Efficient Methods for Exploiting Faults Induced at AES Middle Rounds

Chong Hee Kim

Abstract

Faults occurred during the operations in a hardware device cause many problems such as performance deterioration, unreliable output, etc. If a fault occurs in a cryptographic hardware device, the effect can be even serious because an adversary may exploit it to find the secret information stored in the device. More precisely, the adversary can find the key of a block cipher using differential information between correct and faulty ciphertexts obtained by inducing faults during the computation of ciphertexts. This kind of attack is called *Differential Fault Analysis* (DFA). Among many ciphers *Advanced Encryption Standard* (AES) has been the main target of DFA due to its popularity. AES is widely used in different platforms and systems including Intel and AMD microprocessors.

Normally DFA on AES exploits faults induced at the last few rounds. Hence, a general countermeasure is to recompute the last few rounds of AES and compare it with the original output. As redundancy is a costly countermeasure, one should ascertain exactly which rounds need to be protected. In 2006, Phan and Yen introduced a new type of DFA, so called Square-DFA, that works even when faults are induced into some middle rounds. However, it is impractical as it requires several hundreds of faulty ciphertexts as well as a bit fault model. In this article, we propose new attacks that need only dozens of faulty ciphertexts in a byte fault model. Normally it is believed that randomly corrupting a byte is easier than corrupting a specific bit. In addition, we extend the attacks to the AES-192 and AES-256, which is the first result in the literature.

Index Terms

Fault Analysis, Security, Differential Fault Analysis, AES.

I. Introduction

Reliable computation is one of the main concerns in many hardware devices. Especially faults occurred during the operations cause many problems such as performance deterioration, unreliable output, etc. and hence a lot of works to minimize, detect, or prevent faults have been researched. Nowadays we can easily find cryptographic hardware devices such as smart cards everywhere in our daily lives from banking cards to SIM cards for GSM.

C. H. Kim is with Information Security Group, ICTEAM institute, Université catholique de Louvain, Place Sainte Barbe, 2, 1348 Louvain-la-Neuve, Belgium. E-mail: chong-hee.kim@uclouvain.be.

This work has been partially funded by the Walloon Region Marshall plan through the SPW DG06 Project TRASILUX.

These devices are believed to be tamper-resistant. However, if a fault is occurred, an adversary can find the secret information stored in the device. Therefore, we are challenging a new type of fault problem.

More precisely, an adversary can find the key of a block cipher using differential information between correct and faulty ciphertexts obtained by inducing faults during the computation of ciphertexts. This kind of attack is called *Differential Fault Analysis* (DFA). The block cipher is widely used in many cryptographic applications and has been studied extensively in the literature. Traditionally cryptanalysis of block cipher targets the cipher's design and architecture based on an abstract mathematical approach. However, in practice a cipher has to be implemented on a real device that is exposed to physical cryptanalysis such as side-channel attacks [10], [18], [24] and fault attacks [3], [16].

An adversary gets faulty ciphertexts by giving external impact on a device with voltage variation, glitch, laser, etc. [3]. The first DFA presented by Biham and Shamir in 1997 [5] targeted DES [1]. Afterward many people tried to break several cryptosystems such as Triple-DES [14], CLEFIA [8], [26], AES [4], [6], [7], [11], [13], [15], [17], [20], [21], [23], [25], [27], [28], SMS4 and MacGuffin [19]. Among them, *Advanced Encryption Standard* (AES) [2] has been the main target due to its popularity. AES has three variants according to the key size: 128, 192, and 256-bit keys (they are called AES-128, AES-192, and AES-256 respectively).

DFA on AES exploits faults induced at the last few rounds of AES. The best result of DFA on AES-128 exploits faults induced at round $r-3$ [12], [23], [29] and DFA on AES-192 and AES-256 exploit faults at rounds $r-4$ and $r-3$ [15], where r is the number of rounds. Hence, a general countermeasure is to recompute the last few rounds of AES and compare it with the original output. As redundancy is a costly countermeasure, one should ascertain exactly which rounds need to be protected.

In 2006, Phan and Yen introduced a new type of DFA, so called Square-DFA, that integrated the technique of DFA and conventional cryptanalysis of block ciphers [22]. Square-DFA can be mounted with a more flexible attack model, where faults can be induced even in rounds where previous DFAs are inapplicable. However, Square-DFA has two weaknesses that hinder the use of it in practice. First, it requires several hundreds of faulty ciphertexts. Secondly it works in a bit-level fault model.

In this article, we introduce new Square-DFAs that solve the weaknesses that Phan and Yen's Square-DFA have. Specifically, the contributions of this article are as follows:

- We first point out several flaws in Phan and Yen's Square-DFA in [22]. Let **SDFA-Rx** Phan and Yen's Square-DFA that exploits faults induced between rounds $r-x$ and $r-x+1$.
  - We show that **SDFA-R2** and **SDFA-R3** cannot be applied to recover the secret key.
  - We show that **SDFA-R4** needs more faults than that Phan and Yen claimed. They claimed that only 255 faults are necessary. However, **SDFA-R4** cannot identify the secret key with 255 faults, but actually 2^{16} candidates remain. Therefore, another 255 faults or an exhaustive search using a pair of plaintext and ciphertext is necessary.
  - We show that **SDFA-R5** also needs more faults. Phan and Yen claimed that 255 faults are necessary. In fact, at least 510 faults are necessary, even with a practically-sound exhaustive search at the end.

- We introduce new Square-DFAs, **NSDFA-R4** and **NSDFA-R5**, in the same fault model (i.e., a bit-level fault model) that Phan and Yen used.
  - **NSDFA-R4** shows that 255 faults are enough to identify the key. In other words, an adversary is not required to know the plaintext anymore as no exhaustive search is required.
  - **NSDFA-R5** shows that the time complexity can be reduced from 2^{66} to 2^{42}.
- In Phan and Yen's Square-DFAs, an adversary needs to obtain 255 faulty ciphertexts from the faults induced into the same byte such that it would have all 256 (one correct and 255 faulty) values. Therefore, it is assumed that the adversary could induce a bit of fault in a byte. This is very strong assumption as the bit-level modification by faults is very difficult to achieve in practice. Hence, we relax this and introduce new attacks, **NSDFA2-R4**, in a *byte fault model* where a byte is assumed to be randomly modified by faults.
  - **NSDFA2-R4** finds the secret key with only 46 faults in a byte fault model.
- We extend Square-DFAs to AES-192 and AES-256, which is the first result in the literature. This extension is important because AES-192 and AES-256 have been deployed more and more.

The remainder of this article is organized as follows: we briefly describe AES in the next section. The Square-DFA is presented in Section III. The analysis of Phan and Yen's Square-DFA and our new attacks are given in Section IV. After comparing our attacks with existing DFAs in Section V, we conclude in Section VI.

II. AES

AES [2] can encrypt and decrypt 128-bit blocks with 128, 192, or 256-bit keys. The intermediate computation result of AES, called *state*, is usually represented by a 4×4 matrix, where each cell represents a byte. We denote the output of round i by S^i. For example, $S^i_{j,k}$ denotes the $(j+1)^{th}$ row and the $(k+1)^{th}$ column byte of the i^{th} round output, where $j, k \in \{0, \ldots 3\}$, and $i \in \{1, \ldots r\}$. AES-128, AES-192, and AES-256 have 10, 12, and 14 rounds respectively. Each round function is composed of 4 transformations except the last round: *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. The last round is lacking *MixColumns*.

1) SubBytes: It is made up of the application of 16 identical 8×8 S-boxes. This is a non-linear byte substitution. We denote the function of *SubBytes* by $\mathbf{SB}$. That is, $\mathbf{SB}(S^i)$ = SubBytes(S^i). We denote *Inverse SubBytes* by $\mathbf{SB^{-1}}$.

2) ShiftRows: Each row of the *state* is cyclically shifted over different offsets. Row 0 is not shifted, row 1 is shifted by 1 byte, row 2 is shifted by 2 bytes, and row 3 by 3 bytes. We denote *ShiftRows* and its inverse, *InverseShiftRows*, by $\mathbf{SR}$ and $\mathbf{SR^{-1}}$ respectively.

3) MixColumns: This is a linear transformation to each column of the *state*. Each column is considered as polynomial over $\mathbb{F}_{2^8}$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x) = 03 * x^3 + 01 * x^2 + 01 * x + 02$. We denote the function of *MixColumns* by $\mathbf{MC}$ and its inverse by $\mathbf{MC^{-1}}$.

4) AddRoundKeys: It is a bitwise XOR with a round key.

III. SQUARE-DFA

Phan and Yen proposed a new DFA, so called *Square-DFA*, on the AES at CARDIS 2006 [22] based on the *Square Distinguisher* presented by Daemon et al. in [9]. We briefly describe them.

A. Square Distinguisher

Consider a set of 256 plaintexts that differ by one byte in which they take all the possible values (the plaintexts are equal in the other bytes). *SubBytes* and *AddRoundKeys* are permutations so after them the data still have all the possible values in that byte and equal in the others. Then the inputs of the third *MixColumns* have all 256 possible values in each byte. So the XOR of them in each byte is 0. The *MixColumns* is a linear transformation so this property holds after the *MixColumns* too.

B. Square-DFA

The principle of Square-DFA is as follows: the attacker encrypts the same plaintexts and induces a bit of fault on the same byte of any *state* between the *MixColumns* of rounds $r-4$ and $r-3$, and repeats for 255 times, each time inducing one or more bits of fault into that same byte such that it would have all 256 (one correct and 255 faulty) values. By the Square Distinguisher, the XOR of all these 256 ciphertexts (1 correct and 255 faulty ciphertexts) would result in zero in all byte positions at the output of round $r-1$ as shown in Fig. 1.

The adversary now guesses all possible values of any byte of K^r and partially decrypts these 256 ciphertexts by one round up to the output of round $r-1$, and checks if their XOR equals zero. A correct guess always satisfies this property, while a wrong guess would only satisfy with a very low probability. In the same way, she can find the remaining fifteen bytes of K^r using the same ciphertexts. The time complexity is $2^8 \times 16 = 2^{12}$.

Phan and Yen noticed that it is possible to induce the faults a bit deeper into the middle of the AES, in particular one round before, i.e., between the *MixColumns* of rounds $r-5$ and $r-4$. We call this attack **SDFA-R5** to distinguish it from the previous attack (we also call the previous attack **SDFA-R4**).

In **SDFA-R5**, the adversary needs to apply the Square Distinguisher from round $r-4$ to round $r-2$. Now the adversary should decrypt the ciphertexts by the last two rounds. To do so, she has to guess a column of K^{r-1} and the corresponding four bytes of K^r. Then she partially decrypts the ciphertexts by the last two rounds up to the output of round $r-2$ and checks if the XOR is zero in any byte of the column corresponding to that column of guessed K^{r-1}. Repeating this for the other tree columns, she obtains both K^{r-1} and K^r. However the time complexity is much higher than that of **SDFA-R4**. She has to guess 8 bytes each time and repeat it four times. So the time complexity is $(2^8)^8 \times 4 = 2^{66}$.

Phan and Yen also extended their attack to the cases that faults are induced between the *MixColumns* of rounds $r-3$ and $r-2$ and between the *MixColumns* of rounds $r-2$ and $r-1$. We call them **SDFA-R3** and **SDFA-R2** respectively.

In **SDFA-R3**, we are guaranteed that after round $r-1$ we always have all 256 unique values in each byte. This allows one to consider each byte of K^r at a time and performing an attack similar to **SDFA-R4** with the same

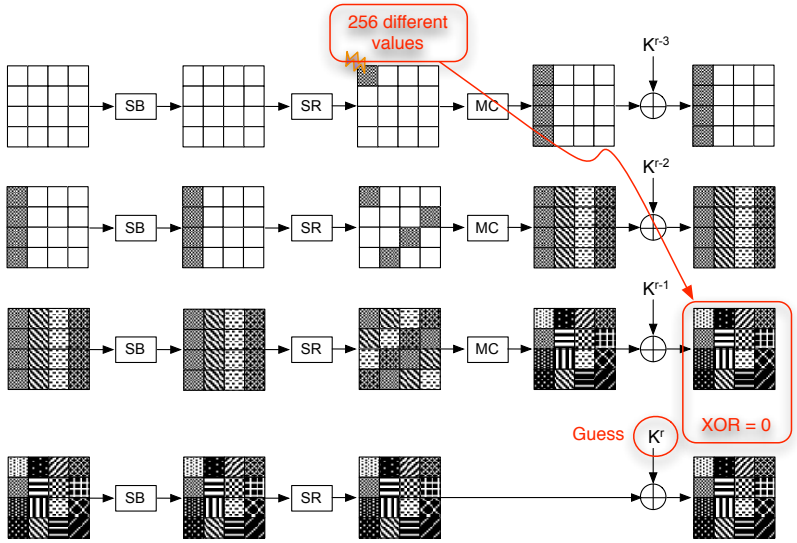

Fig. 1. **SDFA-R4**: faults are injected between the *MixColumns* of rounds $r-4$ and $r-3$. The XOR of all values in each byte is 0 at the output of the *MixColumns* of round $r-1$.

number of faulty ciphertexts, except that instead of computing the XOR value, one would further have to check if all 256 unique values exist.

In **SDFA-R2**, after round $r-1$ we have all 256 unique values in the column in which the fault was induced. We guess at a time each of the four bytes of K^r that corresponds to that column, each time reusing the same faulty ciphertexts. We repeat this four times to obtain all 4 columns of the key, and hence requiring 4 times more ciphertexts.

IV. Improved Square-DFA

A. Analysis of Phan and Yen's SDFAs

1) Probability that a wrong key passes the test:: In **SDFA-R4**, Phan and Yen said that they needed 255 faulty ciphertexts to find the key because for a given byte the probability that the XOR of the 256 values of that byte at the output of round $r-1$ equals zero is very low if a wrong value is guessed for the corresponding byte of K^r [22].

However, we found that this probability is not negligible. The probability of this event (a wrong key satisfies that the XOR equals zero) is $\frac{1}{256}$. As there are 256 candidates for each byte of K^r, on average one wrong guess is left after the test. Hence, for each byte of K^r, on average two candidates (one correct key and one wrong key) remain[1] and for sixteen bytes of K^r, on average 2^{16} candidates remain. Therefore, we can find the correct key by

[1]We performed 1,000 simulations. On average 1.9917 candidates remain after the test.

performing the test again with another set of 255 faulty ciphertexts (hence the total number of faulty ciphertexts becomes 510) or by doing exhaustive search of 2^{16}.

*2) **SDFA-R3** and **SDFA-R2** are impossible::* In **SDFA-R3**, we always have all 256 unique values in each byte at the output of round $r-1$ *regardless of keys*. This property holds even at the output of round r (i.e., in the ciphertexts) since there is no *MixColumns* in the final round. Therefore every candidates for K^r passes the test of **SDFA-R3**. The same consideration applies to **SDFA-R2**. All candidates for K^r pass the test. Therefore, we know that the Square Distinguisher can be used only for three or more rounds to derive the key.

*3) Required number of faulty ciphertexts for **SDFA-R5**::* Phan and Yen said that they could find the key with 255 faulty ciphertexts with **SDFA-R5** [22]. The adversary needs to guess eight bytes (a column for K^{r-1} and the corresponding four bytes for K^r) each time and test if the XOR is equal to zero for each byte of the corresponding column at the output of round $r-2$. The probability that wrong candidates are sorted out by the test for a byte (i.e., the XOR is equal to zero) is $\frac{2}{2^8} = 2^{-7}$ (one correct key and one wrong key survive). The probability for a column is $(2^{-7})^4 = 2^{-28}$. Therefore, among 2^{64} candidates, $2^{64} \times 2^{-28} = 2^{36}$ candidates remain. We have to perform the same procedures for the other three columns. Finally $(2^{36})^4 = 2^{144}$ candidates remain. Therefore, with 255 faulty ciphertexts we cannot find the key.

Then how many faulty ciphertexts are required to find the key? If we have two sets of 255 faulty ciphertexts, the probability that wrong candidates are sorted out by the test for a column is $((2^{-7})^2)^4 = 2^{-56}$. Therefore $2^{64} \times 2^{-56} = 2^8$ candidates remain for a column and hence 2^{32} candidates remain for K^{r-1} and K^r. With three sets, we have $2^{64} \times ((2^{-7})^3)^4 = 2^{64} \times 2^{-84} = 2^{-20}$. Therefore we have one correct key with $3 * 255 = 765$ faulty ciphertexts.

B. Improved SDFAs

We propose two new Square-DFAs, *NSDFA-R4* and *NSDFA-R5*, that reduce the number of required faulty ciphertexts.

*1) **NSDFA-R4**::* The **SDFA-R4** requires two sets of faulty ciphertexts (therefore 510 faulty ciphertexts) to find the key as shown in Section IV-A. We propose a new attack, NSDFA-R4, that can find the key with only one set of faulty ciphertexts.

The **SDFA-R4** partially decrypts 256 ciphertexts by one round up to the output of round $r-1$ and checks if their XOR equals zero. The probability that a wrong guess passes this test is $\frac{1}{256}$. We notice that the inputs of *MixColumns* of round $r-1$ should have all 256 possible values in each byte. Furthermore, we can switch between *MixColumns* and *AddRoundKeys* of round $r-1$ by using $\mathbf{MC^{-1}}(K^{r-1})$ as shown in Fig. 2.

The attack is composed of two steps. In the first step, the adversary uses the property that the XOR equals zero at the output of round $r-1$ and reduces the number of candidates. Then in the second step, she uses the property that each byte has all 256 possible values at the input of *MixColumns* of round $r-1$ and find the correct key. More precisely the adversary finds 2^{16} candidates for K^r by **SDFA-R4** in Step 1. Then for each candidate she computes the input of *MixColumns* of round $r-1$ by performing *InvMixColumns* on the output of round $r-1$ as shown in

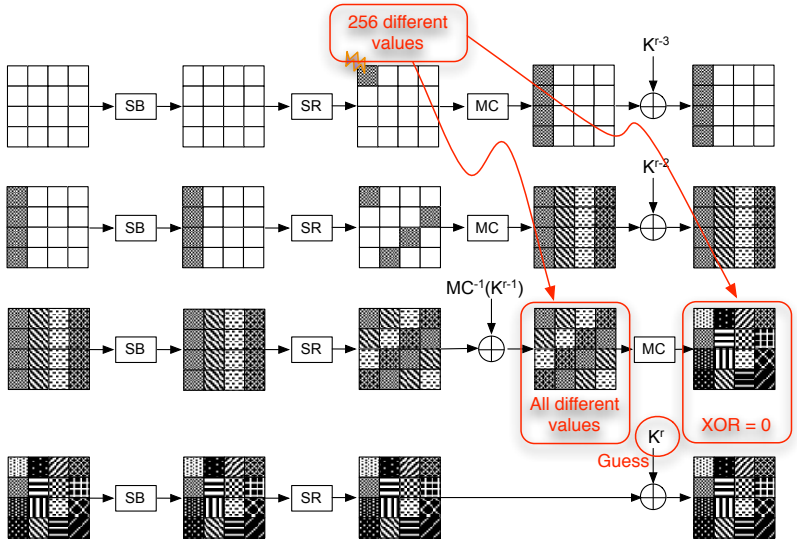

Fig. 2. **NSDFA-R4**: faults are injected between the *MixColumns* of rounds $r-4$ and $r-3$. Then each byte at the input of *MixColumns* of round $r-1$ has all different values. The XOR of all values in each byte is 0 at the output of *MixColumns* of round $r-1$.

Fig. 2. Finally she checks if each byte has all 256 different values. The probability that a wrong guess passes the test is $\frac{256!}{256^{256}} \simeq 0$. Therefore, we have only one correct key at the end.

2) NSDFA-R5:: We propose a new attack, NSDFA-R5, that reduces the time complexity and required number of faulty ciphertexts when the faults are induced between *MixColumns* of rounds $r-5$ and $r-4$.

The adversary guesses one byte of $\widehat{K}^{r-1} = \mathbf{MC^{-1}}(K^{r-1})$ and the corresponding four bytes of K^r (for example, $\widehat{K}^{r-1}_{0,0}, K^r_{0,0}, K^r_{1,3}, K^r_{2,2}$, and $K^r_{3,1}$) as shown in Fig. 3. Then she partially decrypts 256 ciphertexts by the last two rounds to the output of round $r-2$. We denote the output of round $r-2$ by S^{r-2}. She checks if the XOR of the values for the corresponding byte, for example, $S^{r-2}_{0,0}$, at S^{r-2} equals zero. We do the same test with another set of faulty ciphertexts. Then the number of remaining candidates is $(2^8)^5 \times (2^{-7})^2 = 2^{26}$ as we have $(2^8)^5$ candidates for $\widehat{K}^{r-1}_{0,0}, K^r_{0,0}, K^r_{1,3}, K^r_{2,2}$, and $K^r_{3,1}$ and two sets of faulty ciphertexts eliminate $(2^7)^2$ wrong candidates.

She guesses another byte in the same column of $\widehat{K}^{r-1}$, for example, $\widehat{K}^{r-1}_{1,0}$. The number of candidates for $\widehat{K}^{r-1}_{1,0}$ and the four corresponding bytes of K^r ($K^r_{0,0}, K^r_{1,3}, K^r_{2,2}$, and $K^r_{3,1}$) is $2^8 \times 2^{26} = 2^{34}$. With two sets of faulty ciphertexts she checks if the XOR of the values for the corresponding byte, $S^{r-2}_{1,0}$, equals zero. The number of remaining candidates for ($K^r_{0,0}, K^r_{1,3}, K^r_{2,2}$, and $K^r_{3,1}$) is $2^{34} \times 2^{-14} = 2^{20}$. She applies the same procedure for $\widehat{K}^{r-1}_{2,0}$ and $\widehat{K}^{r-1}_{3,0}$. Finally the number of remaining candidates for ($K^r_{0,0}, K^r_{1,3}, K^r_{2,2}$, and $K^r_{3,1}$) is $(2^{20} \times 2^8 \times 2^8) \times (2^{-14})^2 = 2^8$.

After performing the same procedures for the other three columns of $\widehat{K}^{r-1}$, 2^{32} candidates for K^r and K^{r-1} remain. Therefore she can find the key with an exhaustive search.

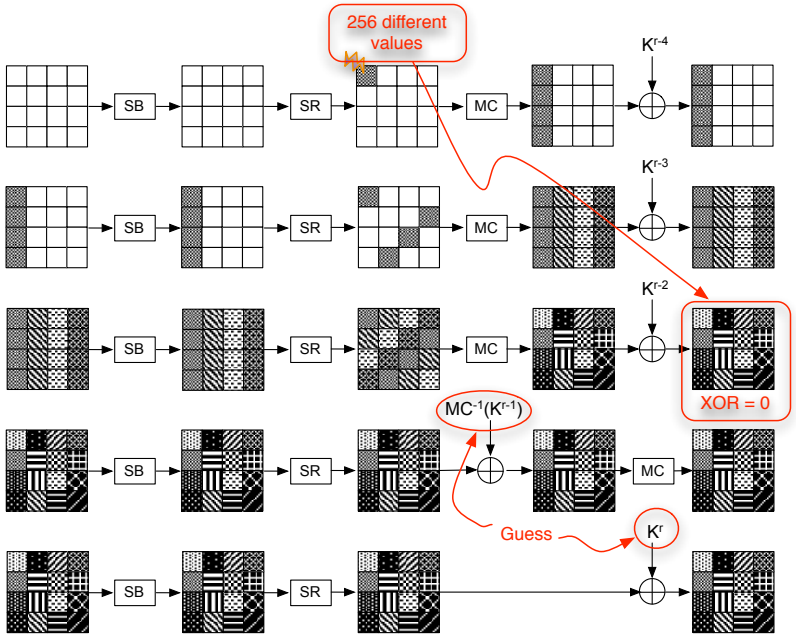

Fig. 3. **NSDFA-R5**: faults are injected between *MixColumns* of rounds $r-5$ and $r-4$. The XOR of all values in each byte is 0 at the output of *MixColumns* of round $r-2$.

C. SDFAs on the AES-192 and AES-256

We extend SDFAs and NSDFAs to the AES-192 and AES-256. In the cases of AES-192 and AES-256, we need two round keys, K^{r-1} and K^r, to compute the master secret key.

1) Application of Phan and Yen's attack:: With two sets of 255 faulty ciphertexts obtained with the faults injected between the *MixColumns* of round $r-4$ and $r-3$, the adversary can find K^r. Then she gets two new sets of 255 faulty ciphertexts with the faults induced between *MixColumns* of rounds $r-5$ and $r-4$. She partially decrypts two rounds with K^r and a guessed byte of $\mathbf{MC^{-1}}(K^{r-1})$ and checks if the XOR of the corresponding byte equals zero. She repeats this search for the other fifteen bytes with the same ciphertexts. Therefore a total of 1020 faulty ciphertexts are required.

2) Improved attack:: The adversary obtains 255 faulty ciphertexts with the faults induced between *MixColumns* of rounds $r-4$ and $r-3$ and finds K^r with **NSDFA-R4** as explained in Section IV-B. She decrypts one round up to the output of round $r-1$ with K^r. Then she can find $\mathbf{MC^{-1}}(K^{r-1})$ with a general DFA (for example, [23]) with any two of 255 faulty ciphertexts as shown in Fig. 4. Therefore total 255 faulty ciphertexts are required.

D. Square-DFA with a Byte Fault Model

In Square-DFA the adversary needs to obtain 255 faulty ciphertexts from the faults induced into the same byte such that it would have all 256 (one correct and 255 faulty) values. Therefore we assumed that the adversary could

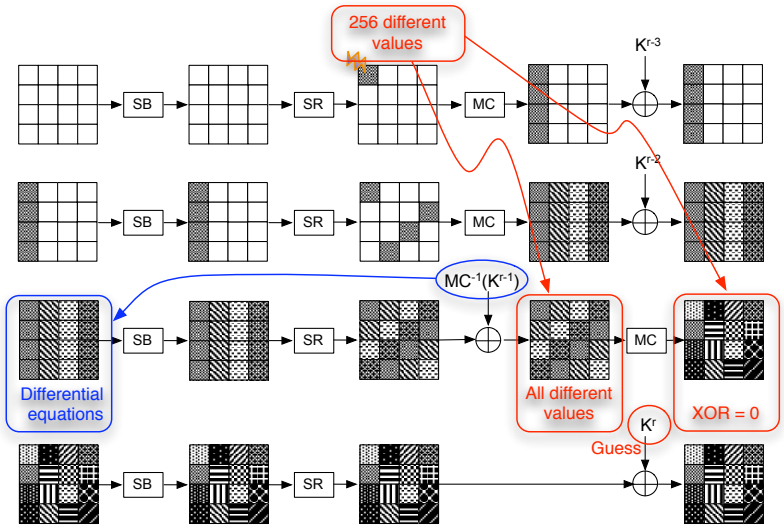

Fig. 4. **Improved attacks on the AES-192 and the AES-256**: faults are injected between *MixColumns* of rounds $r-4$ and $r-3$. The K^r is derived by Square-DFA and the K^{r-1} is derived by DFA.

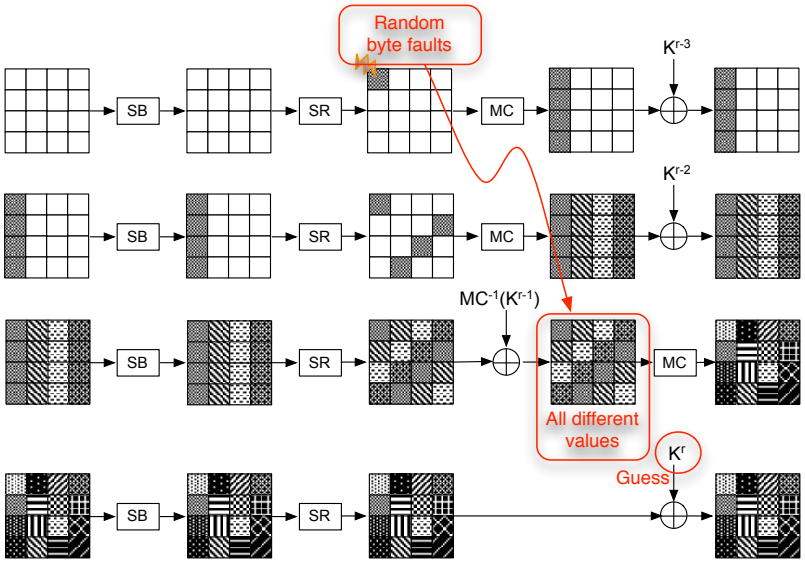

Fig. 5. **New Square DFA with byte faults - NSDFA2-R4**: Random byte faults are injected between *MixColumns* of rounds $r-4$ and $r-3$.

induce a bit of fault in a byte. This is very strong assumption as the bit-level modification by faults is very difficult to achieve in practice.

What if the adversary can only induce byte-level faults instead of bit-level faults? Then how many faulty ciphertexts do we need to have all 255 different values assuming that random byte faults are induced on a byte? We simulated 100,000 times to figure out this and as a result, we found that on average 1560.77 faulty ciphertexts were required. In other words, if we have about 1560 faulty ciphertexts obtained with random byte faults on the same byte, each of 255 different values appears at least once. Therefore we can apply Square-DFA.

We introduce a new method, so called *NSDFA2-R4*, in a byte fault model that requires less faulty ciphertexts by carefully observing the Square-Distinguisher. In NSDFA-R4, we first reduce the candidates by checking if the XOR equals zero at the output of round $r-1$. Then for the remaining candidates we check if all values are different at the input of *MixColumns* of round $r-1$ for 255 faulty ciphertexts (see Fig. 2). The probability that a wrong candidate passes the test is $\frac{256!}{256^{256}}$, which is almost equal to 0. If we have n, ($n < 255$), faulty ciphertexts, the probability that n values randomly chosen between 0 and 255 are different is $p_n = \frac{255\cdot(254)\cdot(255-n+1)}{255^n}$. When n is large enough, this probability goes to 0. Therefore we can find the key with less than 255 faulty ciphertexts.

The new attack works as follows (see Fig. 5): the adversary encrypts the same plaintext and induces a *byte fault* on the same byte of any state between *MixColumns* of rounds $r-4$ and $r-3$, and repeats it n times. She then guesses four bytes of K^r corresponding to the first column of the output of round $r-1$ (i.e., $K^r_{0,0}, K^r_{1,3}, K^r_{2,2}$, and $K^r_{3,1}$) and partially decrypts n ciphertexts by one round up to the input of *MixColumns* of round $r-1$ (we switch between *MixColumns* and *AddRoundKeys* of round $r-1$. So we do not need to consider the effect of the $(r-1)^{th}$ round key). For each byte of the first column of the input of *MixColumns* of round $r-1$, she checks if n values are different. If it is not, she removes that candidate. She repeats this for the three remaining columns and finally finds K^r.

The number of remaining candidates for a column of K^r is $2^{32} \times (p_n)^4$. Therefore the number of remaining candidates for K^r is $(2^{32} \times (p_n)^4)^4$. We provide the theoretical values in Table I.

Table II contains the results of our attack simulated on a PC with a 3.20 GHz Intel and 8GB memory using Visual C++ 7.1 Compiler. From Tables I and II we can conclude that we can find the master key with 46 faulty ciphertexts obtained with *random byte* faults enabling exhaustive key search of 2^{32}.

1) NSDFA2-R5:: Let us consider the application of NSDFA2-R4 to one round earlier. In other words, faults are assumed to be induced between *MixColumns* of rounds $r-5$ and $r-4$. Then, we have to check if each byte of the input of *MixColumns* of round $r-2$ has different values with n faulty ciphertexts. This makes us to guess all sixteen bytes of K^r together and hence, it is impossible to realize.

2) Application to AES-192 and AES-256:: We assume that the faults are induced between *MixColumns* of rounds $r-4$ and $r-3$. The adversary first finds k candidates of K^r with NSDFA2-R4 and finds K^{r-1} with DFA as explained in Section IV-C. The time complexity for the first step is 2^{34} and that of the second step is $k \times 2^{20}$. Therefore we need around $k = 50$ faulty ciphertexts to make the total time complexity less than 2^{34}.

TABLE I

NUMBER OF CIPHERTEXTS AND REMAINING WRONG CANDIDATES IN NSDFA2-R4: THEORETICAL ANALYSIS

# of ciphertexts	p_n	# of remaining wrong candidates for K^r
1	1	2^{128}
2	0.9961	$2^{127.9}$
3	0.9883	$2^{127.7}$
...		
45	0.0161	$2^{32.7}$
46	0.0132	$2^{28.2}$
47	0.0109	$2^{23.6}$
...		
49	0.0072	$2^{14.07}$
50	0.0058	$2^{9.14}$
51	0.0047	17.24
52	0.0037	0.49
53	0.0030	0.01
54	0.0024	0.0003
55	0.0019	0.000007

TABLE II

NUMBER OF CIPHERTEXTS AND REMAINING WRONG CANDIDATES IN NSDFA2-R4: SIMULATION RESULTS

# of ciphertexts	# of remaining wrong candidates for K^r
44	$2^{37.46}$
45	$2^{32.92}$
46	$2^{28.61}$
47	$2^{24.04}$
48	$2^{19.49}$
49	$2^{14.98}$
50	$2^{10.24}$
51	33.18
52	27.98
53	2.86
54	2.86
55	1.45
56	2.07
57	1
58	1
59	1

TABLE III
COMPARISON OF DFAS ON AES-128

Attack	Ref.	Fault model	Fault location (Between MCs of)	# of faulty ciphertexts	Time complexity	Remaining candidates
DFA	[23]	Byte	r-3 and r-2	2	2^{20}	1
	[12]			1	2^{20}	2^{32}
	[29]			1	2^{32}	2^{8}
SDFA-R4	[22]	Bit	r-4 and r-3	510	2^{12}	1
				255	2^{12}	2^{16}
NSDFA-R4	This paper	Bit		255	2^{16}	1
NSDFA2-R4	This paper	Byte		57	2^{34}	1
		Byte		46	2^{34}	2^{32}
SDFA-R5	[22]	Bit	r-5 and r-4	765	2^{66}	1
				510	2^{66}	2^{32}
NSDFA-R5	This paper	Bit		510	2^{42}	2^{32}

V. COMPARISON

We summarize DFAs on AES-128 in Table III and DFAs on AES-192 and AES-256 in Table IV. Generally we notice that Square-DFAs can exploit the faults induced one or two rounds earlier compared to *general* DFAs.

In the case of AES-128, our attack (NSDFA-R4) requires half of the faulty ciphertexts compared to Phan and Yen's (SDFA-R4) or does not require a plaintext for an exhaustive search in a bit fault model. We can further reduce the required number of faulty ciphertexts down to 46 even in a byte fault model. If the faults are induced between *MixColumns* of rounds $r-5$ and $r-4$, our attack (NSDFA-R5) can find the key with time complexity of 2^{42} while Phan and Yen's (SDFA-R5) requires time complexity of 2^{66}.

In the case of AES-192 and AES-256, our attacks require less faulty ciphertexts compared to the extension of Phan and Yen's. Furthermore our attacks need only one attack point (between *MixColumns* of rounds $r-5$ and $r-4$ *or* between *MixColumns* of rounds $r-4$ and $r-3$). However, the extension of Phan and Yen's and general DFAs require two attack points that can be cumbersome for the adversary.

VI. CONCLUSION AND FUTURE WORK

Square-DFA has good advantage over general DFA in exploiting faults induced into some middle rounds. However, the previous Square-DFA requires several hundreds of faulty ciphertexts and works in a bit-level fault model that have hindered the use of it in practice for several years.

In this article, we introduced new Square-DFAs that solved these two problems. When faults are induced between *MixColumns* of rounds $r-4$ and $r-3$, we can find AES-128 key with 46 faulty ciphertexts in a byte fault model. We also extended Square-DFAs to AES-192 and AES-256 where our new attack could find the key with 50 faulty ciphertexts, which is the first result in the literature.

TABLE IV
COMPARISON OF DFAS ON AES-192 AND AES-256

Attack	Fault model	Fault location (Between MCs of)	# of faulty ciphertexts	Time complexity	Remaining candidates
DFA on AES-192 [15]	Byte	(r-3 and r-2) & (r-4 and r-3)	2	2^{32}	1
DFA on AES-256 [15]	Byte	(r-3 and r-2) & (r-4 and r-3)	3	2^{32}	1
App. of Phan and Yen's to AES-192/256	Bit	(r-4 and r-3) & (r-5 and r-4)	1020	2^{16}	1
New Square-DFA on AES-192/256	Bit	r-4 and r-3	255	2^{20}	1
	Byte	r-4 and r-3	50	2^{34}	1

When faults are induced between *MixColumns* of rounds $r-5$ and $r-4$, our attack showed that the time complexity could be reduced from 2^{66} to 2^{42} in a bit model. Unfortunately, we could not find a practical attack in a byte fault model.

Therefore, we can conclude that round $r-4$ should be protected against fault attacks but round $r-5$ has a marginal security. Reducing time complexity in exploiting faults induced at round $r-5$ seems an interesting topic for future research.

REFERENCES

[1] National Institute of Standard and Technology, *Data Encryption Standard*, NIST FIPS PUB 46-2, 1993.

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

[3] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. In *Fault Diagnosis and Tolerance in Cryptography in association with DSN 2004 – The International Conference on Dependable Systems and Networks*, pages 330–342, 2004.

[4] A. Barenghi, G. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi. Low voltage fault attacks to AES and RSA on general purpose processors. IACR eprint archive, 2010-130, 2010.

[5] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[6] J. Blömer and J.-P. Seifert. Fault based cryptanalysis of the advanced encryption standard (AES). In *Financial Cryptography, 7th International Conference, FC 2003*, volume 2742 of *Lecture Notes in Computer Science*, pages 162–181. Springer, 2003.

[7] C.-N. Chen and S.-M. Yen. Differential fault analysis on AES key schedule and some countermeasures. In *Information Security and Privacy, 8th Australasian Conference, ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2003.

[8] H. Chen, W. Wu, and D. Feng. Differential fault analysis on CLEFIA. In *Information and Communications Security, 9th International Conference, ICICS 2007*, volume 4861 of *Lecture Notes in Computer Science*, pages 284–295. Springer, 2007.

[9] J. Daemen, L. R. Knudsen, and V. Rijmen. The Block Cipher Square. In *Fast Software Encryption, 4th International Workshop, FSE '97*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

[10] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems. A practical implementation of the timing attack. In *Smart Card Research and Applications, This International Conference, CARDIS '98*, volume 1820 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 1998.

[11] P. Dusart, G. Letourneux, and O. Vivolo. Differential fault analysis on AES. In *Applied Cryptography and Network Security, First International Conference, ACNS 2003*, volume 2846 of *Lecture Notes in Computer Science*, pages 293–306. Springer, 2003.

[12] T. Fukunaga and J. Takahashi. Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers. In *6th International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2009*, pages 84–92. IEEE Computer Society, 2009.

[13] C. Giraud. DFA on AES. In *Advanced Encryption Standard - AES, 4th International Conference, AES 2004*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2005.

[14] L. Hemme. A differential fault attack against early rounds of (Triple-) DES. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop*, volume 3156 of *Lecture Notes in Computer Science*, pages 254–267. Springer, 2004.

[15] C. H. Kim. Differential fault analysis against AES-192 and AES-256 with minimal faults. In *Fault Diagnosis and Tolerance in Cryptography, 7th International Workshop, FDTC 2010*, pages 3–9. IEEE Computer Society, 2010.

[16] C. H. Kim and J.-J. Quisquater. Faults, injection methods, and fault attacks. *IEEE Design and Test of Computers*, 24(6):544–545, 2007.

[17] C. H. Kim and J.-J. Quisquater. New differential fault analysis on AES key schedule: Two faults are enough. In *Smart Card Research and Advanced Applications, 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008*, volume 5189 of *Lecture Notes in Computer Science*, pages 48–60. Springer, 2008.

[18] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[19] W. Li, D. Gu, and Y. Wang. Differential fault analysis on the contracting UFN structure, with application to SMS4 and Macguffin. *Journal of Systems and Software*, 82(2):346–354, 2009.

[20] A. Moradi, M. T. M. Shalmani, and M. Salmasizadeh. A generalized method of differential fault attack against AES cryptosystem. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 91–100. Springer, 2006.

[21] D. Mukhopadhyay. An improved fault based attack of the advanced encryption standard. In *AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 421–434. Springer-Verlag, 2009.

[22] R. C.-W. Phan and S.-M. Yen. Amplifying side-channel attacks with techniques from block cipher cryptanalysis. In *Smart Card Research and Advanced Applications, 7th IFIP WG 8.8/11.2 International Conference, CARDIS 2006*, volume 3928 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 2006.

[23] 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 Systems - CHES 2003, 5th International Workshop*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.

[24] J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[25] J. Takahashi and T. Fukunaga. Differential fault analysis on the AES key schedule. IACR eprint archive, 2007-480.

[26] J. Takahashi and T. Fukunaga. Improved differential fault analysis on CLEFIA. In *Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2008, FDTC 2008*, pages 25–34. IEEE Computer Society, 2008.

[27] J. Takahashi and T. Fukunaga. Differential fault analysis on AES with 192 and 256-bit keys. IACR eprint archive, 2010-023, 2010.

[28] J. Takahashi, T. Fukunaga, and K. Yamakoshi. DFA mechanism on the AES key schedule. In *4th International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2007*, pages 62–74. IEEE Computer Society, 2007.

[29] M. Tunstall and D. Mukhopadhyay. Differential fault analysis of the advanced encryption standard using a single fault. IACR eprint archive, 2009-575.