Reduction in the Number of Fault Injections for Blind Fault Attack on SPN Block Ciphers

著者（英）	Yang Li, Mengting Chen, Zhe Liu, Jian Wang
journal or publication title	ACM Transactions on Embedded Computing Systems
volume	16
number	2
page range	1-20
year	2017-04
URL	http://id.nii.ac.jp/1438/00009014/

doi: 10.1145/3014583

Reduction in Number of Fault Injections for Blind Fault Attack on SPN Block Ciphers

Yang Li, Mengting Chen, Zhe Liu, Jian Wang, Nanjing University of Aeronautics and Astronautics

In 2014, a new fault analysis called blind fault attack (BFA) was proposed, in which the attackers can only obtain the number of different faulty outputs without knowing the public data. The original BFA requires 480,000 fault injections to recover a 128-bit AES key. This work attempts to reduce the number of fault injections under the same attack assumptions. We analyze BFA from an information theoretical perspective and introduce a new probability-based distinguisher. Three approaches are proposed for different attack scenarios. The best one realized a 66.8% reduction of the number of fault injections on AES.

CCS Concepts: •**Security and privacy** → **Side-channel analysis and countermeasures;** *Block and stream ciphers;* Embedded systems security;

Additional Key Words and Phrases: Blind fault attack, fault analysis, AES, information theory

ACM Reference Format:
Yang LI, Zhe Liu and Jian Wang. 2016. Reduction in Number of Fault Injections for BFA on SPN Block Ciphers. *ACM Trans. Embedd. Comput. Syst.* 1, 1, Article 1 (January 2016), 20 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Cryptographic algorithms implemented in embedded systems are under the threats of various physical attacks such as side-channel attacks and fault injection attacks. This work focuses on a new type of fault injection attack called blind fault attack (BFA), which was proposed in 2014 [Korkikian et al. 2014]. The purpose of this work is to accurately evaluate the number of fault injections for BFA key recovery.

In fault injection attacks, attackers intentionally inject computational fault into the executing cryptographic algorithms. Usually, via fault injections, the faulty outputs and the faulty behaviors of the target embedded device are observed. These fault-based information is used together with some public data, *i.e.* plaintext and ciphetext, to recover the key. Different from all previous attacks, the BFA attacker is assumed to be blind to the values of public data. Thus, in the BFA key recovery, the attackers can only rely on the information gained from fault injections without any "main-channel" information. Previous famous fault attacks, *e.g.* differential fault analysis [Boneh et al. 1997; Biham and Shamir 1997; Piret and Quisquater 2003; Tunstall et al. 2011] and fault sensitivity analysis [Li et al. 2010; Moradi et al. 2011; Li et al. 2012] cannot be applied in the attack scenario of BFA.

All the previous fault attacks rely on the values of the public data in the key recovery. The assumption of BFA fits many practical applications of a block cipher. For example,

This work is supported by China Postdoctoral Science Foundation (No.2015M581795), Jiangsu Province Postdoctoral Science Foundation (No.1501014A) and Research start up fund of NUAA under grant 90YAH15029.
Author's addresses: Yang Li (corresponding author), Zhe Liu and Jian Wang, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Jiangjundadao No. 29, Nanjing, Jiangsu, China, 211106. Email: li.yang@nuaa.edu.cn.

DOI: 0000001.0000001

the encryption system when whitening is applied to both the input and the output. In addition, when block ciphers are used in an authentication system, the attackers may have difficulty in accessing the inputs and outputs. The original proposal of BFA successfully recovered the key of the several block ciphers, *i.e.* AES [National Institute of Standards and Technology 2001], LED [Guo et al. 2011] and SAFER++ [Massey et al. 2000], which are using the substitution permutation networks (SPN) structure.

The fault model used by BFA is the multi-bit reset or set fault injections, in which the fault injection can probabilistically reset or set the bit values of a specific byte or nibble. In BFA, it is assumed that the calculation outputs under fault injections can be compared with each other. Therefore, the attackers can know the total number of different outputs, which are caused by the fault injections when encrypting an unknown plaintext.

However, a successful BFA key recovery requires much more fault injections compared to other fault attacks. For example, it is estimated that 480,000 faults are required to recover a 128-bit key of AES. For DFA, it is well known that 1 or 2 fault injections are enough to identify the same AES key. This is of course due to the difficult attack setting of BFA. However, by analyzing the original BFA, we find that there is some room to reduce the required number of fault injections. In order to reveal the true threats of BFA, this work comprehensively studies the possibility of reducing the number of fault injections without changing any assumptions.

Contributions. First, this work analyzes the BFA key recovery via an information theoretical perspective. This is inspired by [Sakiyama et al. 2012], in which the DFA attack on AES is analyzed to achieve the optimal data complexity. For BFA, we evaluate how much information is gained via each time of fault injection, which is called information gained for a fault injection (IGFI). It is found that both the target intermediate value and the fault position in the injection sequence affect IGFI significantly. In fact, many fault injections in the original BFA have a low IGFI. If the BFA attacker can set up a strategy that mainly uses the fault injections with high IGFI, the number of fault injections for a successful key recovery can be reduced.

We apply the idea of using high IGFI to BFA in three approaches. Each of them corresponds to an attack scenario, and achieves a reduction to the number of fault injections compared to the original BFA.

This result is summarized in Table I. The original BFA uses 30,000 fault injections to achieve a 99% of key identification rate where only the correct key remains. The first attack, selective plaintext BFA, uses 47.4% of fault injections of the original BFA to reduce the key space to 1.6 candidates in average. The key identification rate of this attack is 61.1%. The second attack, fixed faults per plaintext BFA, introduces a probability-based distinguisher to BFA, which eliminates the possibility that the correct key being deleted in the key sifting phase. This attack also shows that BFA

Table I. Number of fault injections for recovery of an AES key byte

Attacks	References	Number of Faults	KIR[1]	ERK[2]
Original BFA	[Korkikian et al. 2014]	30,000 (100%)	99%	1
Selective Plaintext BFA	Section 4.1	14,219 (47.4%)	61.1%	1.6
Fixed Faults Per-Plaintext BFA	Section 4.2	17,748 (59.2%)	99%	1
Dynamic Fault Injection BFA	Section 4.3	9,978 (33.3%)	99%	1

[1] KIR stands for key identification rate.
[2] ERK stands for expected remaining keys.

key recovery is possible and efficient when the number of fault injections on every plaintext is limited. The probability distinguisher BFA achieves 99% key identification rate using 59.2% of fault injections. The third attack, dynamic fault injection BFA, requires the attacker dynamically evaluates the expected information gain for each fault injection and decides whether or not to apply it. This dynamic fault injection strategy results in the best performance, which uses 33.3% of fault injections to achieve the same attack result of the original BFA.

Our work shows that BFA can be applied more efficiently with regard to the number of fault injections. One third of the fault injections are already enough to achieve the same key recovery results. This work also shows that even when the attackers cannot apply many fault injections to the same plaintext, the BFA key recovery is still possible and efficient. Thus, this work accurately evaluates the key recovery requirement for BFA.

Organization. The rest of this paper is organized as follows. Section 2 briefly reviews the previous fault attacks and explains the original BFA. Section 3 explains several observations by evaluating BFA via an information theoretical perspective. In section 4, three improved BFA methods under different attack scenarios and their attack results on AES are explained. In section 5, the impact of this work on the feasibility of BFA is discussed. Section 6 concludes the paper and discusses the future work.

2. PREVIOUS FAULT ATTACKS AND ORIGINAL BFA

Fault attack uses the faulty calculations of the implemented cryptographic algorithms to help recovering the secret key, which was first proposed in 1997 [Boneh et al. 1997]. Different kinds of fault attacks have been proposed, which have become serious security threats to cryptographic algorithms implemented in embedded systems. In this section, we briefly review the previous fault attacks on block ciphers and explain the uniqueness of BFA.

2.1. Existing fault attacks on block ciphers

Differential fault analysis (DFA) is the most widely discussed fault attack applied on block ciphers. The basic concept of DFA was proposed in [Biham and Shamir 1997]. In DFA attack, the attackers inject certain computational faults into the running cryptographic devices. DFA attack usually requires the attacker to have certain control of the injected fault, which is usually called the fault model. With the fault model, the fault-free output and faulty output for the same input, the key recovery can be performed. The key recovery is similar to the differential cryptanalysis, which analyzes the differential equations to restrict the key space. With an appropriate fault model, the data complexity of DFA could be very small. Take AES as an example, a representative DFA recovers the full key using less than 2 fault injections [Piret and Quisquater 2003]. Later, researchers show that 1 fault injection is enough for the key recovery [Tunstall et al. 2011]. Note that the fault-free and faulty ciphertexts are necessary for DFA.

Another type of fault attack depends on the implementation details of the attack target. Recently, many new fault attacks belong to this type have been proposed. In 2010, fault sensitivity analysis (FSA) has been proposed [Li et al. 2010] and later be improved in [Moradi et al. 2011; Li et al. 2012]. In FSA, fault injections are used to measure how much the current calculation is sensitive to the fault injection intensity as the fault sensitivity. Measuring and analyzing fault sensitivity can obtain the information of processed intermediate value, which can be used to recover the secret key. For FSA, usually a few hundreds to a few thousands of fault injections are required

for a successful key recovery. The fault-free ciphertext is a must for a successful FSA attack.

After FSA, there are a series of new fault attacks, which use the biased (non-uniformity) of the injected faults [Lashermes et al. 2012a; Wang et al. 2013; Liu et al. 2015], the biased faulty values [Ghalaty et al. 2014] or the biased faulty output [Fuhr et al. 2013; Li et al. 2013; Santis et al. 2014] to perform key recovery. This type of attack usually requires more than a few hundreds of fault injections for a successful key recovery and must use the value of ciphertexts in the key recovery.

2.2. BFA on SPN-based block ciphers

All the above mentioned fault attacks cannot succeed if the attacker does not know the value of neither the plaintexts nor the ciphertexts. BFA was proposed under this special attack scenario. There are two special requirements for BFA compared to the previous fault attacks.

— BFA requires the multi-bit reset or set fault model, which is relatively difficult to realize in practice. These fault models have been applied previously in [Blömer and Seifert 2003] and [Lashermes et al. 2012b]. Also, successful multi-bit set or reset fault injections have been verified practically in [Moro et al. 2013; Roscian et al. 2013] based on laser shots at SRAM.
For a set fault, each internal "0" bit has a 50% chance to become "1", while "1" bit remains "1". Denote the fault free intermediate value as $D \in \mathbb{F}_2^n$, and faulty intermediate value as $D' \in \mathbb{F}_2^n$. We have $D' = D \vee e$, where e is a random number and $e \in \mathbb{F}_2^n$.
In contrast, for a reset fault, each "1" bit has a 50% chance to become "0", while "0" bit remains "0". For a reset fault, $D' = D \wedge e$, where e is a random number and $e \in \mathbb{F}_2^n$.
Without loss of any generality, the rest of paper only discusses the reset fault model.
In original BFA, the attacker is assumed to be able to inject multi-bit reset fault into an internal byte/nibble of the block cipher at desired timing and location. This work follows this assumption.
— The BFA attacker cannot access the value of either the plaintext or the ciphertext. However, the attackers can encrypt the same plaintext many times and the outputs for all the encryptions can be compared pairwise. By doing so, the attackers can understand how many different outputs are generated under the multi-bit reset fault injections.
To summarize, for each unknown plaintext, the BFA attackers can control the time of multi-bit reset fault injections at any desired byte/nibble and timing, and can understand the number of different outputs.

2.2.1. Basic BFA key recovery problem. The BFA is applied to block ciphers with SPN structure. The basic concept of SPN structure is illustrated in the left part of Figure 1. The internal state is repeatedly processed by a sequence of a substitution layer, a permutation layer and a key addition layer. The substitution layer uses carefully selected components such as S-box to achieve non-linear calculation. The permutation layer shuffles the bit positions or performs a linear transformation of the internal state. Finally, the internal state is mixed with a round key.

Same with all the other physical attacks, BFA performs the key recovery of the full key in a divide-and-conquer process. Each time, the attacker only targets the calculation related to 1 byte or 1 nibble of key. For a block cipher based on the SPN structure, a round key is divided into small pieces to be recovered one by one. Each small key piece is inside a basic structure as shown in the right part of Figure 1.

The BFA only focuses on the substitution part and the key addition part. The permutation part does not affect BFA. Note that the sequence of the key addition and the substitution does not affect the key recovery efficiency. The rest of this paper follows

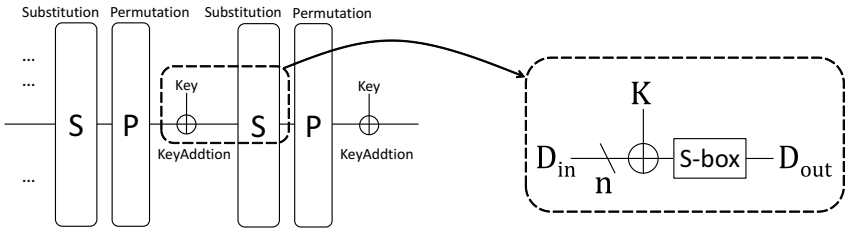

Fig. 1. Basic structure of BFA.

the basic structure as shown in Figure 1. The input data, the output data and the key inside the basic structure are denoted as D_{in}, D_{out} and K, respectively. The length of the path of the basic structure is n. For a modern block cipher, n is usually 8, *e.g.* AES, or 4 , *e.g.* LED, in order to achieve a reasonable balance of the implementation cost and the cryptanalysis resistance.

2.2.2. Original BFA key recovery. This section explains the key recovery for BFA described in [Korkikian et al. 2014], which can be divided into two steps as *Hamming weight recovery phase* and *key recovery phase*.

Hamming weight recovery phase. First, BFA attacker recovers the Hamming weights of D_{in} and D_{out} one by one to obtain a Hamming weights pair, *i.e.* $(W_{D_{in}}, W_{D_{out}})$. Denote target data as D, the Hamming weight and the bit length of D are denoted as W_D and n, respectively. Under a multi-bit reset fault injection, the total number of possible faulty outputs is $B = 2^{W_D}$. With enough fault injections, the attacker can obtain all the possible faulty outputs and know the number of it. Thus, B and W_D can be recovered.

The relations between the fault injections and faulty outputs can be considered as an "occupancy problem". The probability that after L fault injections, T out of B different possible ciphertexts are received can be considered as the probability that T out of B bins are occupied after throwing randomly L balls. This probability $T = t$, $t \in \{0, 1, \ldots, \min(L, B)\}$ can be calculated as follows:

$$\Pr(T = t) = \begin{cases} \frac{B!\alpha_{t,L}}{(B-t)!B^t}, & t \in \{1, 2, ..., \min(L, B)\}, \\ 0, & \text{else} \end{cases} \tag{1}$$

where $\alpha_{t,L}$ is the Stirling number of the second kind, *i.e.*

$$\alpha_{t,L} = \frac{1}{t!} \sum_{i=1}^{t} (-1)^{t-i} \binom{t}{L} i^t. \tag{2}$$

In BFA, the values of L and T are known by fault injections, and W_D must be determined. To estimate W_D, a maximum likelihood estimator $\hat{W_D}$ as a function of T and L is used as

$$\hat{W_D} = \arg\max_{W_i} \Pr(T|W_i, L). \tag{3}$$

With simulation-based fault injection experiments, achieving the 99% accuracy of recovering a Hamming weight requires 15 and 62 fault injection for $n = 4$ and $n = 8$, respectively.

Key recovery phase. With the several Hamming weight pairs for a certain K. BFA attackers try to identify K, which is divided into two steps. The first step is key sifting,

which eliminates the key candidates that fail to satisfy any obtained Hamming weight pair with any possible $D_{in} \in \mathbb{F}_2^n$.

The second step is key likelihood estimation. For each key candidate K, one can pre-compute the Hamming weight probability distribution (HWPD) as $\mathbf{Pr}\left(W_{D_{in}}, W_{S(D_{in} \oplus K)}\right)$ for uniformly distributed $D_{in} \in \mathbb{F}_2^n$ and an S-box S. It turns out the HWPD, $\mathbf{Pr}\left(W_{D_{in}}, W_{S(D_{in} \oplus K)}\right)$, depends on the value of K. The attackers have an observed distribution of $\mathbf{Pr}_r(W_{D_{in}}, W_{D_{out}})$ with the Hamming weight pairs obtained in the Hamming weight recovery phase. By comparing the obtained HWPD and the key-dependent pre-computed real HWPD, the attackers can identify which key candidate likes the real key the most. In [Korkikian et al. 2014], the Euclidean distance between two probability distributions is used as the distinguisher to identify the likelihood. The Euclidean distance between $\mathbf{Pr}_r$ and $\mathbf{Pr}_k$ is computed for each key as

$$D(\mathbf{Pr}_r, \mathbf{Pr}_k) = \sqrt{\sum_{\forall w_i, w_j} (\mathbf{Pr}_r(W_i, W_j) - \mathbf{Pr}_k(W_i, W_j))^2}. \tag{4}$$

The key candidate with the minimum Euclidean distance is considered as the correct one:

$$\hat{k} = \arg\min_k D(\mathbf{Pr}_r, \mathbf{Pr}_k). \tag{5}$$

The required amount of Hamming weight pairs for the successful key recovery depends on the S-box lookup table. Based on attack simulations, it is found that the 99% key recovery success rate requires 50 plaintexts for LED and 250 plaintexts for AES. The number of fault injections for the recovery of a byte/nibble key is shown in Table II. For AES, each key byte requires 30,000 fault injections.

Table II. Number of fault injections to recover a byte/nibble key in original BFA attack

Cipher	Number of plaintexts	Number of faults per plaintext	Total number of faults
AES	250	120	30,000
LED	50	40	2,000

For AES-128, the original BFA requires 480,000 fault injections to recover the 128-bit full key. Obviously, the number of fault injections is the main limitation of the feasibility of BFA. The challenge is whether or not the number of fault injections can be reduced for BFA key recovery. Then the security threat of BFA can be more accurately evaluated.

3. REVIEW BFA VIA INFORMATION THEORETICAL PERSPECTIVE

This section reviews BFA key recovery via an information theoretical perspective in order to find the possibility of reducing the number of fault injections. In this perspective, each fault injection is considered as a method to obtain the information of the intermediate values, which is linked to the recovery of the secret. By analyzing the information gained from each fault injection, the minimum number of fault injections for a key recovery can be evaluated accurately. This approach has been applied to evaluate the DFA attack on AES in [Sakiyama et al. 2012] and DFA attack on CLEFIA in [Krämer et al. 2014]. Both of them achieved a more accurate evaluation. Without loss of generality, the following discussion focuses on AES where $n = 8$.

As mentioned in Section 3, the original BFA first recovers the Hamming weight of intermediate values, then recovers the secret key. Instinctively, we can have the following observations for the original BFA.

— The recovered Hamming weights contribute differently to the key recovery considering the remaining entropy of the target value given the Hamming weights. For example, the recovery of $W_D = 0$ corresponds to the recovery of the target value, whose remaining entropy is 0 bit. On the other hand, with the recovery of $W_D = 4$, the remaining entropy of the target value still has $-\log_2 1/70 = 6.129$ bits.
— The higher the Hamming weight of the target intermediate value is, the more fault injections are required to recover the Hamming weight. When $W_D = 0$, since every reset fault does not lead to a different output from the fault-free one, the attackers can easily identify $W_D = 0$ with only a few fault injections. However, in the case of $W_D = 8$, at least $2^7 = 128$ fault injections are required to accurately distinguish whether $W_D = 7$ or $W_D = 8$.
— For a given W_D and a sequence of fault injections in the recovery process of W_D, the information gain for each fault injection is different from each other. Instinctively, the first ten or so fault injections are the most valuable ones for the key recovery. Note that the number of different outputs is the only clue for the attackers to recover W_D. Along the sequence of fault injections, the probability to obtain a new output is generally decreasing. Thus, the expected information gain for the fault injection is decreasing as well.

By summarizing these observations, it is found that in the recovery of information of D, the information gain of fault injections has a large difference. Mainly, it is related to the Hamming weights of D and the position of the fault in the injection sequence. In the process of BFA, the recovery of secret information is the result of accumulating the information recovery of intermediate values. Each fault injection is contributing to the entropy reduction of the secret key. The total entropy of the secret key is fixed, in order to reduce the number of fault injections, the BFA attacker should take advantage of the fault injections that has higher information gains.

All the probabilities and entropy transformation in the key recovery process mentioned in the observations can be calculated. Thus, the attackers can take advantage of them to understand the expected information gain of each fault injection that is to be performed. By setting up a fault injection strategy, the attacker makes sure that each performed fault injection is expected to have a high information gain for the key recovery. Then, the required number of fault injections is reduced and optimized,

3.1. Evaluation of Information Gain of Fault Injections

Increasing the expected information gain of each performed fault injection is the key idea of this work. Hereafter, the expected information gain for each fault injection is evaluated for AES.

Given $n = 8$ and the total number of fault injections $L = l$, $l \in \{1, 2, \ldots\}$, a tuple of Hamming weight of D and total number of different outputs as (W_D, T), the probability $\Pr(W_D = w_D, T = t)$ for $w_D \in \{0, 1, \ldots, 8\}$ and $t \in \{1, 2, \ldots, l+1\}$ can be computed. They can form a joint probability distribution $\mathbf{Pr}(W_D, T)_{L=l}$.

Our first attempt is to follow equations (1) and (2) to mathematically compute $\mathbf{Pr}(W_D, T)_{L=l}$. However, we find the probability calculation described in equations (1) and (2) is not accurate. They omit the fact that the attackers can always obtain a fault-free output when no fault injection action is applied, which should be counted in a calculation output. For example, with only 1 time of fault injection. *i.e.* $L = 1$, there is a possibility to obtain 2 different outputs as the fault-free output and a faulty one caused by the fault injection. Thus, when $L = 1$, we have $t \in \{1, 2\}$.

When L is small enough, it is relatively easy to calculate the probability distribution $\mathbf{Pr}(W_D, T)_L$. For example, in the case of $L = 1$, the probability distribution of $\mathbf{Pr}(W_D, T)_{L=1}$ is shown in Table III. For example, when $W_D = 2$, the probability of

obtaining a fault-free output for a fault injection is $(1/2)^{W_D} = 1/4$. Thus, we have $\Pr(T = 1|W_D = 2) = 1/4$ and $\Pr(T = 2|W_D = 2) = 3/4$, when $L = 1$. By multiplying them with $\Pr(W_D = 2) = 28/256$, one have $\Pr(T = 1, W_D = 2) = 7/256$ and $\Pr(T = 2, W_D = 2) = 21/256$, which is shown in the third column of Table III. One can calculate all the joint probability for $W_D \in \{0, 1, \ldots, 8\}$ and $T \in \{1, 2, \ldots, l + 1\}$. Note that $\mathbf{Pr}(W_D, T)_L$ can be converted to $\mathbf{Pr}(D, T)_L$ since D is uniformly distributed for every W_D.

Table III. Probability distribution of $\mathbf{Pr}(w_D, t)$ when $l = 1$

		w_D								
		0	1	2	3	4	5	6	7	8
t	1	$\frac{1}{256}$	$\frac{4}{256}$	$\frac{7}{256}$	$\frac{7}{256}$	$\frac{35}{2048}$	$\frac{7}{1024}$	$\frac{7}{4096}$	$\frac{1}{4096}$	$\frac{1}{65536}$
	2	0	$\frac{4}{256}$	$\frac{21}{256}$	$\frac{49}{256}$	$\frac{525}{2048}$	$\frac{385}{1024}$	$\frac{441}{4096}$	$\frac{127}{4096}$	$\frac{255}{65536}$

Using the joint probability table as shown in Table III, one can evaluate how much information of W_D and information of D are obtained by this fault injection. Before this fault injection, the entropy of the W_D, *i.e.* $H(W_D)$, can be calculated as

$$H(W_D) = -\sum_{i=0}^{8} \frac{\binom{8}{i}}{256} \log_2 \left(\frac{\binom{8}{i}}{256} \right) = 2.544 \text{ bits}. \tag{6}$$

After this 1 time of fault injection, the conditional entropy of W_D after knowing T is

$$H(W_D|T)_{L=1} = -\sum_{t,w_D} \Pr(W_D = w_D, T = t) \log_2(\Pr(W_D = w_D|T = t)) \tag{7}$$

$$= 2.465 \text{ bits}, \tag{8}$$

where $t \in \{1, 2\}$ and $w_D \in \{0, 1, \ldots, 8\}$. Therefore, the average information gain of W_D for this fault injection is $2.544 - 2.465 = 0.079$ bits. One can calculate the information gain of D for this fault injection by $H(D) - H(D|T)$ as 0.079 bits as well. Following the same approach, using the joint probability distribution $\mathbf{Pr}(W_D, T)_{L=l}$ where $l \in \{1, 2, \ldots\}$, one can use $H(D|T)_{L=l} - H(D|T)_{L=l-1}$ to obtain the expected information gain of D for the l-th time fault injection.

In our evaluation, it is not easy to mathematically compute $\mathbf{Pr}(W_D, T)_{L=l}$ for all possible l. Since the purpose of this evaluation is to have a sense of the fault injection efficiency, we decided to use a Monte Carlo approach to replace all the probability calculations. In the Monte Carlo approach, random numbers are generated to perform the fault injection simulations. After repeating the simulated fault injections for 10^7 times, the occurrences of all the events are counted to obtain the estimation of the corresponding probabilities. By the Monte Carlo approach, we obtained all the joint probability distribution of $\mathbf{Pr}(W_D, T)_{L=l}$ for $l = \{1, 2, \ldots, 100\}$ with reasonably small noise.

Finally, we can evaluate the information gain of D for each fault injection in BFA. The evaluation results are shown in Figures 2 and 3. Figure 2 shows the remaining entropy of D for the first 100 fault injections. Figure 3 shows the information gain of D for the first 100 fault injections.

As shown in Figure 2, the remaining entropy of D decreases dramatically for the first 20 fault injections. Then the decreasing speed decreases. After about 60 fault injections, the remaining entropy does not decrease any more and close to the limitation, which fits the result shown in the original BFA. The meaning of the limitation around 60 fault injections is that W_D is almost fully recovered but no more information can be

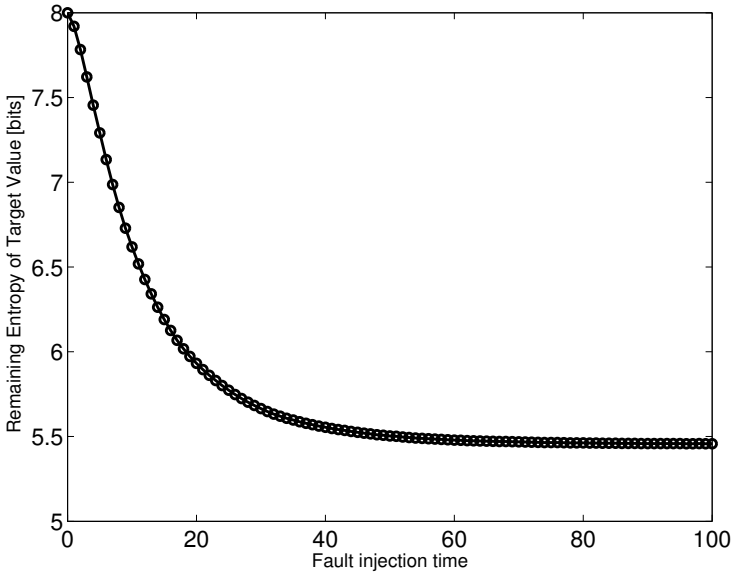

Fig. 2. Expected remaining entropy of D for first 100 fault injections.

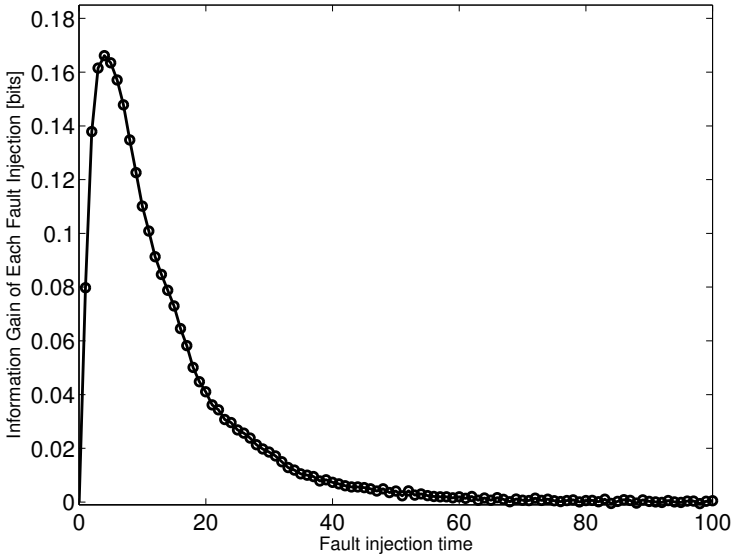

Fig. 3. Information gain of D for 100 fault injections.

recovered for D. In Figure 3, one can see that the 4-th fault injection has the peak information gain. After 60 fault injections, the information gain is much less. The large difference of information gain between fault injections implies a large improvement room of the original BFA.

3.1.1. Information gain for different Hamming weights. As mentioned in our observations, the Hamming weight of target value W_D affects the information gain a lot. Assume $W_D = 0$, the attackers will observe T in the probability distribution corresponding to

$W_D = 0$, which is the first column of Table III. The attacker tries to recover the information of D with the observed distribution of T. Table III can be used to calculate the expected distribution of W_D given an observed distribution of T. Then the expected distribution of W_D can be converted to a distribution of D, which is used to compute the remaining entropy of D. Following these calculations, one can understand the influence of W_D for the information gain of each fault injection.

We compute the results based on the probability distribution obtained with a Monte Carlo approach. Figures 4 and 5 show the evolution of the expected entropy of D and the information gain of D against the first 100 fault injections when W_D is fixed. Note that the attackers don't have any information of the fixed W_D, the attackers only try to recover the information of D based on the observation of T, while the distribution of T is determined with a fixed W_D.

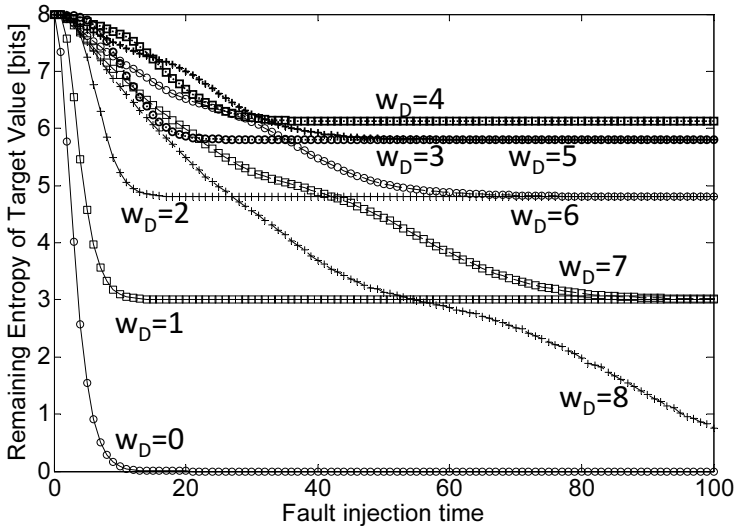

Fig. 4. Expected remaining entropy of D along fault injections for different W_D.

As shown in Figures 4 and 5, we can see the result is fitting the expectations. The first a few fault injections are most efficient for $D = 0$, which can be as high as 1.8 bits for the information gain for recovering D. After repeating a few fault injections, the entropy of D can be reduced to 0 since D can be identified. In the case of $w_D = 4$, the information gain of each fault injection is always below 0.2 bits, and the entropy of D cannot be reduced to below $\log_2(1/70) \approx 6.13$ bits, The attackers can take advantage of the above result to perform fault injections with high information gain as much as possible.

3.1.2. Proposal of Probability-Based Distinguisher for BFA. The original BFA has a possibility of eliminating the correct key in its key sifting step. In the first step of the original BFA, the attackers want to identify the Hamming weight of intermediate value. However, even after 62 fault injections, there is still a possibility that the identified Hamming weight is different from the real one. The problem is that if this error Hamming weight is used in the key sifting step, there is a possibility that the correct key byte is eliminated from the key space. Then it results in a failed key recovery. Considering that this 99% success rate of identifying correct Hamming weight is repeated over 500 trails, the possibility of eliminating the correct key should not be ignored.

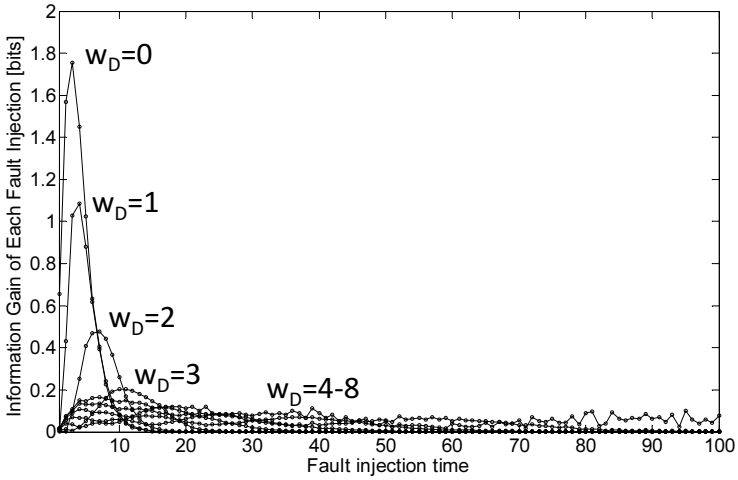

Fig. 5. Information gain of D along fault injections for different W_D.

Note that in [Korkikian et al. 2014], it is possible to omit the key shifting step in the key recovery. Then, the possibility of eliminating the correct key can be fixed. However, the key recovery efficiency is further reduced since more fault injections is required to identify the correct key.

Our work proposes a new approach to connect two attack phases for BFA, which has two main differences from the previous one.

— First, in the phase of Hamming weight recovery, instead of identifying the Hamming weight, the attack result is a probability distribution of the Hamming weights. The probability distributions for $W_{D_{in}}$ and $W_{D_{out}}$ can be combined to form a probability distribution of the Hamming weights pair, *i.e.*, $\mathbf{Pr}(W_{D_{in}}, W_{D_{out}})$. The idea is to use probability distribution to replace the necessity of identification, then all the information in the Hamming weight recovery phase is used in the key recovery phase. The key recovery efficiency and flexibility is increased.
— Second, in the key recovery phase, instead of using the Euclidean distance as the distinguisher, we apply a new distinguisher as shown in Eq. (9).

$$\hat{k} = \arg\max_{k} \prod_{i} \Pr(k|\ \Pr(W_{D_{in}}, W_{D_{out}})_i)). \tag{9}$$

The idea is to calculate the probability of each key candidate using the probability distribution of the Hamming weights pair and HWPD for each key candidate. After the calculation, the key candidate with the highest probability is considered as the correct key. In this distinguisher, the key sifting and the accumulation of key likelihood are combined. A key with a 0 probability will be eliminated automatically in the multiplication of the probability distinguisher. The probability distribution of the Hamming weights pair as $\mathbf{Pr}(W_{D_{in}}, W_{D_{out}})$ can directly connect two attack phases.

As long as enough data is used in the key recovery, the correct key can be identified just like other side-channel attacks such as differential power analysis. The disadvantage is that in our approach the attackers need more computational abilities to perform the probability distribution calculations. In the new key recovery approach, there is no need to identify the correct Hamming weight. There is also no worry about eliminating the correct key since all the keys are evaluated in the key recovery phase using probability calculations.

4. IMPROVED BFA ATTACK WITH REDUCED FAULT INJECTION TIMES

Based on analyzing BFA from an information theoretical perspective and the newly proposed distinguisher for BFA, this section discusses several methods to reduce the number of fault injections. Specifically, we describe three BFA methods.Each BFA method is optimized under a certain attack scenario, which is related to the size of plaintext space, fault injection times per plaintext and the calculation ability of the attackers.

4.1. Selective Plaintexts BFA

The first attack scenario is the case where the attackers can freely choose plaintext, and there is no limitation of the fault injection times for each plaintext. In this BFA method, we follow the original fault attack to identify the Hamming weight in the first attack phase, without using the probability-based distinguisher. The point of showing this attack is to illustrate that instead of the new distinguisher, the selection of fault injections with high information gain is the main contribution of this work.

As mentioned in Section 4, the Hamming weight of the target value largely affects the information gain per fault injection. The attackers can use a few fault injections to test the plaintexts first. For the basic BFA structure shown in Figure 1, a few fault injections are applied at D_{in} first. With the fault injection results, the attackers already be able to identify whether the Hamming weight is extremely low. A low Hamming weight implies that the performed fault injections are information theoretically efficient. For each plaintext that has efficient fault injections at D_{in}, we keep on performing fault injections at D_{out} to identify its Hamming weights to obtain the Hamming weight pair. We can repeat the same process by focusing on obtaining the efficient plaintext with regard to D_{out}. After the fault injections, we obtain the pair of Hamming weights of D_{in} and D_{out}, *i.e.* $(W_{D_{in}}, W_{D_{out}})$. The fault injection strategy is described in Algorithm 1.

In the selective plaintexts BFA, the attackers first use a few fault injections to select the plaintext that has a high information gain per fault injection. Then only those plaintexts are used in the further analysis for the key recovery. All the inefficient plaintexts are abandoned.

Note that we consider $W_D < 2$ as the efficient plaintexts in Algorithm 1. In Algorithm 1, Λ_r^0 is the set of the Hamming weight pairs with $W_{D_{in}} = 0$, which is obtained from the fault injection results. Similarly, Λ_r^1, Λ_r^2, Λ_r^3 are the sets of the Hamming weight pairs with $W_{D_{in}} = 1$, $W_{D_{out}} = 0$ and $W_{D_{out}} = 1$, respectively. The number of total fault injections σ can be estimated as follows

$$\sigma = Nl_1 + Nl_2\big(\Pr(W_{D_{in}} \in \{0,1\}) + \Pr(W_{D_{out}} \in \{0,1\})\big), \tag{10}$$

$$= \frac{128l_1 + 9l_2}{128}N, \tag{11}$$

where N is the number of used plaintexts, l_1 is the number of fault injections to find good efficiency plaintext, l_2 is the number of fault injections required to recover a Hamming weight.

In the key identification phase, for each key guess, the perfect set of the Hamming weight pairs are calculated for all the efficient plaintexts. Then as long as the fault injection result is a subset of the corresponding perfect set of the Hamming weight pair, we consider the current key guess as a key candidate. The key recovery algorithm is illustrated in Algorithm 2. Note that the key distinguisher in Algorithm 2 directly matches the Hamming weight pairs between the experiment result and the expected result for every possible key guess. Unlike the one used in [Korkikian et al. 2014], this distinguisher does not rely on the distribution of Hamming weight pairs.

ALGORITHM 1: Data Collection of Improved BFA with Selective Plaintexts

Input: N random plaintexts: $P_0, P_2, \ldots, P_{N-1}$.
Output: Hamming weight pair set $\Lambda_r^0 = \{(0, W_{D_{out}})\}$, $\Lambda_r^1 = \{(1, W_{D_{out}})\}$, $\Lambda_r^2 = \{(W_{D_{out}}, 0)\}$ and $\Lambda_r^3 = \{(W_{D_{out}}, 1)\}$.
for $i = 0$ *to* $N - 1$ **do**
  Fix P_i as plaintext;
  Inject fault at D_{in} for l_1 times ;
  if $t = 0$ **then**
    Inject faults at D_{out} for l_2 times to get $\Lambda_r^0 = \{(0, W_{D_{out}})\}$;
  end
  if $t = 1$ **then**
    Inject faults at D_{out} for l_2 times to get $\Lambda_r^1 = \{(1, W_{D_{out}})\}$;
  end
  Inject fault at D_{out} for l_1 times ;
  if $t = 0$ **then**
    Inject faults at D_{in} for l_2 times to get $\Lambda_r^2 = \{(W_{D_{in}}, 0)\}$;
  end
  if $t = 1$ **then**
    Inject faults at D_{in} for l_2 times to get $\Lambda_r^3 = \{(W_{D_{in}}, 1)\}$;
  end
end

ALGORITHM 2: Key Recovery of Improved BFA with Selective Plaintexts

Input: $\Lambda_r = \{(W_{D_{in}}, W_{D_{out}}) | W_{D_{in}} \in \{0,1\}\} \cup \{(W_{D_{in}}, W_{D_{out}}) | W_{D_{out}} \in \{0,1\}\}$, key space $\kappa = \emptyset$.
Output: Key byte space κ.
for $k_g \in \kappa$ **do**
  $\Lambda_{k_g}^0 = \{(W_{D_{in}}, W_{D_{out}}) | W_{D_{in}} = 0, K = k_g\}$;
  $\Lambda_{k_g}^1 = \{(W_{D_{in}}, W_{D_{out}}) | W_{D_{in}} = 1, K = k_g\}$;
  $\Lambda_{k_g}^2 = \{(W_{D_{in}}, W_{D_{out}}) | W_{D_{out}} = 0, K = k_g\}$;
  $\Lambda_{k_g}^3 = \{(W_{D_{in}}, W_{D_{out}}) | W_{D_{out}} = 1, K = k_g\}$;
  if $\Lambda_r^0 \subset \Lambda_{k_g}^0$ and $\Lambda_r^1 \subset \Lambda_{k_g}^1$ and $\Lambda_r^2 \subset \Lambda_{k_g}^2$ and $\Lambda_r^3 \subset \Lambda_{k_g}^3$ **then**
    $\kappa = k_g \cap \kappa$;
  end
end

We evaluate the appropriate value of l_1 so that the error rate of identifying a low Hamming weight is reasonably low. The calculation is similar to the one discussed in Section 3. The probability of obtaining an error Hamming weight against the l_1 fault injections is shown in Figure 6. One can see that only 10 times of fault injections are good enough to identify them.

The selection of N relates to the total number of fault injections. When N is too small, the random plaintexts cannot cover all the available information theoretically efficient plaintexts. Generally, N should be big enough so that all D_{in} and D_{out} that has a Hamming weight less than 2 can be covered. We performed the attack simulation against the used number of plaintext N, whose result is shown in Figure 7 where $l_1 = 10$.

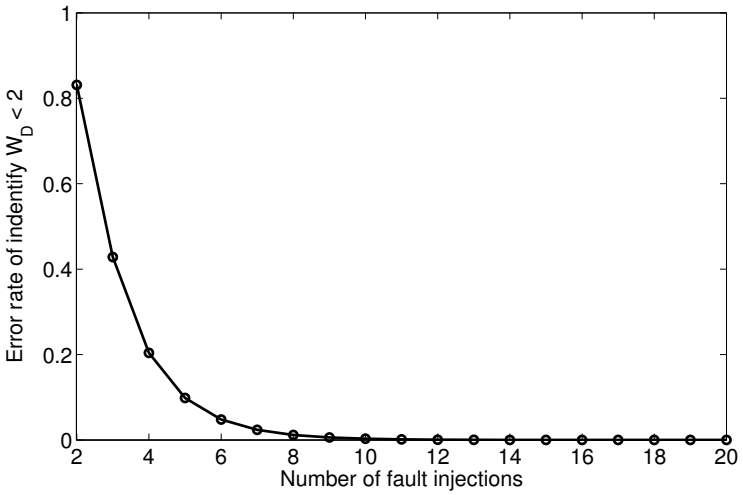

Fig. 6. Error rate of identify $W_D < 2$.

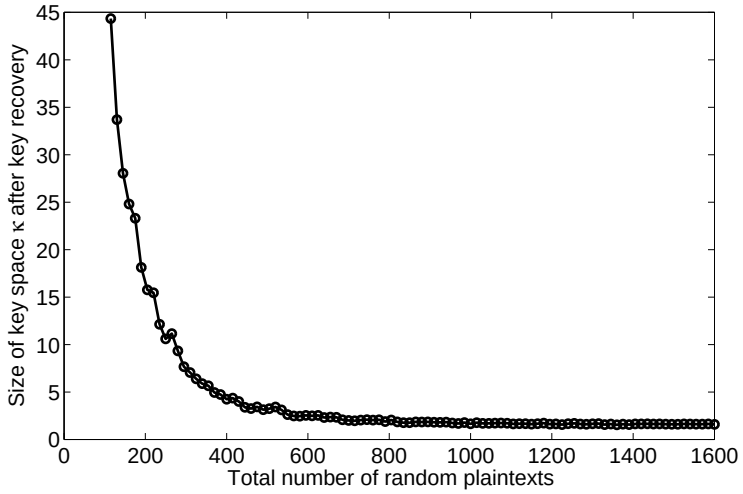

Fig. 7. Key space size after key recovery against the number of used plaintexts N.

From Figure 7, one can see that when $N = 1000$, the key space is reduced to about 1.6. The simulation result shows that this attack cannot identify K in all the cases. We further analyzed that the key identification rate in this case is 61.1%. Even with more fault injections, the key identification rate cannot extend 67% for this attack method. However, the number of fault injections for a byte key only requires $(10 + 60 \times 9/128) \times 1000 = 14219$ fault injections, which is a 52.6% reduction compared to 30,000 in the original BFA. As a summary, using the exactly same assumptions, about a half of the fault injections are enough to almost identify the key bytes by selecting plaintexts with higher information gain. Since the probability distinguisher is not used, the selective plaintexts BFA still has a possibility to identify wrong Hamming weight in the first attack phase.

ALGORITHM 3: Data Collection of Improved BFA with Limited Faults Per Plaintext

Input: N random plaintexts: $P_0, P_2, \ldots, P_{N-1}$, fixed fault injection time l_1.
Output: Hamming weight pair probability distribution $\mathbf{Pr}(W_{D_{in}}, W_{D_{out}})_i$ where $i \in \{0, 1, \ldots, N\}$.
for $i = 0 \rightarrow N - 1$ **do**
  Fix P_i as plaintext;
  Inject fault at D_{in} for l times ;
  Count number of different outputs as t_{in} ;
  Use (l, t_{in}) to obtain probability distribution, $\mathbf{Pr}(w_{D_{in})}$;
  Inject fault at D_{out} for l times ;
  Count number of different outputs as t_{out} ;
  Use (l, t_{out}) to obtain probability distribution, $\mathbf{Pr}(W_{D_{out}})$;
  Combine $\mathbf{Pr}(W_{D_{in}})$ and $\mathbf{Pr}(W_{D_{out}})$ for $\mathbf{Pr}_i(W_{D_{in}}, W_{D_{out}})$;
end

4.2. Fixed Faults Per Plaintext BFA

The second attack scenario assumes that there is a number limitation of encrypting the same plaintext. It becomes suspicious when a plaintext is encrypted for many times. In this attack scenario, the number of fault injections for each plaintext is limited.

When the faults per plaintext are limited to below 120, there are no enough fault injections to identify $W_{D_{in}}$ and $W_{D_{out}}$ accurately. In this attack scenario, the proposed probability-based distinguisher shows its advantages. Without identifying the Hamming weights, the fault injections are used to obtain a probability distribution of $W_{D_{in}}$ and $W_{D_{out}}$. In the key recovery phase, the Hamming weight probability distributions are directly used in the distinguisher.

Assume that for each plaintext only $2l$ times fault injections are allowed, the attacker applies l times fault injections to each of $W_{D_{in}}$ and $W_{D_{out}}$. Simply, the attackers count the number of different outputs as t after l fault injections. Then (l, t) provides the attackers a Hamming weight probability distribution of the target value $\mathbf{Pr}(W_D)$. Finally, the Hamming weight probability distributions for D_{in} and D_{out}, *i.e.* $\mathbf{Pr}(W_{D_{in}})$ and $\mathbf{Pr}(W_{D_{out}})$, are combined to obtain $\mathbf{Pr}_i(W_{D_{in}}, W_{D_{out}})$ for the i-th plaintext. We summarize the data collection for BFA with limited faults per plaintext in Algorithm 3.

The key recovery for BFA with limited faults per plaintext is similar to side channel attack, such as correlation power analysis [Brier et al. 2004]. Each key candidates k_g is given a credibility, $\gamma(k_g)$, as its likelihood to be the real key. For each probability distribution of the Hamming weight pair, we update the credibility for each key candidate using HWPD. After the data for all plaintexts are used, the key candidate with the highest credibility is the recovered key. In many cases, the correct key can be identified much faster than using the data for all the plaintexts. The key recovery for BFA with limited faults per plaintext is shown in Algorithm 4.

We evaluated the value of fixed fault injection time l ranges in $\{1, 2, \ldots, 100\}$. It is found that the best attack efficiency with regard to the fault injection time appears when $l = 18$, in which 36 fault injections are applied to each plaintext. The attack result for $l = 18$ is shown in Figure 8. In order to illustrate the improvement, we also plot the key recovery success rate for $l = 60$. The 99% success rate of the key identification takes 17,748 fault injections for $l = 18$, and 25,320 faults injections for $l = 60$. Using the proposed distinguisher $l = 18$ has a 30% reduction of the fault injection times compared to $l = 60$. Compared to the original BFA that uses 30,000 fault injections, the reduction is 40.8%.

ALGORITHM 4: Key Recovery of Improved BFA with Limited Faults Per Plaintext

Input: $\mathbf{Pr}_i(W_{D_{in}}, W_{D_{out}})$, $i \in \{0, 1, \ldots, N-1\}$, original key space κ.
Output: Recovered key byte $\hat{k}$.
for $k_g \in \kappa$ **do**
  $\gamma(k_g) = 1$;
  for $i = 0 \rightarrow N-1$ **do**
    $\gamma(k_g) = \gamma(k_g)\mathbf{Pr}(k_g|(W_{D_{in}}, W_{D_{out}}))\mathbf{Pr}_i(W_{D_{in}}, W_{D_{out}})$;
  end
end
$\hat{k} = \arg\max_k \gamma(k_g)$

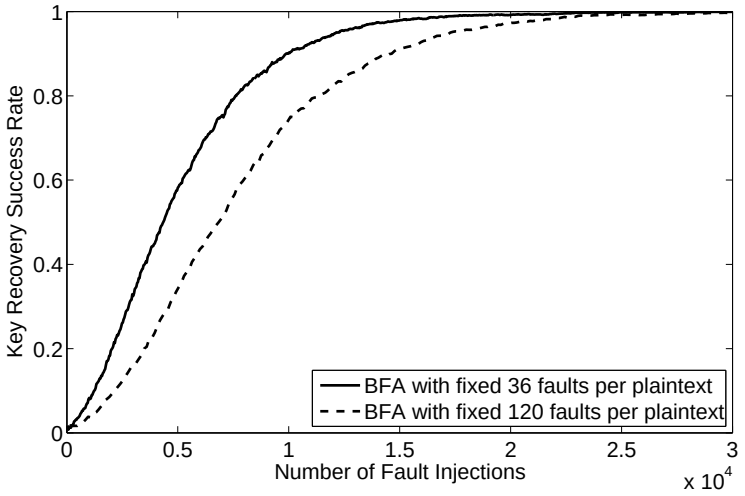

Fig. 8. Key recovery success rate with limited faults per plaintext.

The proposal of probability-distinguisher BFA with limited fault injections per plaintexts does not intend to reduce the number of fault injections. However, due to the fact that the first 20 fault injections have much larger information gain than the others, the attack efficiency is largely improved from the original BFA.

4.3. Dynamic Fault Injection BFA

The third attack is about a dynamic system applied in the fault injections. In this system, the attackers always calculate the expected information gain for a fault injection. The expected information gain is compared with a pre-set threshold. Only when the expected information gain is higher than the threshold, the fault injection will be actually performed. Otherwise, no more fault injections will be applied to this target intermediate value.

We denote the information gain threshold as ϵ_{th} and the expected information gain for the next fault injection as $\epsilon_{l,t}$. Obviously, $\epsilon_{l,t}$ only depends on l and t, which means l times of fault injections lead to t different outputs. In Section 3.1, we obtained the joint probability distribution $\mathbf{Pr}(W_D, T)_{L=l}$ using a Monte Carlo approach. As used in Algorithm 3, using l, t together with $\mathbf{Pr}(W_D, T)_{L=l}$, one can calculate the expected distribution of W_D based on Bayes' theorem from the probability theory. As shown in Figure 5, the expected information gain for different Hamming weights can be calculated. Thus, using the expected distribution of W_D, $\epsilon_{l,t}$ can be calculated as a weighted

summation of the information gain for different Hamming weights when $T = t + 1$. Until $\epsilon_{l,t} < \epsilon_{th}$, the attackers always perform fault injection to the target value and update the values of l and t. When $\epsilon_{l,t} \leq \epsilon_{th}$, no more fault injection is applied to this target value. The attack algorithm is illustrated in Algorithm 5.

ALGORITHM 5: Data Collection of Improved Blind Fault Analysis with Dynamic Faults.

Input: N random plaintexts, $P_0, P_2, \ldots, P_{N-1}$, Information gain threshold, ϵ_{th}.
Output: Probability distribution $\mathbf{Pr}_i(W_{D_{in}}, W_{D_{out}})$, $i \in \{0, 1, \ldots, N\}$.

```
for i = 0 to N − 1 do
    Fix P_i as plaintext;
    l = 0; t = 1;
    while ε_{l,t} > ε_th do
        Inject fault at D_in;
        l = l + 1;
        Update t ;
    end
    Use (l, t) to compute Hamming weight probability distribution, Pr(W_{D_in}) ;
    l = 0; t = 1;
    while ε_{l,t} > ε_th do
        Inject fault at D_out;
        l = l + 1 ;
        Update t ;
    end
    Use (l, t) to compute Hamming weight probability distribution, Pr(W_{D_out}) ;
    Combine Pr(W_{D_in}) and Pr(W_{D_out}) for Pr(W_{D_in}, W_{D_out})_i;
end
```

The key recovery phase is the same with BFA with limited faults per plaintext using the probability-based distinguisher, which is illustrated in Algorithm 4. This BFA with dynamic fault injection strategy requires a real-time feedback system in the fault injection phase. Furthermore, in order to achieve the least fault injection time, it is better to process the key recovery in real-time of the fault injections. These conditions make the practical attack more difficult. However, it is expected to achieve the minimum number of fault injections for a success key recovery.

The choice of the threshold ϵ_{th} is important to achieve the least number of fault injections. Intuitively, we want to set this threshold relatively high so that the average information gain for each fault injection is high. However, the problem is that when the threshold is too high, many plaintexts will be abandoned in the first ten or so fault injections. As shown in Section 3, the information gain curve is not a monotonous decreasing function. Furthermore, the curves of the information gain for different Hamming weights have totally different shapes. It is difficult to decide the best threshold instead of running simulation tests.

In order to evaluate the effects of the threshold choices, we perform attack simulations to evaluate the key identification rate for different threshold setting that ranges from 0.01 bits to 0.60 bits in the step of 0.01 bits. We found that when the threshold is set to 0.29 bit, the key recovery achieved the best efficiency with regard to the number of fault injections. The attack result for $\epsilon_{th} = 0.29$ is shown in Figure 9, which achieved the 99% key identification rate using only 9,979 fault injections. For comparison, the attack results for BFA with fixed faults per plaintext are plotted in Figure 9 as well.

Actually, the attack efficiency is stable when the threshold setting ranges from 0.01 to 0.40, in which less than 13,000 fault injections can achieve a 99% key identification

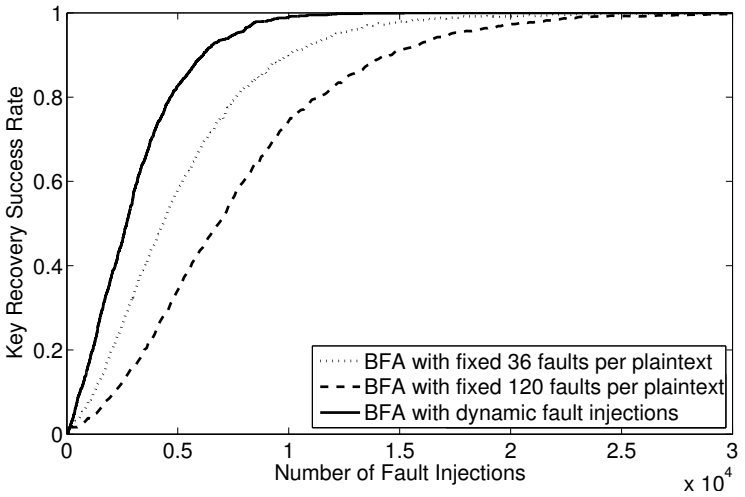

Fig. 9. Key recovery success rate with dynamic faults injections.

rate. The dynamic fault injection strategy achieved the least number of fault injections, which is less than one third of the original BFA.

5. FEASIBILITY OF BFA AFTER THIS WORK

Cryptographic engineers should pay more attentions to BFA due to its unique assumptions among all existing fault attacks. In specific systems, only BFA has a chance to recover the key. Furthermore, BFA is also the only fault attack that can directly recover any round key. The disadvantage of BFA includes the special fault model and a large amount of fault injections.

This work proved that only one third of the fault injections of the original BFA is enough to achieve the same key recovery result. Furthermore, this work showed that when the number of fault injections to the same plaintext is limited, the BFA key recovery is still possible and efficient. Thus, we believe the feasibility of BFA is largely improved under the study of this work. A more accurate evaluation of the BFA vulnerability is achieved.

Note that, along the reduction to the number of fault injections, the proposed attacks require more plaintexts and more computational ability from the attacker. As for the increased plaintexts, both original BFA and the proposed one requires a large amount of plaintexts. Since there are some cases the number of plaintexts are limited, a BFA strategy that minimizes the number of plaintexts is an interesting future work to explore. As for the increased computational complexity, it is totally tolerable in a practical attack. In this work, the most complex attack is the dynamic fault injection BFA. However, it is possible to pre-calculate all the information gain for all possible fault injection results in an off-line phase. Thus, in the online attack phase, table lookups can be used to replace the calculations. The newly proposed distinguisher does not change the divide-and-conquer concept as well, thus the increased complexity is not significant at all. The entire key recovery can finish with in 1 minute using a personal PC. We believe the trade-off for fewer fault injections is meaningful in the evaluation of BFA feasibility. When considering the practical threat of BFA attack, the most applicable target could be a micro-controller with an AES hardware module with a serial architecture, in which the multi-bit set or reset faults are more likely to be achieved for every S-box calculation.

As for the countermeasures, those countermeasures that fundamentally disable the information leakage, *e.g.* masking mentioned in [Korkikian et al. 2014], are still secure. The countermeasures that are based on the limitations of the total number of fault injections or the fault injections for each plaintext should reconsider their parameters based on this work.

6. CONCLUSION

This work improved the lately proposed blind fault attack (BFA) by reducing the number of required fault injections without changing the attack assumptions. First, this work analyzed the BFA fault injections from an information theoretical perspective, and discussed the strategies to take advantage of the fault injections with high information gain. Second, this work introduced a probability-based distinguisher to BFA to make the key recovery process error-tolerant. Finally, three improved BFA methods were proposed specifically for three slightly different attack scenarios. Improved BFA methods can reduce 40.8% to 66.8% of the fault injections when targeting AES. The future work includes exploiting the possible application of cryptanalysis techniques for BFA.

REFERENCES

Eli Biham and Adi Shamir. 1997. Differential Fault Analysis of Secret Key Cryptosystems. In *CRYPTO (Lecture Notes in Computer Science)*, Burton S. Kaliski Jr. (Ed.), Vol. 1294. Springer, 513–525.

Johannes Blömer and Jean-Pierre Seifert. 2003. Fault based cryptanalysis of the advanced encryption standard (AES). In *Financial Cryptography*. Springer, 162–181.

Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. 1997. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In *EUROCRYPT (Lecture Notes in Computer Science)*, Walter Fumy (Ed.), Vol. 1233. Springer, 37–51.

Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation Power Analysis with a Leakage Model. In *CHES (Lecture Notes in Computer Science)*, Marc Joye and Jean-Jacques Quisquater (Eds.), Vol. 3156. Springer, 16–29.

Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. 2013. Fault Attacks on AES with Faulty Ciphertexts Only. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*, Wieland Fischer and Jörn-Marc Schmidt (Eds.). IEEE Computer Society, 108–118. DOI:http://dx.doi.org/10.1109/FDTC.2013.18

Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont. 2014. Differential Fault Intensity Analysis. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*. IEEE, 49–58.

Jian Guo, Thomas Peyrin, Axel Poschmann, and Matt Robshaw. 2011. The LED block cipher. In *Cryptographic Hardware and Embedded Systems–CHES 2011*. Springer, 326–341.

Roman Korkikian, Sylvain Pelissier, and David Naccache. 2014. Blind Fault Attack against SPN Ciphers. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea, September 23, 2014*, Assia Tria and Dooho Choi (Eds.). IEEE Computer Society, 94–103. DOI:http://dx.doi.org/10.1109/FDTC.2014.19

Juliane Krämer, Anke Stüber, and Ágnes Kiss. 2014. On the Optimality of Differential Fault Analyses on CLEFIA. *IACR Cryptology ePrint Archive* 2014 (2014), 572. http://eprint.iacr.org/2014/572

Ronan Lashermes, Guillaume Reymond, Jean-Max Dutertre, Jacques J. A. Fournier, Bruno Robisson, and Assia Tria. 2012a. A DFA on AES Based on the Entropy of Error Distributions. In *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium, September 9, 2012*, Guido Bertoni and Benedikt Gierlichs (Eds.). IEEE Computer Society, 34–43. DOI:http://dx.doi.org/10.1109/FDTC.2012.18

Ronan Lashermes, Guillaume Reymond, Jean-Max Dutertre, Jacques Fournier, Bruno Robisson, and Assia Tria. 2012b. A DFA on AES Based on the Entropy of Error Distributions. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*. IEEE, 34–43.

Yang Li, Yu-ichi Hayashi, Arisa Matsubara, Naofumi Homma, Takafumi Aoki, Kazuo Ohta, and Kazuo Sakiyama. 2013. Yet Another Fault-Based Leakage in Non-uniform Faulty Ciphertexts. In *Foundations and Practice of Security - 6th International Symposium, FPS 2013, La Rochelle, France, October 21-22, 2013, Revised Selected Papers (Lecture Notes in Computer Science)*, Jean Luc Danger, Mourad Debbabi,

Jean-Yves Marion, Joaquín García-Alfaro, and A. Nur Zincir-Heywood (Eds.), Vol. 8352. Springer, 272–287. DOI:http://dx.doi.org/10.1007/978-3-319-05302-8_17

Yang Li, Kazuo Ohta, and Kazuo Sakiyama. 2012. New Fault-Based Side-Channel Attack Using Fault Sensitivity. *IEEE Transactions on Information Forensics and Security* 7, 1 (2012), 88–97.

Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. 2010. Fault Sensitivity Analysis. In *CHES (Lecture Notes in Computer Science)*, Stefan Mangard and François-Xavier Standaert (Eds.), Vol. 6225. Springer, 320–334.

Yannan Liu, Jie Zhang, Lingxiao Wei, Feng Yuan, and Qiang Xu. 2015. DERA: Yet another differential fault attack on cryptographic devices based on error rate analysis. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 1–6.

James L Massey, Gurgen H Khachatrian, and Melsik K Kuregian. 2000. Nomination of SAFER++ as candidate algorithm for the New European Schemes for Signatures, Integrity, and Encryption (NESSIE). *Primitive submitted to NESSIE by Cylink Corp* (2000), 5.

Amir Moradi, Oliver Mischke, Christof Paar, Yang Li, Kazuo Ohta, and Kazuo Sakiyama. 2011. On the Power of Fault Sensitivity Analysis and Collision Side-Channel Attacks in a Combined Setting. In *CHES (Lecture Notes in Computer Science)*, Bart Preneel and Tsuyoshi Takagi (Eds.), Vol. 6917. Springer, 292–311.

Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. 2013. Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*. IEEE, 77–88.

National Institute of Standards and Technology. 2001. FIPS 197: Advanced Encryption Standard. (November 2001).

Gilles Piret and Jean-Jacques Quisquater. 2003. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In *CHES (Lecture Notes in Computer Science)*, Colin D. Walter, Çetin Kaya Koç, and Christof Paar (Eds.), Vol. 2779. Springer, 77–88.

Cyril Roscian, Alexandre Sarafianos, J-M Dutertre, and Assia Tria. 2013. Fault model analysis of laser-induced faults in sram memory cells. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*. IEEE, 89–98.

Kazuo Sakiyama, Yang Li, Mitsugu Iwamoto, and Kazuo Ohta. 2012. Information-Theoretic Approach to Optimal Differential Fault Analysis. *IEEE Transactions on Information Forensics and Security* 7, 1 (2012), 109–120.

Fabrizio De Santis, Oscar M. Guillen, Ermin Sakic, and Georg Sigl. 2014. Ciphertext-Only Fault Attacks on PRESENT. In *Lightweight Cryptography for Security and Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers (Lecture Notes in Computer Science)*, Thomas Eisenbarth and Erdinç Öztürk (Eds.), Vol. 8898. Springer, 85–108. DOI:http://dx.doi.org/10.1007/978-3-319-16363-5_6

Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. 2011. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. In *WISTP (Lecture Notes in Computer Science)*, Claudio Agostino Ardagna and Jianying Zhou (Eds.), Vol. 6633. Springer, 224–233.

An Wang, Man Chen, Zongyue Wang, and Xiaoyun Wang. 2013. Fault Rate Analysis: Breaking Masked AES Hardware Implementations Efficiently. *Circuits and Systems II: Express Briefs, IEEE Transactions on* 60, 8 (2013), 517–521.