Hindawi
Security and Communication Networks
Volume 2018, Article ID 2870475, 9 pages
https://doi.org/10.1155/2018/2870475

Research Article

Towards Optimized DFA Attacks on AES under Multibyte Random Fault Model

Ruyan Wang, Xiaohan Meng, Yang Li, and Jian Wang

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China

Correspondence should be addressed to Yang Li; liyang@uec.ac.jp

Received 10 May 2018; Revised 24 June 2018; Accepted 5 July 2018; Published 13 August 2018

Academic Editor: Xuyun Zhang

Differential Fault Analysis (DFA) is one of the most practical methods to recover the secret keys from real cryptographic devices. In particular, DFA on Advanced Encryption Standard (AES) has been massively researched for many years for both single-byte and multibyte fault model. For AES, the first proposed DFA attack requires 6 pairs of ciphertexts to identify the secret key under multibyte fault model. Until now, the most efficient DFA under multibyte fault model proposed in 2017 can complete most of the attacks within 3 pairs of ciphertexts. However, we note that the attack is not fully optimized since no clear optimization goal was set. In this work, we introduce two optimization goals as the fewest ciphertext pairs and the least computational complexity. For these goals, we manage to figure out the corresponding optimized key recovery strategies, which further increase the efficiency of DFA attacks on AES. A more accurate security assessment of AES can be completed based on our study of DFA attacks on AES. Considering the variations of fault distribution, the improvement to the attack has been analyzed and verified.

1. Introduction

In the age of IoT, IoT technologies can widely perceive our physical world and generate sensing data for further research. There are lots of scenarios in IoT where people have to collaborate through devices to complete tasks; for example, a device sends data to other devices [1], or one user shares EHR in mobile cloud computing [2], and these transmitted data are often the privacy data of users. At the same time, in the big data environment [3, 4], many enterprises need to constantly assimilate big data knowledge and private knowledge by multiple knowledge transfers to maintain their competitive advantage [5]. Thus, the protection of data is especially important during the transmission and encryption of data. However, in recent years, attackers increasingly have access to various cryptographic algorithms. In most cases, attackers develop fault attacks [6] on cryptographic devices and then the private information is leaked. Thus, a lot of sensitive data suffer from severe security and privacy threats.

In general, security and privacy protection are crucial in the field of cloud, fog, or IoT [7–9]. The basis of the security mechanism is the implementation of the cryptosystem. It should be pointed out that the security of cryptosystem includes not only design security but also implementation security. In several ways to assess the implementation security, fault attack is a common method. By studying fault attacks, researchers can evaluate the security of cryptographic algorithms and provide ideas for strengthening protection of sensitive data. This work focuses on the security assessment of AES in fault attacks, which is the most common algorithm in a block cipher system. Among numerous fault attacks, DFA is one of the most practical methods to retrieve the secret key and has become a wide research topic in many fields. Although DFA attacks have been successfully applied to AES, the attack process requires a certain number of faulty ciphertexts or a large key search space. How to reduce the number of faulty ciphertexts required or the search space of keys for attack is a hot research topic.

In this paper, we propose two optimization goals and corresponding strategies. One goal is completing a DFA attack on AES with the fewest ciphertext pairs, and the other is completing a DFA attack on AES with the least computational complexity. The DFA attacks using our strategies can realize the goal of the fewest ciphertext pairs or the least computational complexity, respectively. The optimized DFA attacks in this work take fewer resources and can be completed

faster, achieving higher efficiency. As a result, a more accurate security assessment of AES can be completed based on our work. An earlier version of this paper was presented at the International Conference on Cloud Computing and Security (ICCCS 2018).

The rest of this paper is organized as follows: In Section 2, we introduce the related work proposed by predecessors. Section 3 explains a classical DFA on AES and Liao's method in [10]. Two strategies applied to DFA attacks on AES we propose are introduced in Section 4. The theoretical analysis of our method is given in Section 5 and we conclude in Section 6.

2. Related Work

The concept of DFA was first introduced in [11] in 1996. The principle of DFA is to induce faults (unexpected environmental conditions) into cryptographic implementations to reveal their internal states. In 2003, Gilles Piret and JeanJacques Quisquater described a DFA attack technique [12] and could break the AES-128 with only 2 faulty ciphertexts, assuming the fault is in MixColumns operation of the eighth or ninth round. In 2004, Christophe Giraud proposed two different DFA attacks on AES [13]. The first one induces a fault to only one bit of an intermediate result and the key can be obtained with 50 faulty ciphertexts for AES-128. While the second one induces a fault to a whole byte and less than 250 faulty ciphertexts are needed for key recovery for AES-128. In [14] in 2011, Tunstall, Mukhopadhyay, and Ali proposed a two-stage algorithm of DFA that could recover the AES 128-bit key using one fault injection. However, without plaintext-ciphertext exhaustive search, the most efficient DFA key recovery on AES-128 with a single-byte fault requires 2 pairs of ciphertexts [15]. In terms of DFA attacks on AES with a random multibyte fault, the existing literature shows that 6 pairs of ciphertexts are required to develop the attack [16]. In particular, in extreme cases that the injected faults are four-byte ones, attackers need 1500 pairs of ciphertexts for key recovery.

In 2017, Nan Liao et al. [10] proposed an improved DFA attack method on AES with unknown and random faults. They focused on multibyte faults whose locations and values are unknown to the attackers. The fault model in their work combined the single-byte fault model and multibyte fault model and took both accuracy and efficiency into considerations. Their experimental results showed that most of the attacks could be completed within 3 pairs of ciphertexts. After that, a hybrid model was proposed in [17] to improve availability of ciphertext for DFA against AES and 6 pairs of correct and faulty ciphertexts could recover the secret key of AES-128. In [17], the attack models available for analysis include single-byte random faults in encryption process, multibyte random faults in encryption process, and single-byte faults in key schedule. In addition, one improved DFA attack using all-fault ciphertexts on AES was proposed in [18]. The all-fault ciphertexts were used to optimize the selection of the brute-force space, helping to recover the secret key quickly and improve the analysis efficiency. Their experiment result demonstrated that the time consumed on the brute-force attack could be reduced 60.81% on average.

3. DFA on AES

3.1. Generic Fault Model. Two kinds of fault models are widely used in most of DFA attacks on AES, which are single-byte fault model and multibyte fault model. In this paper, multibyte fault model assumes that the size of the injected fault ranges from one byte to three bytes in one column of AES state. The four-byte faults are not discussed in this work since they are not as useful as others in the key recovery, also they can be omitted in practical fault injections. When some techniques like laser beam [19] are used to induce faults, the fault can be fixed to single byte and the specific location of the fault can even be selected. However, when other techniques are used, such as supply voltage variation [20] and clock glitch injection [21], the size of the fault may be more than one byte, and attackers cannot control the location. It should be noted that though the fault injection techniques like laser beam enable attackers to control the characteristics of the fault, they are sophisticated and high-cost. On the contrary, fault injection techniques such as supply voltage variation and clock glitch injection are noninvasive and they need lower cost, which are more practical.

Therefore, this research focuses on the more general fault model, which is multibyte fault model since the methods to induce multibyte faults are more practical. In addition, it is necessary to introduce one kind of fault model that combines the advantages of the two fault models.

3.2. Basic Key Recovery in DFA on AES. For AES that consists of 10 round functions, DFA attacks usually target the last two rounds. When the fault is injected to the last two rounds, the fault only affects some bytes of the ciphertext. Therefore, it is feasible to retrieve the key by analyzing the differential value of the corresponding parts of the correct and faulty ciphertext.

Assume that a single-byte fault is injected to the first column of the state after `ShiftRows` operation of R_9. After R_9, the fault spreads to the entire column. After R_{10}, which omits `MixColumns` operation, the fault affects specific four bytes of the output. In other words, only four bytes of the ciphertext will be affected by the injected fault. Also, the locations of four bytes are determined by the location of the initial fault. Attackers can make assumptions about the four-byte round-key of R_{10} in affected locations and verify whether the fault information derived is consistent with the fault model.

The specific key recovery process is as follows: assuming key values and calculating (1)-(3), attackers can achieve two internal states after `ShiftRows` operation of R_9, respectively, from the ciphertext pair. Calculating their difference and comparing the information with the fault model, those incorrect key assumptions are eliminated. In (1)-(3), δ is the difference of the correct and faulty ciphertexts; K_9, K_{10} denote the round-keys of R_9, R_{10}; S_9, S_{10} denote the internal states after `AddRoundKeys` operation of R_9, R_{10}; $S^{\sim}{}_9$, $S^{\sim}{}_{10}$ denote the corresponding states in faulty encryptions. *InvMC*, *InvSB,* and *InvSR* are the inverse processes for `MixColumns`, `SubBytes`, and `ShiftRows` operations. We can have (1) based on the characteristics of the encryption functions of

Table 1: Some notations and their meanings.

Notations	Meaning
$P_{candidates}$	The proportion of the number of candidate keys to the number of all possible keys
N_{can_key}	The number of candidate keys after the first analysis in DFA process
P_{can_1b}	$P_{candidates}$ for the single-byte fault model
P_{can_2b}	$P_{candidates}$ for the two-byte fault model
P_{can_3b}	$P_{candidates}$ for the three-byte fault model

AES. Since `MixColumns` is not included in R_{10}, we can get S_9 and $S\tilde{}_9$ as shown in (2)-(3).

$$\begin{aligned}\delta &= InvMC\left(K_9 \oplus S_9\right) \oplus InvMC\left(K_9 \oplus S_9^{\sim}\right) \\ &= InvMC\left(S_9 \oplus S_9^{\sim}\right),\end{aligned} \tag{1}$$

$$S_9 = InvSB\left(InvSR\left(K_{10} \oplus S_{10}\right)\right), \tag{2}$$

$$S_9^{\sim} = InvSB\left(InvSR\left(K_{10} \oplus S_{10}^{\sim}\right)\right). \tag{3}$$

If the injected fault is multibyte, the circumstance is almost the same. Though the outputs of two fault diffusion processes are identical, the numbers of ciphertext pairs required for key recovery under two kinds of models differ. In the case of single-byte faults, 2 pairs of ciphertexts are enough to retrieve four bytes of the round-key [15]. However, in the case of multibyte faults, 6 pairs of ciphertexts are required [16].

3.2.1. DFA Method Proposed by Nan Liao et al. In 2017, Nan Liao et al. proposed improved DFA attacks on AES with multibyte faults [10]. Since our method is based on their contributions, their method is introduced first. They classified faults into four types according to the number of faulty bytes. In their attack, four-byte faults are not under discussion since four-byte faults hardly appear in real attacks. The occurrence rate of the fault type is denoted as P_t, t denotes the number of faulty bytes, and $t \in [1, 3]$. The notations used are provided in Table 1. $P_{candidates}$ denotes the proportion of the number of candidate keys to the number of all possible keys, which is approximate to the proportion of the number of covered faults to the number of all possible faults. N_{can_key} is defined as multiplying $P_{candidates}$ and the number of all possible keys N_{all}, which is shown in

$$N_{can\_key} = N_{all} \times P_{candidates}. \tag{4}$$

The amount of all possible round-keys is always 2^{32} for the first analysis in DFA process, corresponding to analyzing the first ciphertext pair in DFA process. Every time an analysis in DFA process is completed, the amount of candidate keys is decreased.

For the single-byte fault model, the proportion of the number of covered faults to the number of all possible faults is

$$\frac{C_4^1 \times 255^1}{2^{32} - 1} = 2.37 \times 10^{-7}. \tag{5}$$

As mentioned above, P_{can_1b} is defined as

$$P_{can\_1b} \approx 2.37 \times 10^{-7}. \tag{6}$$

Similarly, P_{can_2b} and P_{can_3b} are defined as follows:

$$P_{can\_2b} \approx \frac{C_4^2 \times 255^2}{2^{32} - 1} = 9.08 \times 10^{-5}, \tag{7}$$

$$P_{can\_3b} \approx \frac{C_4^3 \times 255^3}{2^{32} - 1} = 0.0154. \tag{8}$$

P_{can_1b} is so small that only 2 pairs of ciphertexts are enough to retrieve the round-key. Similarly, 3 pairs of ciphertexts are required for two-byte faults and 6 pairs of ciphertexts are required for three-byte faults. The theoretical candidate numbers in three fault models after each analysis are, respectively, given in Table 2.

It is claimed in [10] that $P_{candidates}$ decides the number of ciphertext pairs required in the DFA attack. If small $P_{candidates}$ like P_{can_1b} is used in the attack, the number of ciphertext pairs required will be greatly reduced and the efficiency of the attack will be increased.

It can be found that if the fault type is known to attackers, they are able to use the consistent fault model to complete the DFA attack, leading to fewer ciphertexts required. Especially when single-byte faults occur frequently, fewer ciphertext pairs are needed. However, in the practical environment, attackers have no idea about the fault type. They can only assume the fault type and verify the correctness.

Similar to the analysis process under multibyte model in [16], Nan Liao et al. considered three fault types without four-byte faults. The biggest difference between their methods is that Liao's method divided faults into three types in each analysis and calculated, respectively. Nan Liao et al. refined the object of each analysis and obtained more detailed information after each analysis. They found that many fault type series could be analyzed with 2-5 pairs of ciphertexts. Thus, it is suggested in Liao's method to give priority to assuming and verifying the fault type series that need less ciphertext pairs. Only one candidate left after DFA process means the remaining candidate is the correct round-key. No candidate left means the assumed fault type series is wrong.

Figure 1 shows all possibilities of fault type series that need less than 6 pairs of ciphertexts. The line connecting two oval frames is defined as a path since it represents one possible situation of fault type series. All figures in the frames represent the number of key candidates after the last analyses including three fault situations. Take 1024615 in the oval frame in the left column for example. After the second analysis under three-byte fault model, the number of key candidates is $66142496 \times P_{can\_3b} + 389983 \times P_{can\_2b} + 1020 \times P_{can\_1b} \approx 1024615$. Thus, the figure in the oval frame is the sum of the results under three situations. For the dotted line path, the number of key candidates after analysis is more than 1, which means analysis needs to be continued to recover the key. For the red solid line path, the number of key candidates after analysis is close to 0. For example, the rightmost red path represents $1020 \times P_{can\_1b} = 0.00024$. If the path is consistent with the real fault type series at this point, then only the

Table 2: The theoretical candidate numbers after each analysis in three fault models.

Analyses completed	1	2	3	4	5	6
Single-byte fault model	1020	0.00024	0	0	0	0
Two-byte fault model	389983	35	0.003	0	0	0
Three-byte fault model	66142496	1018594	15686	241	3.72	0.06

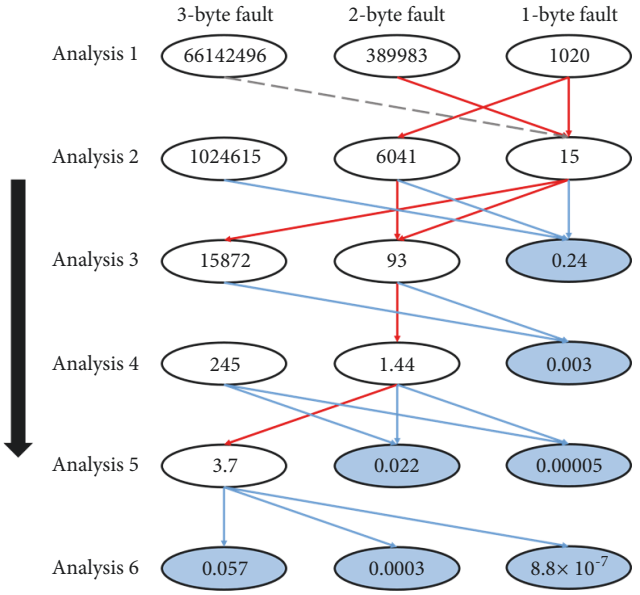

Figure 1: The possible situations that can identify the key within 6 pairs of ciphertexts [10].

correct round-key survives. For the blue oval frames, the figures in these frames are already close to 0. In Liao's method, they defined the paths that can retrieve the round-key within 5 pairs of ciphertexts as exploitable paths and give priority to assuming and verifying exploitable paths. When the real fault type series are consistent with these paths, the remaining key candidate is the correct round-key and the attack can be completed quickly.

Nan Liao et al. verified the effectiveness of their method by experiments. They collected 12000 ciphertexts and faults are injected in the same column in the same round for these ciphertexts. The 12000 ciphertexts are divided into 2000 groups and each group consists of 6 ciphertexts. If multibyte fault model is used, 6 ciphertext pairs in each group will all be exploited to recover the round-key. If Liao's method is used, there is a big probability that less than 6 ciphertext pairs are enough for the round-key recovery.

4. Proposed DFA Method and Application on AES

4.1. Inspirations from [10]. In [10], the attack successfully recovers the correct round-key with 2-5 pairs of ciphertexts in most cases. However, Liao's method is not perfectly reasonable, especially the method of choosing exploitable paths. It should be noted that, in Figure 1, the number in the oval frame is the cumulative sum of the previous analysis results of three fault situations. Thus, it is incomplete to determine exploitable paths through the number in the oval frame. Some paths will be missed if the cumulative sum is considered barely. For instance, fault type series "223$\cdots$" only needs 3 pairs of ciphertexts to be verified, but it is not included in exploitable paths in Liao's method. The reason for missing paths is that they add up the number of key candidates from three situations and make an analysis on the sum, leading to neglect of the number of key candidates after each analysis for a single path. As a result, some paths that can retrieve the round-key within 5 pairs of ciphertexts are eliminated and more calculations are required. In order to avoid such omissions, we further refine the analysis process and discuss one fault type at a time. That is to say, one path is taken as the unit of analysis instead of three fault types being discussed at the same time.

In addition, the goal in [10] is not clear enough. The authors stressed that their method could retrieve the round-key with fewer ciphertext pairs and the least computation. They mixed required ciphertext pairs and computation together for discussion, which made the goal ambiguous.

Generally, we find that the goal in [10] is not clear and their strategy is optimizable. We first set the optimization goal and then develop the method to find the optimized key recovery strategy. Our improvement will be introduced in the following section. Note that although this work mainly focuses on AES, our work can be easily adapted to DFA attacks on other ciphertexts.

4.2. Improved DFA Attack on AES under Multibyte Random Fault Model. The following content is our improved DFA attack on AES, showing great advantages compared with the previous DFA attacks.

We denote m_i as the amount of key candidates after i^{th} analysis in DFA process. P_{can_tb} is the proportion of the key candidates after one analysis in DFA process under t-byte fault model. The theoretical m_i value is calculated as

$$m_i = m_{i-1} \times P_{can\_tb}, \quad i \in [1,6], \ t \in [1,3]. \tag{9}$$

The amount of key candidates decreases after each analysis. When $i = 0$, $m_0 = (2^8)^4 = 2^{32}$, which means the initial amount of key candidates before the first analysis in DFA process is always 2^{32}. When $i = 6$, m_6 must be zero based on the fact that the key can be determined with 6 pairs of ciphertexts under multibyte fault model.

The procedure of our attack method is as follows:

(1) Obtain the correct ciphertext and several faulty ciphertexts.

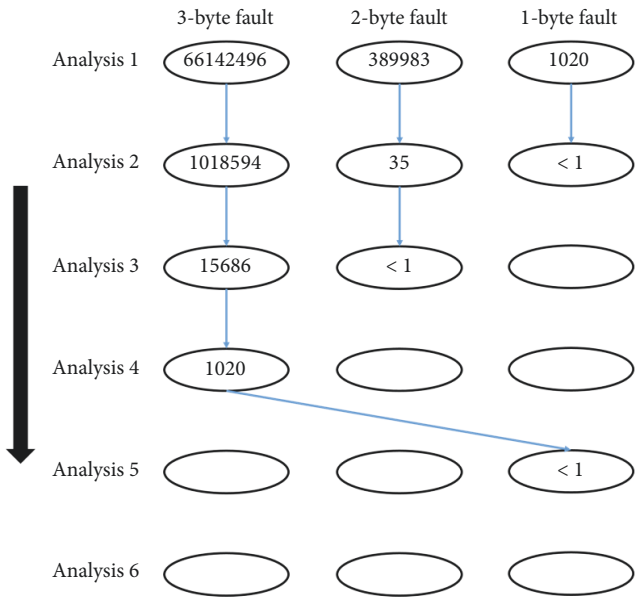

Figure 2: The DFA process of some paths.

(2) Choose one path assumption, analyze the ciphertext pair, and verify the assumption.

(3) After i^{th} analysis, if $m_i = 1$, then the key candidate is the correct round-key.

(4) After i^{th} analysis, if $m_i = 0$, then repeat (2) until the correct round-key is recovered.

According to three fault types and the maximum ciphertext pairs described earlier, there are 3^6 possible paths for AES totally. We review each path and calculate the amount of key candidates after each analysis in DFA process to determine the fewest ciphertext pairs required. A figure for better understanding is shown as Figure 2. It shows all intermediate results of analyses in DFA process for three paths: "33331$\cdots$"; "22$\cdots$"; and "11$\cdots$." The number in the oval frame stands for the amount of remaining key candidates after the last analysis for the current path. Different from Figure 1, the number in the oval frame is related to only one path, which is the current path being verified. In our method, it is intuitive and accurate to see the fewest ciphertext pairs each path required.

When guessing and verifying the paths, different assuming orders lead to different computational complexity and different numbers of ciphertext pairs required. Given a fault model and a specific goal, it is possible to determine an optimized strategy for 3^6 paths. In this paper, we consider two specific goals to be optimized, which are recovering the round-key with the fewest ciphertext pairs and the least computational complexity.

Key Recovery with Fewest Ciphertext Pairs. If the goal is to recover the round-key with the fewest ciphertext pairs, the strategy of assuming paths is as follows. First, list all paths that require 2, 3, 4, 5, and 6 pairs of ciphertexts, respectively. Afterwards, start from assuming and verifying the paths that require 2 ciphertext pairs; if only one key candidate is left after DFA process, this candidate is viewed as the correct round-key. Otherwise, we keep verifying paths until we retrieve the round-key, with the order of **2 ciphertexts** $\longrightarrow$ **3 ciphertexts** $\longrightarrow$ **4 ciphertexts** $\longrightarrow$ **5 ciphertexts** $\longrightarrow$ **6 ciphertexts**. When there is more than one path that can be analyzed with the same number of ciphertext pairs, we preferentially verify the path whose occurrence rate is higher. In other words, the first priority is less ciphertext pairs required, and the second priority is higher occurrence rate. Finally, when round-key is recovered, the ciphertext pairs used are consistent with the actual ciphertext pairs the path requires. The process of sorting all 3^6 paths is shown in Algorithm 1. After Algorithm 1, φ is a set of 3^6 paths in an order of less ciphertext pairs required to more ciphertext pairs required.

For the first strategy, the overview of the DFA attack is shown in Figure 3.

Key Recovery with Least Computational Complexity. If the goal is to recover the round-key with the least computational complexity, we need to take P_1, P_2, and P_3 into consideration and reaffirm the order of assuming paths. For the purpose of reducing computational complexity, one needs to eliminate duplicated calculations by saving the internal values and key candidates that survive the analysis. When later analysis requires the same data, the information saved can be directly exploited without recalculations. Furthermore, we should consider the occurrence rate of each path to determine the order of assuming paths. However, it is not sufficient to make decisions based on the occurrence rates of the paths. The calculations cost for different fault types in each analysis are also an important part. Hence, we introduce the concept of cost-efficiency, which is the determinant for each analysis. We define *ce* as follows:

$$ce = \frac{P_t}{P_{can\_tb}}, \tag{10}$$

where t is the type of fault. It contains two variables that are the occurrence rate of some fault type and $P_{candidates}$ corresponding to that fault type. When the occurrence rate of some fault type is higher, this fault type appears more in actual situations. That means it is more likely to find the correct path quickly if we first assume and verify this fault type. Thus, the higher the occurrence rate of the fault type is, the higher the priority should be. When $P_{candidates}$ for some fault type is smaller, it means the analysis corresponding to the fault type can pick out fewer key candidates, leading to less computation. Thus, the smaller $P_{candidates}$ for the fault type is, the higher the priority should be. Therefore, we consider two variables in total that are closely related to the second goal and it is reasonable to determine the order of assuming paths relying on the variable *ce*. Before the first analysis in DFA process, we calculate *ce* for all fault types and determine the order of assuming fault types based on *ce*. The greater *ce* is, the more the fault type is worth being verified preferentially.

The specific approach is as follows. Before the first analysis in DFA process, *ce* for three fault types are calculated and the order of assuming fault types is determined. In each analysis, we first verify the fault type whose *ce* is the greatest and next the second great and last the left. When one analysis in DFA process is finished and the amount of key candidates is far greater than 1, we continue assuming and verifying fault

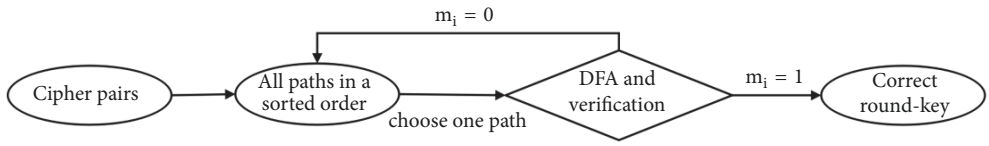

Figure 3: The attack method.

Input: φ: 3^6 paths of disorder.
Output: φ: ordered 3^6 paths.
(1) **for** each $j \in [2, 6]$ **do**
(2) initialize $\psi_j = \Phi$
(3) **end for**
(4) **for** each $p \in \varphi$ **do**
(5) $j \longleftarrow$ the number of ciphertext pairs p requires
(6) $\psi_j = p \cup \psi_j$
(7) **end for**
(8) **for** each $j \in [2, 6]$ **do**
(9) sort paths in ψ_j in order of occurrence rate from low to high
(10) **end for**
(11) **return** $\varphi \longleftarrow \psi_2 \parallel \psi_3 \parallel \psi_4 \parallel \psi_5 \parallel \psi_6$

Algorithm 1: The process of sorting all 3^6 paths.

type. After several analyses, if the amount of key candidates is 1, the current path matches the real situation and the left candidate is the round-key. If the amount of key candidates is 0, it means the current path assumed is wrong. At this point, we have to turn back to the previous analysis and assume the next fault type. Similar to the first strategy, the analysis above can be viewed as assuming ordered paths from the macroscopic angle. However, the order of assuming paths here is connected with *ce* for three fault types. For better understanding, a detailed example of the strategy is given. Suppose that the situation of three fault types in practical environment is $P_1 > P_2 > P_3$, and the fault type series is "131221." First, *ce* for three fault types are calculated in advance. The values of *ce* for three fault types are denoted as ce_t, and t denotes the fault type.

$$ce_1 = \frac{P_1}{2.37 \times 10^{-7}},$$
$$ce_2 = \frac{P_2}{9.08 \times 10^{-5}}, \quad (11)$$
$$ce_3 = \frac{P_3}{0.0154}.$$

It is known that $P_1 > P_2 > P_3$, so clearly $ce_1 > ce_2 > ce_3$. According to the second strategy, we first assume single-byte faults and then two-byte faults and three-byte faults at last. After the first analysis, the amount of key candidates decreases from 2^{32} to 1020. Since the amount is not 1, we continue assuming that the fault is single-byte. After the second analysis, the amount of key candidates is 0, which means the current path "11$\cdots$" is wrong. So we turn back to the second analysis and assume that the fault is two-byte one. There are 0 key candidates after the second analysis, which means the current path "12$\cdots$" is also wrong. Similarly, we turn back and assume that the fault is three-byte. This time, 15 key candidates are left after the second analysis. We continue assuming that the fault is single-byte. After the third analysis, only one key candidate survives. Obviously, the key candidate left is the round-key and the fault type series is "131$\cdots$." The computation in the analyses above is significantly less than that in 6 analyses under multibyte fault model.

5. Results and Discussion

This section analyzes the two strategies proposed and they are more efficient than the previous attacks. In the following, the first strategy is called data-complexity priority strategy and the second strategy is called computation-complexity priority strategy.

5.1. Data-Complexity Priority Strategy. As mentioned earlier, 6 pairs of ciphertexts are required in DFA attacks under multibyte fault model. However, in our method using data-complexity priority strategy, most of the DFA attacks can be completed within 5 pairs of ciphertexts.

Here the comparison of the expected amount of used ciphertext pairs between our method and Liao's method is also given. For all 3^6 paths, we denote the expected amount of used ciphertext pairs as *E(r)*. It is the accumulation of the product of the occurrence rate and used faulty ciphertext pairs for all paths. In the case of known fault type,

$$E(r) = \sum P_{path} \times r, \quad (12)$$

in which r denotes the actual ciphertext pairs that *path* requires ($2 \le r \le 6$), and P_{path} denotes the occurrence rate of

path. Take path "11⋯" as an example; its occurrence rate is $P_{11\ldots} = P_1 \times P_1$.

In Liao's method, the number of used ciphertext pairs is more than the actual required ciphertext pairs for *path*. Take fault type series "331⋯" as an example; it only needs 3 pairs of ciphertexts to be verified. But, in Liao's method, it will not be verified until all exploitable paths have been verified since it is not included in exploitable paths. In this example, they will use 6 pairs of ciphertexts to complete the attack. So the expected amount of used ciphertext pairs in Liao's method is greater than $E(r)$.

In our method using data-complexity priority strategy, the number of used ciphertext pairs always equals to the actual required ciphertext pairs for *path*. That is to say, the expected amount of used ciphertext pairs in our method equals to $E(r)$. According to the expected amount of used ciphertext pairs, we can clearly see that our method using data-complexity priority strategy is better than Liao's method.

In [10], authors give the probability of attacks that succeed within 3 pairs of ciphertexts. In Liao's method, the fault type series that can be successfully verified with 2 pairs of ciphertexts are "11⋯", "12⋯", "21⋯"; the fault type series that can be successfully verified with 3 pairs of ciphertexts are "311⋯", "312⋯", "313⋯", "321⋯", "331⋯", "221⋯", "222⋯", "231⋯". Hence, the probability of attacks that succeed within 3 pairs of ciphertexts are denoted as follows.

(1) The probability of attacks that succeed with 2 pairs of ciphertexts is

$$P_{2ciphertexts} = P_1^2 + 2 \bullet P_1 \bullet P_2. \tag{13}$$

(2) The probability of attacks that succeed with 3 pairs of ciphertexts is

$$P_{3ciphertexts} = P_1 \bullet P_3 \bullet (P_3 + 1) + P_2^2 \bullet (P_1 + P_2) + 2 \bullet P_3 \bullet P_2 \bullet P_1. \tag{14}$$

(3) Thus the probability of attacks that succeed within 3 pairs of ciphertexts is

$$P_{2ciphertexts+3ciphertexts} = P_{2ciphertexts} + P_{3ciphertexts}. \tag{15}$$

In our method using data-complexity priority strategy, the fault type series that can be successfully verified with 2 pairs of ciphertexts are the same as those in Liao's method. Besides those in Liao's method, we can verify more fault type series within 3 pairs of ciphertexts: "131⋯"; "132⋯"; "133⋯"; "223⋯"; "232⋯"; "322⋯." Hence, the probability of attacks that succeed within 3 pairs of ciphertexts in our method is

$$P_{3ciphertexts} = P_1 \bullet P_3 \bullet (2 + P_3 + 2 \bullet P_2) + P_2^2 \bullet (1 + 2 \bullet P_3). \tag{16}$$

Suppose that the probabilities of occurrence are $P_1 = 0.89$, $P_2 = 0.1$, and $P_3 = 0.01$, respectively. In Liao's method, $P_{2ciphertexts+3ciphertexts} = 99.15\%$. In our method using data-complexity priority strategy, $P_{2ciphertexts+3ciphertexts} = 99.997\%$. In this case, it can be seen that the probability of attacks using our strategy that succeed within 3 pairs of ciphertexts is slightly greater than theirs: 99.997% > 99.15%. Then suppose that $P_1 = 0.4$, $P_2 = 0.3$, and $P_3 = 0.3$, respectively. In Liao's method, the probability of attacks that succeed within 3 pairs of ciphertexts is 72.7%. In our method using data-complexity priority strategy, the probability of attacks that succeed within 3 pairs of ciphertexts is 92.8%. The probability of attacks that succeed within 3 pairs of ciphertexts using our strategy is greater than theirs by 20.1%. Generally, it is obvious that our data-complexity priority strategy is better than Liao's method.

5.2. Computation-Complexity Priority Strategy. The following gives the analysis of the computational complexity in DFA attacks using computation-complexity priority strategy.

In this work, for each possible key value, we calculate the same basic key recovery in DFA (see (1)-(3)), so the computation in one analysis in DFA process is fixed. Therefore, the computation in a DFA attack actually is equivalent to the number of times of calculating (1)-(3), which is the sum of the amount of key candidates in each analysis. Hence, we can use the cumulative sum of the amount of key candidates per step to represent the computation.

For the multibyte fault model in [16], the times of calculating (1)-(3) are

$$2^{32} + 66571993 + 1031865 + 15993 + 247 + 3.84 \approx 2^{32} + 2^{26.01}. \tag{17}$$

So the computation in DFA attacks under multibyte fault model is fixed, which can be represented as $2^{32}+2^{26.01}$ for comparison with computation in other DFA attacks.

For computation-complexity priority strategy, the computation in a DFA attack is closely related to the occurrence rates of three fault types. In order to compare them in a more intuitive way, we give 3 possible situations about the occurrence rates of three fault types. In each situation, we calculate the product of the occurrence rate of each path and the total computation used for the path, and then we add up the product values of all paths. The corresponding average computation is shown in Table 3.

As we can see in Table 3, the average computation for three situations of three fault types is all less than the fixed value for the general multibyte fault model. It can be found from the table that, in the case of P_1 being much greater than P_2 and P_3, the computation in DFA attacks is the least and the computation decreases a lot compared to the fixed value. At this time, DFA attacks are the most successful. In practical attack scenarios, this strategy can be exploited when single-byte faults appear frequently. The computation-complexity priority strategy can help attackers to reduce the computation in DFA attacks to a certain extent.

6. Conclusion

This work contributes to a more accurate security evaluation for the DFA attack under the multibyte random fault model. Previous work proposed the idea to take advantages of

Table 3: Average computation for different situations of three fault types.

Occurrence rate	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3
	0.7	0.2	0.1	0.2	0.7	0.1	0.1	0.2	0.7
Average computation	$2^{32} + 2^{22.68}$			$2^{32} + 2^{22.72}$			$2^{32} + 2^{25.48}$		

exploitable faults for better key recovery efficiency. To define better efficiency of DFA attack, this work introduces two optimization goals as the fewest ciphertext pairs and the least computational complexity. With different optimization order of these two goals, we propose the corresponding optimized key recovery strategies. Using AES as the analysis target, the improvement compared with previous work has been verified theoretically and with examples. We focus on the security assessment of AES in fault attacks and thus provide reference for the protection of private data. In future work, researchers may apply the approach of optimization to DFA attacks on other ciphers and investigate the attack process and efficiency in detail. In addition, the computation-complexity priority strategy may be further optimized to achieve higher efficiency of the attack.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by National Natural Science Foundation of China (no. 61602239) and Jiangsu Province Natural Science Foundation (no. BK20160808).

References

[1] C. Wang, J. Shen, Q. Liu, Y. Ren, and T. Li, "A Novel Security Scheme Based on Instant Encrypted Transmission for Internet of Things," *Security and Communication Networks*, vol. 2018, Article ID 3680851, 7 pages, 2018.

[2] Z. Cai, H. Yan, P. Li, Z.-A. Huang, and C. Gao, "Towards secure and flexible EHR sharing in mobile health cloud under static assumptions," *Cluster Computing*, vol. 20, no. 3, pp. 2415–2422, 2017.

[3] Y. Wen, J. Liu, W. Dou, X. Xu, B. Cao, and J. Chen, "Scheduling workflows with privacy protection constraints for big data applications on cloud," *Future Generation Computer Systems*, 2018.

[4] X. Xu, X. Zhao, F. Ruan et al., "Data Placement for Privacy-Aware Applications over Big Data in Hybrid Clouds," *Security and Communication Networks*, vol. 2017, Article ID 2376484, 15 pages, 2017.

[5] C. Wu, "Time optimization of multiple knowledge transfers in the big data environment," *Computers, Materials and Continua*, vol. 03, 2018.

[6] B. Dan, R. A. Demillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *International Conference on Theory and Application of Cryptographic Techniques*, pp. 37–51, 1997.

[7] Y. Cao, Z. Zhou, X. Sun, and C. Gao, "Coverless information hiding based on the molecular structure images of material," *Computers, Materials and Continua*, vol. 54, no. 2, pp. 197–207, 2018.

[8] J. Li, X. Chen, X. Huang et al., "Secure distributed deduplication systems with improved reliability," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3569–3579, 2015.

[9] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, and I. You, "Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing," *Information Sciences*, vol. 379, pp. 42–61, 2017.

[10] N. Liao, X. Cui, K. Liao, T. Wang, D. Yu, and X. Cui, "Improving DFA attacks on AES with unknown and random faults," *Science China Information Sciences*, vol. 60, no. 4, 2017.

[11] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Proceedings of the International Cryptology Conference*, pp. 513–525, 1997.

[12] G. Piret and J. J. Quisquater, "A differential fault attack technique against spn structures, with application to the aes and khazad," in *Proceedings of the Cryptographic Hardware and Embedded Systems - CHES 2003, International Workshop*, pp. 77–88, Cologne, Germany, 2003.

[13] C. Giraud, "Dfa on aes," Cryptology ePrint Archive 2003/008, 2003, http://eprint.iacr.org/.

[14] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," in *Proceedings of the WISTP*, pp. 224–233, 2011.

[15] D. Mukhopadhyay, "An improved fault based attack of the advanced encryption standard," in *Proceedings of the AFRICACRYPT*, pp. 421–434, 2009.

[16] A. Moradi, M. T. M. Shalmani, and M. Salmasizadeh, "A generalized method of differential fault attack against aes cryptosystem," in *Proceedings of the CHES*, pp. 91–100, 2006.

[17] Y. Liu, X. Cui, J. Cao, and X. Zhang, "A hybrid fault model for differential fault attack on AES," in *Proceedings of the 2017 IEEE 12th International Conference on ASIC (ASICON)*, pp. 784–787, Guiyang, October 2017.

[18] Y. Ni, X. Cui, T. Wang et al., "Improving DFA on AES using all-fault ciphertexts," in *Proceedings of the 12th IEEE International Conference on Advanced Semiconductor Integrated Circuits, ASICON 2017*, pp. 283–286, October 2017.

[19] M. Agoyan, J. Dutertre, A. Mirbaha, D. Naccache, A. Ribotta, and A. Tria, "Single-bit DFA using multiple-byte laser fault injection," in *Proceedings of the 2010 IEEE International Conference on Technologies for Homeland Security (HST 2010)*, pp. 113–119, November 2010.

[20] A. Barenghi, G. M. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi, "Low voltage fault attacks to AES," in *Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented*

Security and Trust (HOST), pp. 7–12, Anaheim, CA, USA, June 2010.

[21] N. Selmane, S. Guilley, and J.-L. Danger, "Practical setup time violation attacks on AES," in *Proceedings of the 7th European Dependable Computing Conference, EDCC-7*, pp. 91–98, May 2008.