 12 C9 3F AF EA CA 26 37 52 8B 04 87 60 65
39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B 32
39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B 32
BB 19 12 C9 3F AF EA CA 26 37 52 8B 04 87 60 65
39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B 32
39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B 32
```

Fig. 9. Ciphertexts from target chip.

operation of the counter index with a loop constant 9 in Fig. 3.

In practical experiments, using a digital oscilloscope, we observed the power signal and the I/O signals to distinguish the specific operations. The operational time for the first round can be approximated by various methods, for example, observation of the power signal, one round execution time in proportion to the total execution time of AES encryption, and the number of clocks used for the assembly codes. Thus, we can guess the location of the comparison operation in the first round of AES. Subsequently, after several trial and error runs, we could inject a fault into the comparison operation and obtain faulty ciphertexts from a faulty AES with a reduced round. As a result, Fig. 8 shows the power signal of round-reduced AES encryption. It can be seen that round-reduced AES encryption takes about 985.72 μs.

The peak of the fourth signal in the circled area of Fig. 8 indicates the trigger for laser beam injection. The trigger resulted in a 631.60 μs delay time after starting the AES encryption process, and there was about 200 μs preparation time for the laser beam emanation. Consequently, the subsequent rounds after the first round of the AES encryption were successfully skipped as intended. Thus, we were able to obtain the faulty ciphertext of the round-reduced AES as shown in Fig. 9.



```
"D:₩2010₩Debug₩Key search of AES.exe"
>>> Secret key used in AES-128 : 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 9 cf 4f 3c

** Step 1 ** : Key bytes : 2b, ae, 15, 3c, cf

** Step 2 ** : Key bytes : 28, f7, 4f, 16

** Step 3 ** : Key bytes : ab, 15, a6

** Step 4 ** : Key bytes : 9, 7e, d2, 88

>>> The secret key was found : 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 9 cf 4f 3c

Press any key to continue
```

Fig. 10. Key search simulation on PC.

Table 2. Run-time of key search.

Step	Run-time (s)
Step 1 (exhaustive search for 5 bytes)	≈34,560
Step 2 (exhaustive search for 4 bytes)	≈960
Step 3 (exhaustive search for 3 bytes)	≈3.7
Step 4 (exhaustive search for 4 bytes)	≈960

In Fig. 9, "39 25 ... 0B 32" is the correct ciphertext of the 128-bit AES with input "32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 07 34" and the master secret key "2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C" in hexadecimal form. To show the correctness of implementation, we use example input and key given in FIPS 197. However, "BB 19 ... 60 65" is the faulty ciphertext, which is the output of only two AES rounds: the first and last round.

2. Key Search by Computer Analysis

The second phase of DFA is a computational analysis to extract the secret key using faulty ciphertexts. We tried to extract the master secret key on a PC with an i3-530 3.0 GHz CPU with 2 GB memory using Visual C++ software. Figure 10 shows the simulation result of each step in subsection III.1 when 10 pairs of plaintext and faulty ciphertext are used. As shown in Fig. 10, we successfully retrieved the key bytes at each step. Thus, we were able to find the 128-bit secret key after performing 4 steps. The average run-time of the key search at each step is shown in Table 2.

V. Conclusion

We presented a practical DFA for a faulty AES with a reduced round by means of a semi-invasive fault injection. To extract the secret key of AES, we made candidates of possible key bytes and then reduced the number of candidates using the pairs of plaintext and faulty ciphertext obtained from the round-reduced AES. To verify the DFA method, we conducted an experiment on the target chip. We implemented the software AES recommended in FIPS 197 on the ATmega128 microcontroller and injected a laser beam during the execution of AES to obtain faulty ciphertexts with the reduced number of round. After obtaining 10 ciphertexts from the reduced-round AES, we successfully extracted 128-bit AES secret key. Our DFA method can be applied to 192-bit AES and 256-bit AES, and the experimental result is a new one with a semi-invasive fault injection for the recommended AES. Thus, chip developers implementing AES need particular carefulness and vigilance against the fault injection attacks.

References

[1] E. Biham and A. Shamir, “Differential Fault Analysis of Secret Key Cryptosystems,” *Proc. CRYPTO*, LNCS, vol. 1294, 1997, pp. 513-525.

[2] L. Hemme, “A Differential Fault Analysis Against Early Rounds of (Triple-) DES,” *Proc. CHES*, LNCS, vol. 3156, 2004, pp. 254-267.

[3] J. BlÄomer and J. Seifert, “Fault Based Cryptanalysis of the Advanced Encryption Standard (AES),” *Proc. FC*, LNCS, vol. 2742, 2003, pp. 162-181.

[4] P. Dusart, G. Letourneux, and O. Vivolo, “Differential Fault Analysis on AES,” *Proc. ACNS*, LNCS, vol. 2846, 2003, pp. 293-306.

[5] G. Piret and J. Quisquater, “A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD,” *Proc. CHES*, LNCS, vol. 2779, 2003, pp. 77-88.

[6] A. Moradi, M. Shalmani, and M. Salmasizadeh, “A Generalized Method of Differential Fault Attack against AES Cryptosystem,” *Proc. CHES*, LNCS, vol. 4249, 2006, pp. 91-100.

[7] C. Chen and S. Yen, “Differential Fault Analysis on AES Key Schedule and Some Countermeasures,” *Proc. ACISP'03*, LNCS, vol. 2727, 2003, pp. 118-129.

[8] C. Giraud, “DFA on AES,” *Proc. AES*, LNCS, vol. 3373, 2005, pp. 27-41.

[9] J. Takahashi, T. Fikunaga, and K. Yamakoshi, “DFA Mechanism on the AES Key Schedule,” *Proc. FDTC*, 2007, pp. 62-72.

[10] C. Kim and J. Quisquater, “New Differential Fault Analysis on AES Key Schedule: Two Faults Are Enough,” *Proc. CARDIS*, LNCS, vol. 5189, 2008, pp. 48-60.

[11] H. Choukri and M. Tunstall, “Round Reduction Using Faults,” *Proc. FDTC*, 2005, pp.13-24.

[12] H. Chen, W. Wu, and D. Feng, “Differential Fault Analysis on CLEFIA,” *Proc. ICICS*, LNCS, vol. 4861, 2007, pp. 284-295.

[13] T. Shirai et al., “The 128-Bit Block Cipher CLEFIA (Extended Abstract),” *Proc. FSE*, LNCS, vol. 4953, 2007, pp. 181-195.

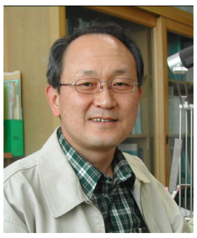
[14] W. Li, D. Gu, and J. Li, “Differential Fault Analysis on the ARIA Algorithm,” *Information Sciences*, Elsevier, vol. 178, no. 19, Oct. 2008, pp. 3727-3737.

[15] NIST, “Announcing the Advanced Encryption Standard,” FIPS 197, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[16] Atmel Corp. Available: http://www.atmel.com/dyn/resources/prod documents/doc2467.pdf

[17] New Wave Research Available: http://www.new-wave.com/1nwrProducts/EZLaze3.htm

JeaHoon Park received the BE and ME in electronics engineering from Kyungpook National University, Rep. of Korea, in 2004 and 2006, respectively. He is currently working toward the PhD at Kyungpook National University. His research interests include side-channel analysis and information security.



SangJae Moon received the BE and ME in electronics from Seoul National University, Rep. of Korea, in 1972 and 1974, respectively. He received the PhD in communication engineering from the University of California, Los Angeles, USA, in 1984. He is currently a professor with the School of Electronics Engineering, Kyungpook National University, Rep. of Korea. Since 2000, he has been the director of the Mobile Network Security Technology Research Center. He is also the honorary president of the Korea Institute of Information Security and Cryptology. His current research interests are information security and side-channel cryptographic analysis. He took part in the Korea Certificate-Based Digital Signature Algorithm Standard project. He has a number of issued patents and more than one hundred technical publications in international journals and conferences in the area of information security.

DooHo Choi received the BS in mathematics from Sungkyunkwan University, Seoul, Korea, in 1994, and the MS and PhD in mathematics from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1996, 2002, respectively. He has been a senior researcher at ETRI, Daejeon, Korea, since January 2002. His current research interests are side-channel analysis and its resistant crypto design, security technologies of RFID and wireless sensor network, lightweight cryptographic protocol/module design, and cryptography based on non-commutativity. He was an editor of the ITU-T Rec. X.1171.

YouSung Kang received the BE and ME degrees in electronics engineering from Chonnam National University, Gwangju, Korea, in 1997 and in 1999, respectively. He is now pursuing the PhD in electrical and electronic engineering from KAIST. In November 1999 he joined ETRI, and he is now a senior member of engineering staff. He is a member of the IEEE, IACR, and the Korea Institute of Information Security & Cryptology (KISC), and he is on the editorial staff of Journal of KISC. His research interests include the areas of RFID/USN security, wireless LAN security, cryptographic protocol, and side-channel analysis.

JaeCheol Ha received the BE, ME, and PhD in electronics engineering from Kyungpook National University, Rep. of Korea, in 1989, 1993, and 1998, respectively. He is currently a professor of the Department of Information and Security at Hoseo University, Asan, Korea. During 1998 to 2006, he also worked as a professor in the Department of Information and Communication at Korea Nazarene University, Cheonan, Korea. In 2006, he was a visiting researcher at the Information Security Institute of Queensland University of Technology, Australia. His research interests include network security, smart card security, and side-channel attacks.