A Generalized Method of Differential Fault Attack Against AES Cryptosystem

Amir Moradi[1], Mohammad T. Manzuri Shalmani[1], and Mahmoud Salmasizadeh[2]

[1] Department of Computer Engineering, Sharif University of Technology, Azadi St., Tehran, Iran
[2] Electronic Research Center, Sharif University of Technology, Azadi St., Tehran, Iran
a_moradi@ce.sharif.edu, {manzuri, salmasi}@sharif.edu

Abstract. In this paper we describe two differential fault attack techniques against Advanced Encryption Standard (AES). We propose two models for fault occurrence; we could find all 128 bits of key using one of them and only 6 faulty ciphertexts. We need approximately 1500 faulty ciphertexts to discover the key with the other fault model. Union of these models covers all faults that can occur in the 9th round of encryption algorithm of AES-128 cryptosystem. One of main advantage of proposed fault models is that any fault in the AES encryption from start (*AddRoundKey* with the main key before the first round) to *MixColumns* function of 9th round can be modeled with one of our fault models. These models cover all states, so generated differences caused by diverse plaintexts or ciphertexts can be supposed as faults and modeled with our models. It establishes a novel technique to cryptanalysis AES without side channel information. The major difference between these methods and previous ones is on the assumption of fault models. Our proposed fault models use very common and general assumption for locations and values of occurred faults.

Keywords: AES, Fault Attacks, Smart Card, Side Channel Attacks, Cryptanalysis.

1 Introduction

At first, Boneh, Demillo and Lipton in 1997 indicated using computational errors occurred during execution of cryptographic algorithm can help to break it and find the secret key [1]. This idea was applicable only on public key cryptosystems and they presented successful results to discover the secret key of a RSA implementation. Subsequently, Biham and Shamir extended this idea for applying it on implementations of symmetric block ciphers such as DES [2] and introduced Differential Fault Attack (DFA) concept. DFAs are powerful and applicable against cryptographic hardwares specially on smart cards.

Many activities have been done on employing DFA to AES implementations by several researches and some methods were introduced [3,5,4,6]. All previous

L. Goubin and M. Matsui (Eds.): CHES 2006, LNCS 4249, pp. 91–100, 2006.

techniques assumed very specific models for fault location and value. Using these methods, such attacks in real world is applicable only with sophisticated equipments such as narrow Laser beam. The most of the results appeared in these papers are simulation based [3,4], however the second attack of [5] was put into practice. In this paper we present two general models for fault occurrence in AES cryptosystem which neither of them needs any sophisticated equipment. The first model covers 1.55% of all possible faults between the beginning of AES-128 and the input of *MixColumns* in round 9, and the reminder (98.45% of them) are covered with the second one. We should emphasize that these models do not cover faults induced during the Key Scheduling as well as safe-errors attacks described in [3]. But in previous methods coverage rate of fault models were tiny. For example, fault models in [4,5] cover approximately $2.4 \times 10^{-5}\%$ of all possible faults induced at input of *MixColumns* in round 9. Therefore, these attacks are applicable with special equipments for injecting certain faults in desired locations. However, our proposed methods could be implemented by power supply disturbance or glitch in clock pulse.

The rest of this paper organized as follows: we explain both of fault models and illustrate their coverage in section 2. The next section describes algorithm of the proposed attack using presented fault models. Section 4 presents simulation results of the proposed attack. In section 5 we show how we can use proposed methods for breaking AES cryptosystem without fault injection. We will show how the AES encryption will be broken only by changing assumptions. Finally section 6 concludes the paper.

2 Proposed Fault Models

In AES with 128-bit key, faults may occur in any function, i.e. *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*, of each 10 rounds. Some previous works [4,5] assumed faults occur in the input of *MixColumns* of the 9th round. Figure 1 shows the last two rounds of AES encryption algorithm, for more information see [7]. We assumed any type of fault appears as a random data to be added to the original data.

Suppose that only one byte of column 1 of input of *MixColumns* is influenced by fault then, 4 bytes of its output will change. Let M stands for *MixColumns* and considering the fact that *MixColumns* operates on each column independently, then equations (1) to (4) could be summarized as equation (5).

$$M\left(A \oplus \begin{bmatrix} e & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\right) = M(A) \oplus \begin{bmatrix} 2 \bullet e & 0 & 0 & 0 \\ e & 0 & 0 & 0 \\ e & 0 & 0 & 0 \\ 3 \bullet e & 0 & 0 & 0 \end{bmatrix} \tag{1}$$

$$M\left(A \oplus \begin{bmatrix} 0 & 0 & 0 & 0 \\ e & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\right) = M(A) \oplus \begin{bmatrix} 3 \bullet e & 0 & 0 & 0 \\ 2 \bullet e & 0 & 0 & 0 \\ e & 0 & 0 & 0 \\ e & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

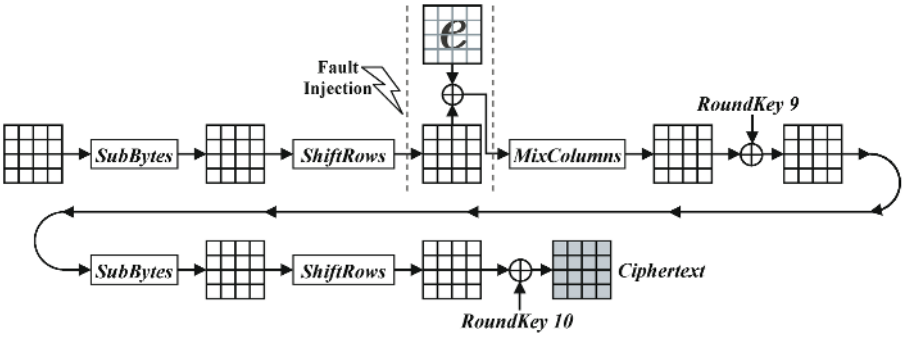

Fig. 1. Last two rounds of AES encryption function

$$M\left(A \oplus \begin{bmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ e\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix}\right) = M(A) \oplus \begin{bmatrix} e & 0\,0\,0 \\ 3 \bullet e & 0\,0\,0 \\ 2 \bullet e & 0\,0\,0 \\ e & 0\,0\,0 \end{bmatrix} \tag{3}$$

$$M\left(A \oplus \begin{bmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ e\,0\,0\,0 \end{bmatrix}\right) = M(A) \oplus \begin{bmatrix} e & 0\,0\,0 \\ e & 0\,0\,0 \\ 3 \bullet e & 0\,0\,0 \\ 2 \bullet e & 0\,0\,0 \end{bmatrix} \tag{4}$$

$$M\left(A \oplus \begin{bmatrix} e_1\,0\,0\,0 \\ e_2\,0\,0\,0 \\ e_3\,0\,0\,0 \\ e_4\,0\,0\,0 \end{bmatrix}\right) = M(A) \oplus \begin{bmatrix} 2 \bullet e_1 \oplus 3 \bullet e_2 \oplus e_3 \oplus e_4 = e'_1\,0\,0\,0 \\ e_1 \oplus 2 \bullet e_2 \oplus 3 \bullet e_3 \oplus e_4 = e'_2\,0\,0\,0 \\ e_1 \oplus e_2 \oplus 2 \bullet e_3 \oplus 3 \bullet e_4 = e'_3\,0\,0\,0 \\ 3 \bullet e_1 \oplus e_2 \oplus e_3 \oplus 2 \bullet e_4 = e'_4\,0\,0\,0 \end{bmatrix} \tag{5}$$

In the first model we suppose that at least one of the bytes e_1 to e_4 is zero.

$$FM_1 = \{\varepsilon : (e_1,\ e_2,\ e_3,\ e_4)\ \mid \exists\ e_i = 0;\ (1 \le i \le 4)\} \tag{6}$$

In other words, at least one byte of *MixColumn* (in one column only) is fault free, but we don't know any other thing about occurred faults such as locations and values. In consequence, this model covers one byte, two bytes and three bytes fault(s) among four bytes of each column. The coverage rate of this model, *CR*, is defined as the proportion of the number of covered faults to the number of all possible faults. Equation (7) gives the *CR* of this model.

$$CR_1 = \frac{\binom{4}{1} \times 255 + \binom{4}{2} \times 255^2 + \binom{4}{3} \times 255^3}{256^4 - 1} = 0.0155 \tag{7}$$

The second model is the complement of the first one i.e., in the second model all four bytes of one column should be faulty.

$$FM_2 = \{\varepsilon : (e_1,\ e_2,\ e_3,\ e_4)\ \mid \forall\ e_i \neq 0;\ (1 \le i \le 4)\} \tag{8}$$

So, all four bytes of one column are influenced by the occurred fault. In this case the fault coverage is given by (9).

$$CR_2 = \frac{255^4}{256^4 - 1} = 0.9845 \tag{9}$$

The second model is more general than the first one, but the first model is more similar with assumed fault models in previous attacks. Additionally, all possible faults can be covered by one of the two presented models and there is no fault that is not included in one of these two models.

It should be emphasized that the intersection of the two presented models is empty and the union of them is all possible faults which can occur in four bytes $(256^4 - 1)$. Consequently, any occurred fault in other units of the encryption algorithm from the beginning of the algorithm up to *MixColumns* of round 9 can be considered as another fault occurred in *MixColumns* input of the 9th round, then it's coverable with one of the illustrated models. None of previous fault models against AES had this capability.

According to the structure of AES, *ShiftRows* exchanges contents of the rows and *MixColumns* composes each column of exchanged rows. Thus, changes in one byte before *ShiftRows* will affect at most on four bytes after *MixColumns*. Figure 2 shows an example that two bytes of *ShiftRows* were induced by fault injection and finally two columns of *MixColumns* output were affected. Consequently, every fault which occurs in a round with high probability leads to big changes in the next round.

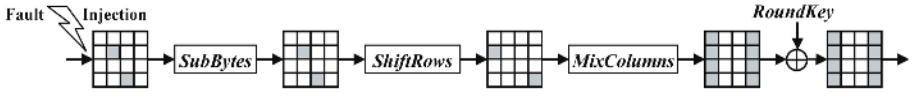

Fig. 2. Effects of faults that occur before *ShiftRows* on *MixColumns*

3 Attack Methods

In this section we show how the new proposed models can be used and then illustrate attack techniques. Consideration equation (5) we generated two set S_1 and S_2.

$$\begin{aligned} S_1 = \{\varepsilon' : (e'_1, e'_2, e'_3, e'_4) \mid \forall\ e'_i \neq 0;\ (1 \leq i \leq 4), \\ \exists\ \varepsilon : (e_1, e_2, e_3, e_4) \in FM_1;\ MixColumn\,(\varepsilon) = (\varepsilon')\} \end{aligned} \tag{10}$$

$$\begin{aligned} S_2 = \{\varepsilon' : (e'_1, e'_2, e'_3, e'_4) \mid \forall\ e'_i \neq 0;\ (1 \leq i \leq 4), \\ \exists\ \varepsilon : (e_1, e_2, e_3, e_4) \in FM_2;\ MixColumn\,(\varepsilon) = (\varepsilon')\} \end{aligned} \tag{11}$$

These two sets can be generated using function *MixColumns* independent of plaintext and key. The (12) and (13) show the number of elements of S_1 and S_2 respectively.

$$|S_1| = \binom{4}{1} \times 255 + \binom{4}{2} \times 255^2 \binom{4}{3} \times 255^3 = 66,716,670 \tag{12}$$

$$|S_2| = 255^4 = 4,228,250,625 \tag{13}$$

According to the figure 3, after *MixColumns* of round 9 each byte of its output affects on one byte of ciphertext independent of other bytes, because the *MixColumns* of round 10 is omitted. In fact this algorithmic weakness of AES causes the success of these attacks. As a result, we could consider each column of *MixColumns* output in round 9 independently. Gray cells in figure 3 show the effects of the first column of the input of *MixColumns* in round 9 on the other internal values. Therefore, errors on each byte of output of *MixColumns* can be traced independently. Equations (15) to (18) show it for the first column.

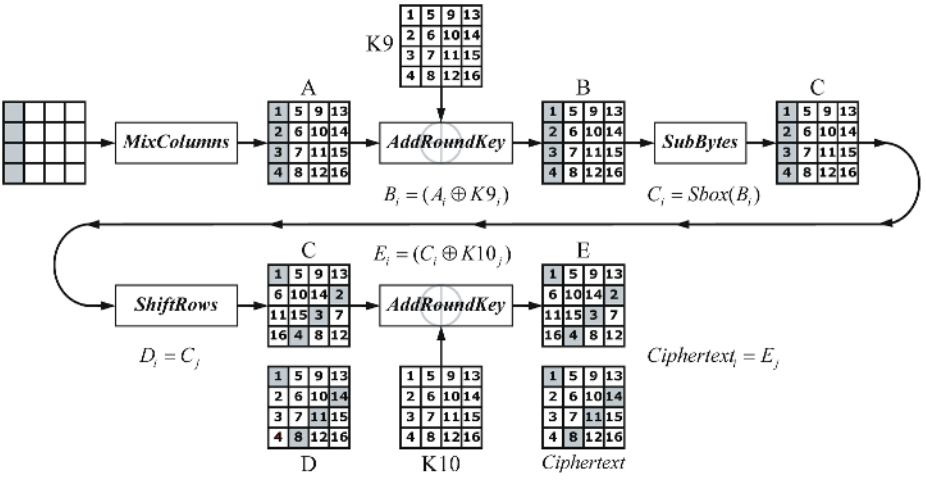

Fig. 3. The AES encryption scheme from *MixColumns* of round 9 to the end

$$Ciphertext = ShiftRows\,(SubBytes\,(A \oplus RoundKey_9)) \oplus RoundKey_{10} \tag{14}$$

A : output of *MixColumns* in round 9, *AddRK : AddRoundKey*

$$AddRK\left(\begin{bmatrix} A_1 \oplus e'_1 \\ A_2 \oplus e'_2 \\ A_3 \oplus e'_3 \\ A_4 \oplus e'_4 \end{bmatrix}, \begin{bmatrix} K9_1 \\ K9_2 \\ K9_3 \\ K9_4 \end{bmatrix}\right) = AddRK\left(\begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix}, \begin{bmatrix} K9_1 \\ K9_2 \\ K9_3 \\ K9_4 \end{bmatrix}\right) \oplus \begin{bmatrix} e'_1 \\ e'_2 \\ e'_3 \\ e'_4 \end{bmatrix} \tag{15}$$

$$SubBytes\left(\begin{bmatrix} B_1 \oplus e'_1 \\ B_2 \oplus e'_2 \\ B_3 \oplus e'_3 \\ B_4 \oplus e'_4 \end{bmatrix}\right) = SubBytes\left(\begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix}\right) \oplus \begin{bmatrix} e''_1 \\ e''_2 \\ e''_3 \\ e''_4 \end{bmatrix} \tag{16}$$

$$ShiftRows: \quad \begin{bmatrix} D_1 \\ D_{14} \\ D_{11} \\ D_8 \end{bmatrix} \oplus \begin{bmatrix} e''_1 \\ e''_2 \\ e''_3 \\ e''_4 \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} \oplus \begin{bmatrix} e''_1 \\ e''_2 \\ e''_3 \\ e''_4 \end{bmatrix} \tag{17}$$

$$AddRK\left(\begin{bmatrix} D_1 \oplus e_1'' \\ D_{14} \oplus e_2'' \\ D_{11} \oplus e_3'' \\ D_8 \oplus e_4'' \end{bmatrix}, \begin{bmatrix} K10_1 \\ K10_{14} \\ K10_{11} \\ K10_8 \end{bmatrix}\right) = AddRK\left(\begin{bmatrix} D_1 \\ D_{14} \\ D_{11} \\ D_8 \end{bmatrix}, \begin{bmatrix} K10_1 \\ K10_{14} \\ K10_{11} \\ K10_8 \end{bmatrix}\right) \oplus \begin{bmatrix} e_1'' \\ e_2'' \\ e_3'' \\ e_4'' \end{bmatrix} \quad (18)$$

AddRoundKey is a linear transformation so (e_1', e_2', e_3', e_4') (errors on output of *MixColumn* and input of *AddRoundKey*) are transferred to its output. But *SubBytes* uses *S-box* transformation and it's a non linear function. As a consequence, $(e_1'', e_2'', e_3'', e_4'')$ presented on output of *SubBytes* does not have any linear relation with (e_1', e_2', e_3', e_4') (errors on its input). But each e_i'' relates to only e_i' and the non linearity of this relation is very high. *ShiftRows* and *AddRoundKey* are linear functions, thus $(e_1'', e_2'', e_3'', e_4'')$ appears exactly on ciphertext but in (1, 14, 11, 8) locations respectively. At the first for presenting the attack, we suppose that all occurred fault are coverable by the first model and consider the first column of input of *MixColumns* in round 9 only. We have one fault free ciphertext (*FFC*) and another faulty ciphertext (*FC*) that occurred fault is covered by the first fault model. Consequently, $\varepsilon'' : (e_1'',\ e_2'',\ e_3'',\ e_4'')$ is given by equation (19).

$$\begin{bmatrix} e_1'' \\ e_2'' \\ e_3'' \\ e_4'' \end{bmatrix} = \begin{bmatrix} FFC_1 \\ FFC_{14} \\ FFC_{11} \\ FFC_8 \end{bmatrix} \oplus \begin{bmatrix} FC_1 \\ FC_{14} \\ FC_{11} \\ FC_8 \end{bmatrix} \quad (19)$$

We know that ε'' is the difference at the output of *SubBytes*. So, we generate set EI.

$$EI = \{\ (\varepsilon' : (e_1', e_2', e_3', e_4')\ ,\ \iota : (I_1, I_2, I_3, I_4))\ \mid$$

$$SubBytes\left(\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix}\right) \oplus SubBytes\left(\begin{bmatrix} I_1 \oplus e_1' \\ I_2 \oplus e_2' \\ I_3 \oplus e_3' \\ I_4 \oplus e_4' \end{bmatrix}\right) = \begin{bmatrix} e_1'' \\ e_2'' \\ e_3'' \\ e_4'' \end{bmatrix}\} \quad (20)$$

But all values of ε' are not useful then we generate set I.

$$I = EI \cap S_1 = \{\iota : (I_1, I_2, I_3, I_4)\ \mid \exists\ \varepsilon';\ \varepsilon' \in S_1\ \wedge\ (\varepsilon', \iota) \in EI\} \quad (21)$$

In other words, set I contains all possible values for the first column of *SubBytes* input at the last round. Thus, we gather some faulty ciphertexts caused by same plaintext and different faults that are covered by the first model. Then we will decrease the size of set I by repeating the proposed method using collected faulty ciphertexts until set I has only one element. Now we know four bytes of *SubBytes* input at the last round. As a consequence, we know its output. On the other hand, we know ciphertext (*FFC*) and according to (23) we can calculate four bytes of the 10th *RoundKey* (*K10*).

$$SubBytes\left(\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix}\right) \oplus \begin{bmatrix} K10_1 \\ K10_{14} \\ K10_{11} \\ K10_8 \end{bmatrix} = \begin{bmatrix} FFC_1 \\ FFC_{14} \\ FFC_{11} \\ FFC_8 \end{bmatrix} \tag{22}$$

$$\begin{bmatrix} K10_1 \\ K10_{14} \\ K10_{11} \\ K10_8 \end{bmatrix} = SubBytes\left(\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix}\right) \oplus \begin{bmatrix} FFC_1 \\ FFC_{14} \\ FFC_{11} \\ FFC_8 \end{bmatrix} \tag{23}$$

Running this method for all other columns of *MixColumns* input of round 9, we will find all 16 bytes of 10th *RoundKey* (*K10*). As a result, we can find the secret key of attacked system by knowing one *RoundKey* completely [4]. The essential functions for discovering the main key from *RoundKey* are *Inverse S-box* and *Exclusive-OR* only.

One of the advantages of this attack is that finding every four bytes of 10th *Roundkey* can be processed separately and parallel. Also, we can employ four dedicated systems that each one tries to find four bytes of *K10*. (1, 14, 11, 8) locations of ciphertexts are examined by the first attacker, the second one employs (5, 2, 15, 12) locations, the third one used (9, 6, 3, 16) locations and the final attacker tries with (13, 10, 7, 4). Then, we will find all 128 bits of *K10*.

The other method to attack is completely similar to the presented one but we assume occurred faults can be covered by the second fault model and we use S_2 for limiting (e'_1, e'_2, e'_3, e'_4) in EI. All other specifications and advantages of the first method are true for the second method.

The main difference between the two attack methods is their fault model. The first model based attack uses any faulty ciphertext with probability of 0.0155 but this value is 0.9845 for the second model based attack.

In these two methods we supposed all faulty ciphertexts are coverable with the first model or by the second model. We can use combination of two models, in each round of attack if we know faulty ciphertext caused by a fault that is covered by the first model (the second model) we limit EI by S_1 (S_2). In this method we should know each occurred fault is coverable with which fault model. But knowing this characteristic of happened fault seems not applicable.

4 Experimental Results

According to the coverage rate of the used fault models, we predicated that we need more faulty ciphertexts in the second attack method than the first one. Because the second fault model has greater coverage rate and many faults are covered with this model. Additional experiments verified this idea.

At the first, we implemented the first method of attack. We started with the first column of *MixColumn* input in round 9 and we selected faulty ciphertexts that all four bytes in 1, 14, 11 and 8 locations are different with fault free ciphertext. In this situation, we ran the attack algorithm to 1000 encryption unit with different random generated keys. In average 6 faulty ciphertexts were

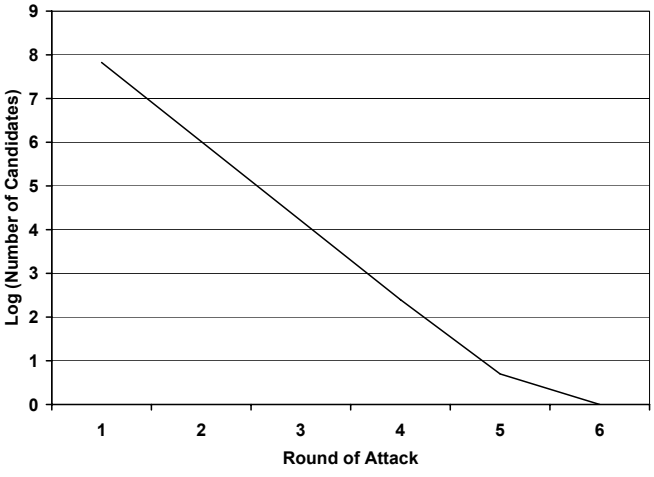

Fig. 4. Average number of candidates for *SubBytes* input in each round of the first attack method

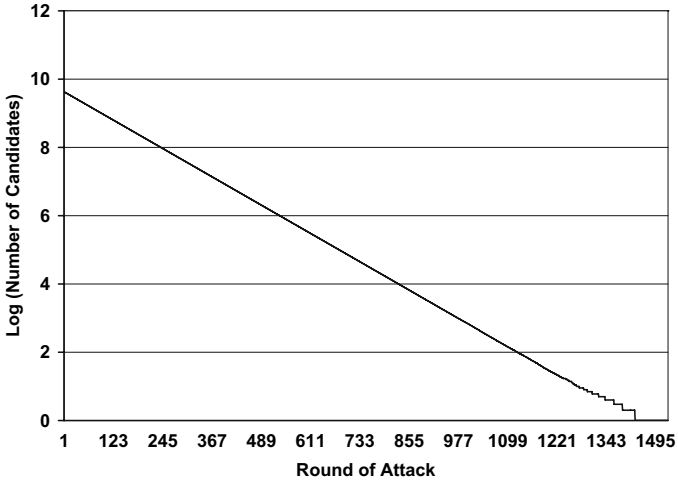

Fig. 5. Average number of candidates for *SubBytes* input in each round of the second attack method

needed to find all four bytes of 10th *RoundKey* and the needed time is not considerable (10 seconds). In the first round of attack we had 6.6×10^7 candidates for *SubBytes* input in average and this number of candidates decreased to 10^6 at the second round of attack. Figure 4 shows average number of candidates in each round of attack.

The explained results were for the the first column of *MixColumns* input and for finding four bytes of *RoundKey*, but those results are correct for other

columns and other bytes of *RoundKey*. As we explained previously, the attack algorithm can be applied to each column synchronously.

But conditions for the second attack method were different because S_2 has more elements and calculating of intersection between S_2 and EI needs more time comparing to the first method. On the other hand, S_2 needs 15.5 GB memory. After improving, optimizing and using memory management techniques on the implementation of the attack, we succeeded to do it with 762.5 MB memory and in almost 2 hours. We should specify that the simulations have been done using Visual C++ on a 2GHz centrino with 1GB memory.

We applied this attack to AES with 100 random keys. Each attack needed 1495 faulty ciphertexts and 2 hours in average to find four bytes of *K10*. It's noticeable, these results are expected according to the previous results of coverage rates. Figure 5 presents the average number of candidates for *SubBytes* inputs on this method.

5 Using Fault Attack Assumption for Breaking AES

We used faulty ciphertexts to find secret key of attacked systems. In proposed methods we supposed faults occur only on internal values, but we assumed *RoundKeys* and *KeyExpansion* unit is completely fault free. As previously described, any fault that happen before the *MixColumns* of round 9 is coverable with one of our proposed fault models. We can suppose fault occurred on the beginning of the encryption algorithm means plaintext. Thus, changing in plaintext that leads to different ciphertexts can be assumed as a fault that occurred in the plaintext and is covered by one of our two models. Then that's enough to know that the caused difference in *MixColumns* input of round 9 is coverable with which of our fault models. We implemented this idea and we supposed that we can access to the input of *MixColumns* in round 9 and we can understand only which model can cover the caused changes in this location. The results of this attack were as successful as previous experimental results. Furthermore, finding a way to know the caused changes in *MixColumns* input of 9th round is coverable with which fault model, is enough to break the AES cryptosystem and finish its era.

Additionally, we don't need to know plaintexts and if we can find a method to distinguish and classify the different ciphertexts based on *MixColumns* input of round 9, we will have a successful *Ciphertext Only Attack* and it's not necessary to run *Known Plaintext Attack*.

6 Conclusion and Future Works

We presented two models for covering all possible faults on input of *MixColumns* in round 9 of the AES-128 encryption algorithm. Then we designed two methods to attack using new proposed fault models. The biggest advantage of these attack methods is high coverage rate of used fault models. One of them covers 1.55% and the other one covers 98.45% of all possible faults on each four bytes of

MixColumns input. None of previous DFAs to the AES had this coverage rate and none of them used general fault models. Additionally, we presented very successful results of proposed attacks implementation. With the first fault model we needed only 6 faulty ciphertexts in average for discovering the main key and 1495 faulty ciphertexts for the second one. Hence, we will succeed in attacking to the implementations of AES-128 with simple fault injection equipments such as power supply disturbance or glitch in clock signal. It's applicable for attacking to new smart cards that implemented AES cryptosystem.

At last we introduced a method for breaking AES without fault injection and with changing assumptions that different ciphertexts caused by different plaintexts not by fault occurrence or injection. In consequence, finding a method to know difference between two ciphertexts is coverable with the first fault model or the other one, is one of our future works. We are working on designing a method to generate some ciphertexts that we know which model covers the difference between each of them. Also, we are trying to construct a test method to know the difference between two ciphertexts at *MixColumns* input in round 9 is coverable with which fault models. Then, by finding any method or designing a rule, we will break AES with 128-bit key and its period will be finished.

Another work for future is trying to run these methods for attacking to the AES cryptosystem with 192 and 256 bits keys. It's noticeable that by illustrated methods we can find completely a *RoundKey* of AES-192 and AES-256. But we can not discover the main key of these systems. We should design other methods for finding the half of another *RoundKey* for AES-192 and whole of another *RoundKey* for AES-256 to reach the secret key.

References

1. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Eliminating Errors in Cryptographic Computations. In *Journal of Cryptology 14(2)*, pages 101-120, 2001.
2. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In B. Kaliski, editor, *Advances in Cryptology - CRYPTO 97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513-525. Springer, 1997.
3. J. Blömer and J.-P. Seifert. Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In *Financial Cryptography 03*, LNCS. Springer, 2003. Also available at http://eprint.iacr.org/,2002/075.
4. P. Dusart, G. Letourneux, and O. Vivolo. Differential Fault Analysis on A.E.S. Available at http://eprint.iacr.org/, 2003/010.
5. C. Giraud. DFA on AES. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *Advanced Encryption Standard (AES): 4th International Conference, AES 2004*, volume 3373 of *Lecture Notes in Computer Science*, pages 27-41. Springer-Verlag, 2005.
6. G. Piret and J.J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In *Cryptographic Hardware and Embedded Systmes - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*. Springer, 2003.
7. National Institute of Standards and Technology, *Advanced Encryption Standard*, **NIST FIPS PUB 197**, 2001.