Blind Fault Attack against SPN Ciphers

Roman Korkikian*†, Sylvain Pelissier*, David Naccache‡

* *Kudelski Security*
Route de Genève 22-24, 1033 Cheseaux, Switzerland
firstname.lastname@nagra.com

†*Université Panthéon Assas*
12 Place du Panthéon, 75005, Paris, France.
roman.korkikian@etudiants.u-paris2.fr

‡*École normale supérieure*
Département d'informatique
45, rue d'Ulm, 75005, Paris CEDEX *05, France.*
david.naccache@ens.fr

***Abstract*—This paper presents a novel fault attack against Substitution Permutation Networks. The main advantage of the method is an absence of necessity to know the exact cipher's input and output values. The attack relies only on the number of faulty ciphertexts originated from the same unknown plaintext. The underlying model is a multiple bit-set or bit-reset faults injected several times at the same intermediate round state. This method can be applied against any round thus any round key can be extracted. The attack was shown to be efficient by simulation against several SPN block ciphers.**

***Keywords*-SPN, AES, LED, SAFER++, Block cipher, Differential Fault Analysis, Collision Fault Analysis, Side Channel Analysis**

I. INTRODUCTION

One of the first examples of faults being injected into a chip was accidental. It was noticed that radioactive particles produced by elements naturally present in packaging material [1] caused faults in chips. Specifically, Uranium-235, Uranium-238 and Thorium-230 residues present in the packaging decay to Lead-206 while releasing α particles. These particles create a charge in sensitive chip areas causing bits to flip. Whilst these elements were only present in two or three parts per million, this concentration was sufficient to affect chip behavior. Subsequent research included studying and simulating the effects of cosmic rays on semiconductors [2]. Cosmic rays are very weak at ground level due to the earths atmosphere, but their effect becomes more pronounced in the upper atmosphere and outer space. This problem is further compounded by the fact that the more RAM a computer has the higher the chance of a fault occurring. This has provoked a great deal of research by organizations such as NASA and Boeing. Most of the work on fault resistance was motivated by this vulnerability to charged particles. Considerable engineering endeavors were devoted to the hardening of electronic devices designed to operate in harsh environments. This has mainly been done using simulators to model circuits and study the effect of randomly induced faults. Various fault induction methods have since been discovered but all have in common similar effects on chips. One such example is the use of a laser to imitate the effect of charged particles [3]. The different faults that can be produced have been characterized to enable the design of suitable protections. The first attack that used a fault to derive secret information [4] targeted the RSA public-key cryptosystem. Basically, a fault was introduced to reveal the two secret prime numbers that compromised the RSA system. This led to similar attacks on other cryptographic algorithms. In particular, Fault Attacks (FAs) were adapted to Substitution Permutation Networks (SPNs) [5]. SPN-based algorithms apply several independent transformations that can be efficiently implemented in both software and hardware [6]. Experience, along with many years of cryptanalysis effort, indicates that SPNs are a good block cipher build-blocks [7]–[11], even though they are vulnerable to fault attacks. The two main FAs against SPN are Differential Fault Analysis (DFA) and Collision Fault Analysis (CFA). These attacks have been mostly developed for the AES but have been later adapted for other SPN ciphers.

DFA requires a pair of correct and faulty ciphertexts that are the result of the same plaintext encryption [5], [12], [13]. Since two encryptions perform identically up to the fault injection point, the two ciphertexts can be considered as the outputs of a reduced-round block cipher where the inputs are unknown but have a small difference [14]. Analyzing the propagation of this difference (called differential) over the small number of rounds, an attacker can gain key information involved in these rounds.

When the same plaintext cannot be encrypted twice, a ciphertext-based attack remains practical as shown by [15]. A bias introduced at the input of an S-box can be used to distinguish the correct key. Because the S-box is a non-linear transformation then an input's entropy computed from all the faulty results with a wrong key guess would be undistinguishable from the entropy of a uniformly distributed

978-1-4799-6292-1/14 $31.00 © 2014 IEEE
DOI 10.1109/FDTC.2014.19

variable. The input's entropy computed for the correct key candidate shall be different due to the introduced bias. The multiple bit-reset fault model considered in [15] is a typical fault that can introduce a bias.

While DFA uses a fault injected at the last SPN rounds, CFA exploits errors at the beginning of encryption [16], [17]. CFA looks for a collision between genuine and faulty encryptions of plaintexts M and $\tilde{M}$ respectively. Since the encryptions perform differently up to the point when the fault compensates both state values, the two plaintexts could be considered as the inputs to a reduced-round block cipher that outputs a predictable differential after several rounds.

To the authors' best knowledge all FAs against SPNs require either full control over the cipher's input and/or the opportunity to inspect the encryption's result. Restricting the attacker's access to plaintext and ciphertext is considered as good countermeasure for systems where the secret data can not be easily modified, for example a shared root key used in UMTS [18]. However, these countermeasures cannot protect the system against all FAs.

1) Our contribution: In this paper we present an attack that does not require a direct access to cipher's input and output. The attack assumes the following: (1) an attacker can encrypt several unknown plaintexts multiple times under the same key; (2) encryption results can be compared between themselves without disclosing their values (this is somewhat similar to the "generic model" often used in public-key cryptography [19]); (3) a multiple random bit-reset or bit-set fault can be injected during the encryption rounds.

Our method infers information from the relationship between the number of faulty ciphertexts originated from the same unknown plaintext and an intermediate state's Hamming weight. We show that each SPN round comprises a key-involved operation that can reveal the round key if input and output Hamming weights of this operation are known. Simulations show that this attack is practical against LED [9], AES [8] and SAFER++ [20] algorithms.

The attack is performed in two phases: fault injection and key search. The fault injection phase is used to determine the Hamming weights of intermediate states. When a number of fault injections is limited, we determine an occurrence probability for each possible Hamming weight value. The key search phase is done in two steps: The first step applies a finite field equation to filter-out key candidates. The second step assigns likelihood information to each key candidate which, in turn, reveals the correct key with high probability.

2) Paper organisation: The paper is organized as follows. Section II recalls Substitution Permutation Networks and discusses previous fault-based attacks. Section III introduces the new FA. Section IV explains how to determine the fault injection point and the natural countermeasures against this attack. Section V concludes the paper.

II. Preliminaries

A. Substitution-permutation networks

A Substitution Permutation Network (SPN) is a sequence of invertible transformations used in symmetric key cryptosystems. An SPN is made up of *confusion* and *diffusion* stages. Coupled with key mixing operation, confusion and diffusion transforms form a round. Iteratively applied rounds yield an invertible mapping $f_{\mathcal{K}} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ called the *block cipher*, where $\mathcal{K}$ is a block cipher key, usually referred to as a *master* key, n is the size of plaintext and ciphertext in bits.

A confusion stage or *S*-box, denoted as **S**, is a non-linear bijective transformation used to map a b-bit element into another b-bit element:

$$\mathbf{S} : \mathbb{F}_2^b \to \mathbb{F}_2^b \tag{1}$$

During the confusion stage the current n-bit string is fed into a series of m *S*-boxes, where $n = bm$. The same set of *S*-boxes may be used in each round, or the *S*-boxes may change from round to round.

The diffusion layer, denoted as **P**, is a linear transform that reshuffles n-bit inputs.

$$\mathbf{P} : \left(\mathbb{F}_2^b\right)^m \to \left(\mathbb{F}_2^b\right)^m \tag{2}$$

The main purpose of diffusion is to spread small input variations over a significant amount of output bits. Permutation **P** is designed so that the output bits of any given S-box are spread over different S-boxes in the next round.

A key mixing operation, denoted as **A**, combines the n-bit input with an n-bit round key K_i.

$$\mathbf{A} : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n \tag{3}$$

The sub-keys K_i are derived from the master key $\mathcal{K}$ according to the *key schedule* algorithm. The key schedule is often made of a simple confusion-diffusion operations set.

A typical SPN-based block cipher, shown on Figure 1, can be described by the equation (4).

$$\mathbf{F}_{\mathcal{K}} : \mathbf{A} \mid_{K_R} \circ \mathbf{S} \circ \left(\bigcirc_{r=1}^{R-1} \mathbf{A} \mid_{K_r} \circ \mathbf{P} \circ \mathbf{S}\right) \circ \mathbf{A} \mid_{K_0} \tag{4}$$

Note that the very first and last operations performed in this SPN are sub-key mixing operations. This is called *whitening* and is regarded as a useful way to prevent an attacker from even beginning to carry out an encryption or decryption operation if the key is not known.

In the last round, the permutation **P** is not applied. Consequently, the encryption algorithm can also be used for decryption, if appropriate modifications are made for the key schedule and if permutations are replaced be their inverses.

The following definitions are used along the paper.

Hamming weight Let X be an array of M elements $\mathrm{X} = [x_0, x_1, \ldots, x_{M-1}]$. The number of non-zero elements in X, known as the Hamming weight of X, is denoted by $\mathcal{HW}(\mathrm{X})$.

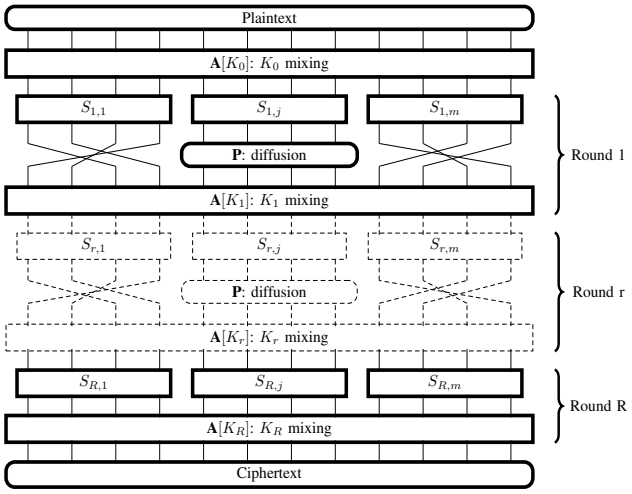


Figure 1: A typical SPN-based block cipher.

Entropy Suppose $\mathbf{X}$ is a discrete random variable which takes values from $\mathbb{F}_2^b$. Then, the *entropy* of a random variable $\mathbf{X}$ is defined to be the following quantity in bits

$$H(\mathbf{X}) = -\sum_{x \in \mathbb{F}_2^b} \mathbf{Pr}[\mathbf{X} = x] \log_2(\mathbf{Pr}[\mathbf{X} = x]) \qquad (5)$$

Note that if variable $\mathbf{X}$ is uniformly distributed, *i.e.* $\mathbf{Pr}[\mathbf{X} = x] = \frac{1}{2^b}$ for all $x \in \mathbb{F}_2^b$, then $H(\mathbf{X}) = b$.

T-radical branch number A T-radical branch number $\mathcal{B}_T$ of a linear diffusion layer $\mathbf{P}$ is defined as:

$$\mathcal{B}_T(\mathbf{P}) = \min_{\mathcal{HW}(x)=T} \{\mathcal{HW}(\mathbf{P}(x))\}, \quad x \in \left(\mathbb{F}_2^b\right)^m$$

A definition of leakage-immunity similar to one given in [21] is given below. We consider an operation $F(k, m)$ taking an unknown key k and a message m as arguments. A definition of Hamming Weight Probability Distribution (HWPD) for operation F which will be used in the section III is also recalled below.

Leakage immune operation F Operation $F(k, m)$ is leakage immune if for all distributions (k, m) and (k', m') the distributions $F(k, m)$ and $F(k', m')$ are statistically indistinguishable.

Hamming weight probability distribution for operation F A Hamming Weight Probability Distribution (HWPD) for operation F

$$\mathbf{Pr}[\mathcal{HW}(x), \mathcal{HW}(F(k, x))] \qquad (6)$$

is a bivariate probability distribution to obtain a pair $(\mathcal{HW}(x), \mathcal{HW}(F(k, x)))$ for a given key $k \in \mathbb{F}_2^b$ and a uniform input $x \in \mathbb{F}_2^b$.

We will use the following notations:

n	block size in bits
R	number of SPN cipher rounds
b	S-box input/output size (**S** stands for the S-box)
m	number of S-box invocations per round
$\mathbf{P}$	the linear diffusion
$\mathbf{A}$	the key mixing
K_r	r-round key
X_r	r-round input
X_r^S	r-round state after S-box
X_r^{SP}	r-round state after linear diffusion
$\mathcal{HW}(x)$	Hamming weight of x
$H(\mathbf{X})$	the entropy of $\mathbf{X}$
Δ	a random variable defined over a subset of $\mathbb{F}_2^b$ (an error space)
$\delta \in \mathbb{F}_2^b$	an outcome of Δ *i.e.* a computational error
$\mathcal{B}_T(\mathbf{P})$	T-radical branch number of a diffusion layer
K^*	a correct key
$(C^i, \tilde{C}^i)$	outputs of correct and faulty encryptions of an identical plaintext
$(M^i, \tilde{M}^i)$	inputs of correct and faulty encryptions that gave the same ciphertext

When indexed by $j \in [1, m]$, the notations $K_{r,j}$, $X_{r,j}$, $X_{r,j}^S$, $X_{r,j}^{SP}$ refer to S-box number j used to indicated parts of $K_r, X_r, X_r^S, X_r^{SP}$ (see Figure 2).

B. Ciphertext-based attacks against SPN

This section recalls ciphertext-based attacks against SPN ciphers.

1) Attack on round R: We consider a fault δ introduced before S-box transformation in the last round R. Since all the SPN operations are reversible the correct X_R and faulty $\delta \oplus X_R$ states can be computed as follows:

$$\begin{aligned} X_R &= \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_{K_R} (C) \\ \delta \oplus X_R &= \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_{K_R} (\tilde{C}) \end{aligned} \qquad (7)$$

Equations (7) can be considered as a round-reduced cipher where two known inputs $C \neq \tilde{C}$ produce an output with a predictable differential δ. For a given key candidate K and a ciphertext pair $(\tilde{C}^i, C^i)$ we have:

$$\delta_K^i = \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_K (\tilde{C}^i) \oplus \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_K (C^i) \qquad (8)$$

The introduced error δ_i is assumed to be the outcome a non uniformly distributed variable Δ, *i.e.* $H(\Delta) = \mathcal{H} < b$. Given the non-linear property of the S-box the corresponding set of errors $\delta_{K \neq K^*}^i$ computed for all the pairs $(C_i, \tilde{C}_i)$ with a wrong key shall be uniformly distributed [22]. The set of errors $\delta_{K=K^*}^i$ computed with the correct key shall be non uniformly distributed; hence, the error entropy can be used for key selection.

$$\lim_{i \to +\infty} H(\Delta_K) = \begin{cases} b & \text{if} \quad K \neq K^* \\ \mathcal{H} & \text{if} \quad K = K^* \end{cases} \qquad (9)$$

The general entropy approach is described in [23], while a special case when $H(\Delta_{K*}) = 1$ is described in [5].

When the same plaintext cannot be encrypted twice, ciphertext-based attacks still apply. In that case the fault has to corrupt the uniformity of the entire input so the entropy would be smaller for the correct key guess as stated by [15]. The fault that can corrupt the uniformity of an entire input is the AND and OR fault models:

$$\begin{aligned} \tilde{X}_R &= \delta \wedge X_R \\ \tilde{X}_R &= \delta \vee X_R \end{aligned}$$

2) Attack on round $R-1$: Another generally considered ciphertext-based attack exploits computational errors before the last $\mathbf{P}$ permutation at round $R-1$. Both correct X^S_{R-1} and faulty $\delta \oplus X^S_{R-1}$ states can be computed from the known ciphertexts:

$$\begin{aligned} X^S_{R-1} &= \mathbf{P}^{-1} \circ \mathbf{A}^{-1} \mid_{K_{R-1}} \circ \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_{K_R} (C) \\ \delta \oplus X^S_{R-1} &= \mathbf{P}^{-1} \circ \mathbf{A}^{-1} \mid_{K_{R-1}} \circ \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_{K_R} (\tilde{C}) \end{aligned}$$

Note that the error space is a subset of $\left(\mathbb{F}_2^b\right)^{\mathcal{B}_{\mathcal{HW}(\delta)}(\mathbf{P})}$, where $\mathcal{B}_{\mathcal{HW}(\delta)}(\mathbf{P})$ is a $\mathcal{HW}(\delta)$-branch number. The differential δ^i can be written as a function of $K_{R-1}, K_R, C^i, \tilde{C}^i$ as shown by equation (10).

$$\begin{aligned} \delta^i = \; & \mathbf{P}^{-1} \circ \mathbf{A}^{-1} \mid_{K_{R-1}} \circ \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_{K_R} (\tilde{C}^i) \oplus \\ & \mathbf{P}^{-1} \circ \mathbf{A}^{-1} \mid_{K_{R-1}} \circ \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_{K_R} (C^i) \end{aligned} \tag{10}$$

To recover the correct key, the variable $\delta_{K=K^*}$ obtained for the correct key has to be at least distinguishable from the variable $\mathbf{X}$ uniformly distributed over $\left(\mathbb{F}_2^b\right)^{\mathcal{B}_{\mathcal{HW}(\delta)}(\mathbf{P})}$. Since equation (10) exploits both K_{R-1}, K_R round keys the search key space has to be squared to $\left(\mathbb{F}_2^b\right)^{2 \cdot \mathcal{B}_{\mathcal{HW}(\delta)}(\mathbf{P})}$.

The expression (10) can be simplified if the key mixing operation is a bit-wise exclusive or (XOR) between a round key and a state, *i.e.* $\mathbf{A} \mid_K \circ X : K \oplus X$, and permutation $\mathbf{P}$ is linear with respect to XOR:

$$\begin{aligned} \delta^i = \; & \mathbf{P}^{-1} \big(\mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_{K_R} (\tilde{C}^i) \oplus \\ & \mathbf{S}^{-1} \circ \mathbf{A}^{-1} \mid_{K_R} (C^i) \big) \end{aligned} \tag{11}$$

In this case the candidate key space is reduced to $\left(\mathbb{F}_2^b\right)^{\mathcal{B}_{\mathcal{HW}(\delta)}(\mathbf{P})}$.

The attack described by equation (11) can be applied against AES if an error has entropy smaller than 32. The fault when up to three out of four bytes of MixColumn's input are modified is described in [5], [13], [24], all 4 bytes modification is presented in [25], [26].

C. Plaintext-based attacks against SPN

Plaintext-based attacks, namely CFA, can be used when a cipher's output cannot be directly accessed but compared with previous encryption results. The main difference with ciphertext-based attacks is that an attacker inject faults in early SPN rounds. When a correct and tampered encryptions of different plaintexts returned the same result an attacker can gain secret key information.

1) Attack on first round: Similar to the previously described last round attack we consider a fault δ introduced after S-box transformation at the first SPN round. The correct X^S_1 and faulty $\tilde{X}^S_1$ states can be computed as follows:

$$\begin{aligned} X^S_1 &= \mathbf{S} \circ \mathbf{A} \mid_{K_0} (M) \\ \tilde{X}^S_1 &= \mathbf{S} \circ \mathbf{A} \mid_{K_0} (\tilde{M}) \end{aligned} \tag{12}$$

Equation (12) can be considered as a round-reduced cipher where two known inputs $M \neq \tilde{M}$ produce an output with a predictable differential δ. An injected error δ modified the state $\tilde{X}^S_1$ so that:

$$\begin{aligned} X^S_1 &= \tilde{X}^S_1 \oplus \delta \\ \delta &= X^S_1 \oplus \tilde{X}^S_1 \end{aligned}$$

$$\delta = \mathbf{S} \circ \mathbf{A} \mid_{K_0} (M) \oplus \mathbf{S} \circ \mathbf{A} \mid_{K_0} (\tilde{M}) \tag{13}$$

For a given key byte candidate K and a byte pair $(\tilde{M}^i, M^i)$ we have:

$$\delta^i_K = \mathbf{S} \circ \mathbf{A} \mid_K (\tilde{M}^i) \oplus \mathbf{S} \circ \mathbf{A} \mid_K (M^i)$$

The errors δ_i are assumed to be non uniformly distributed, *i.e.* $H(\Delta) = \mathcal{H} < b$. Given the properties of S-box the corresponding set of errors $\delta^i_{K \neq K^*}$ computed with a wrong key for all the encryptions of $(M^i, \tilde{M}^i)$ resulted in ciphertext collision shall be uniformly distributed. The set of errors $\delta^i_{K=K^*}$ computed with the correct key shall converge to non uniform distribution. Therefore the error entropy can be used to distinguish the correct key candidate:

$$\lim_{i \to +\infty} H(\Delta_K) = \begin{cases} b & \text{if} \quad K \neq K^* \\ \mathcal{H} & \text{if} \quad K = K^* \end{cases} \tag{14}$$

A special case when $H(\Delta_{K*}) = 1$ is described in [17].

To the authors' best knowledge both the general entropy case [23] and faulty ciphertexts only attack [15] have not been adapted for the first SPN round yet. A comparison of equations (7) and (12) shows that this adaptation is indeed possible as suggested by [14].

III. Fault based attacks of substitution permutation network

A. Substitution layer leakage

Our attack targets an SPN operation between two rounds, r and $r+1$, shown in Figure 2 in blue. This operation can be described by equation (15).

$$X^S_{r+1,j} = \mathbf{S}_{r+1,j} \circ \mathbf{A} \mid_{K_{r,j}} \left(X^{SP}_{r,j} \right), \quad j \in [1, m] \tag{15}$$

For simplicity this cryptographic operation is denoted as:

$$F(K_{r,j}, X^{SP}_{r,j}) = \mathbf{S}_{r+1,j} \circ \mathbf{A} \mid_{K_{r,j}} \left(X^{SP}_{r,j} \right) \tag{16}$$

The input $X^{SP}_{r,j}$ and output $X^S_{r+1,j}$ size is 4-bit for the ciphers LED [9] and KLEIN [11] while 8-bit input and output variables are used in AES [8] and SAFER++ [20]. However any other possible S-box input size can be considered.

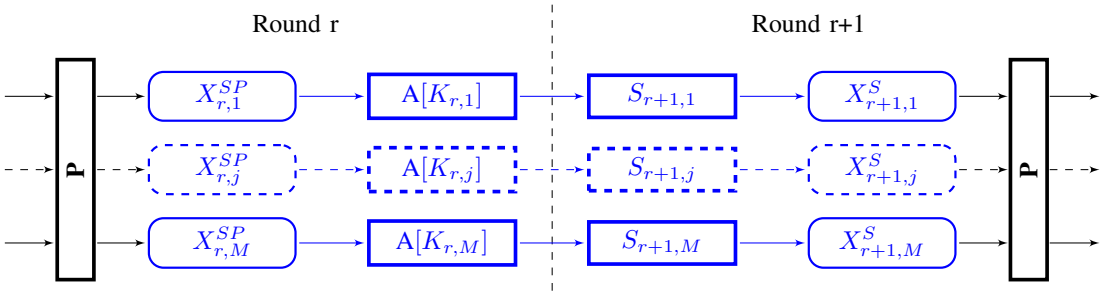


Figure 2: Confusion operation at round $r+1$.

We want to show that a Hamming weight pair $\big(\mathcal{HW}(X^{SP}_{r,j}), \mathcal{HW}(F(K_{r,j}, X^{SP}_{r,j}))\big)$ can be used to distinguish key values $K_{r,j}$. The easiest way to demonstrate this is to consider zero Hamming weight, *i.e.* $X^{SP}_{r,j} = 0$. In that case, only one key $K^{'}_{r,j}$ will provide a zero Hamming weight output. Similarly, only one key $K^{''}_{r,j}$ will result in output which Hamming weight equals b (all bits of output equal to one), thus keys $K^{'}_{r,j}$ and $K^{''}_{r,j}$ can be distinguished. This dependency is illustrated with AES.

$$F(K_{r,j}, X^{SP}_{r,j}) = \mathbf{S}[K_{r,j} \oplus X^{SP}_{r,j}], \quad j \in [1, m] \tag{17}$$

Figure 3 illustrates the HWPD for AES computed for two different key values. Clearly, these distributions are different and can hence serve for key differentiation. Due to the nature of S-boxes, similar results are observed on other SPN block ciphers such as LED or SAFER++.

HWPD key dependency can be exploited when the Hamming weight of the operation's input and output are known to the attacker. So the first problem is to find the input and output Hamming weights for operation F. To obtain a Hamming weight of an intermediate state a multiple bit-reset fault model can be used:

$$\tilde{X}^{\Omega}_{r,j} = X^{\Omega}_{r,j} \wedge e \quad X^{\Omega}_{r,j}, e \in \mathbb{F}^b_2 \tag{18}$$

where X^{Ω} is a part of the state in round r after any SPN transformation. The maximum number of possible values for $\tilde{X}^{\Omega}_{r,j}$, denoted by $\lambda = \#\tilde{X}^{\Omega}_{r,j}$, is a function of the data block's Hamming weight $h^{\Omega}_{r,j}$, as shown by equation (19).

$$\lambda = \#\tilde{X}^{\Omega}_{r,j} = 2^{h^{\Omega}_{r,j}} \tag{19}$$

Since each value of $\tilde{X}^{\Omega}_{r,j} = X^{\Omega}_{r,j} \wedge e$ will lead to a different output, equation (19) also links the maximum number of observed ciphertexts and the Hamming weight of $X^{\Omega}_{r,j}$. For byte values the maximum number of different ciphertexts is $\lambda \in \{1, 2, \ldots, 256\}$ while for nibbles $\lambda \in \{1, 2, \ldots, 16\}$.

In this paper the fault injection is used to determine the Hamming weight of the input and output state-parts $X^{SP}_{r,j}$ and $X^{S}_{r+1,j}$ used in equation (15). The manner in which Hamming weight is determined by fault injection is illustrated in the next subsection III-B. Once the Hamming weight of the pair $X^{SP,1}_{r,j}, X^{S,1}_{r+1,j}$ is found, the attacker can search the correct key value using key sifting and key likelihood information, as will be discussed in the subsection III-C.

B. Hamming weight computation

We assume that a multiple bit-reset error[1] e can be invoked in the a middle cipher round:

$$\tilde{X}^{\Omega}_{r,j} = X^{\Omega}_{r,j} \wedge e \quad X^{\Omega}_{r,j}, e \in \mathbb{F}^b_2 \tag{20}$$

The error e is assumed to be uniformly distributed over the space $\mathbb{F}^b_2$, hence all ciphertexts have equal appearance probabilities. We assume that the value $\tilde{X}^{\Omega}_{r,j}$ cannot be directly accessed, so the attack's principal idea is to determine the Hamming weight of this variable by injecting ℓ random multiple bit-reset faults and observing the number of different outcoming ciphertexts.

We assume that Y_ℓ different ciphertexts are observed after ℓ fault injections. We want to compute the probability that faults injected in a variable with the Hamming weight $\lambda = 2^{h^{\Omega}_{r,j}}$ could produce Y_ℓ different ciphertexts. This can be considered as an occupancy problem [27] where Y_ℓ out of λ bins are occupied after throwing ℓ balls.

In the classical occupancy problem, the probability $\mathbf{Pr}(Y_\ell = \kappa)$ can be computed using equation (21) given in [28].

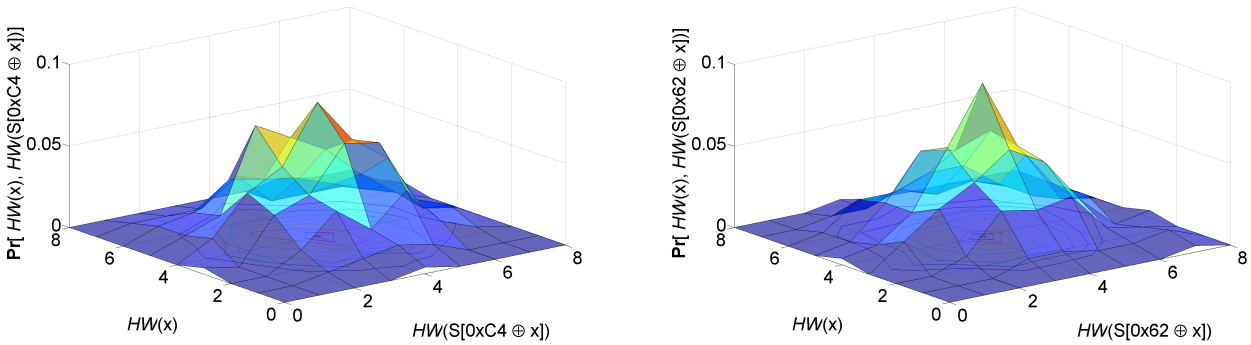
$$\mathbf{Pr}(Y_\ell = \kappa) = \begin{cases} \frac{\lambda! \alpha_{\kappa,\ell}}{(\lambda-\kappa)!\lambda^\ell} & \kappa \in \{1, ..., \min(\lambda, \ell)\} \\ 0 & \text{else} \end{cases} \tag{21}$$

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

$$\alpha_{\kappa,\ell} = \frac{1}{\kappa!} \sum_{i=1}^{\kappa} (-1)^{\kappa-i} \binom{\kappa}{i} i^\ell$$

In our case the values Y_ℓ and ℓ are known but λ must be determined. To estimate λ we build a maximum likelihood estimator $\hat{\lambda}$ as a function of κ and ℓ, *i.e.* we compute equation (21) for all the values $\lambda_i \geq 2^{\lceil \log_2(\kappa) \rceil}$ and amongst

[1]The attack is also working with the multiple bit-set fault model



(a) HWPD distribution for the key byte `0xC4`: $\mathbf{Pr}\left[\mathcal{HW}(X), \mathcal{HW}(\mathbf{S}[X \oplus \texttt{0xC4}])\right]$, $X \in [0, 255]$

(b) HWPD distribution for the key byte `0x62`: $\mathbf{Pr}\left[\mathcal{HW}(X), \mathcal{HW}(\mathbf{S}[X \oplus \texttt{0x62}])\right]$, $X \in [0, 255]$

Figure 3: Hamming weight probability distribution for the AES operation $\mathbf{S}[X \oplus k]$.

them the λ_i with the maximum probability is assumed to be correct:

$$\hat{\lambda} = \arg\max_{\lambda_i} \mathbf{Pr}(Y_\ell = \kappa | \lambda_i) \tag{22}$$

The above Hamming weight detection method was simulated for nibbles and bytes. A given number ℓ of randomly generated multiple bit-reset faults were injected into a randomly generated variable x and the number of different faulty values Y_ℓ were used to determine the Hamming weight of the variable using formula (22). The total number of successfully determined Hamming weights was recorded for 10^5 trials and the success rate was computed for each number of faults ℓ as shown on Figure 4.

On the average 15 faults sufficed to detect the Hamming weight of a nibble with a 99% probability. Determining bytes with the same probability required 62 faults.

The occupancy problem and various estimators were previously discussed in [29], however the maximum likelihood estimator was chosen due to the limited number of possible bins and computational simplicity.

C. Key search

An attacker has N pairs of Hamming weights $\left(\mathcal{HW}(X^{SP,i}_{r,j}), \mathcal{HW}(X^{S,i}_{r+1,j})\right), i \in [1, N]$ received for the same operation $X^{S,i}_{r+1,j} = \mathbf{S}_{r+1,j} \circ \mathbf{A} \mid_{K_{r,j}} \left(X^{SP,i}_{r,j}\right)$. In the following part, the notations $\left(h^i_r, h^i_{r+1}\right)$ and $\left(\mathcal{HW}(X^{SP,i}_{r,j}), \mathcal{HW}(X^{S,i}_{r+1,j})\right)$ are interchangeable.

The key search process is performed in two steps: The first step, called *key sifting*, is a typical equation-based approach when the key has to satisfy a set of equations. The present attack uses N pairs $\left(h^i_r, h^i_{r+1}\right)$ to find the key candidates that satisfy the following constraint:

$$\begin{aligned} \mathcal{L} \quad = \quad & \left\{k \in \mathbb{F}_2^b : \ \forall i \in [1, N] \quad \exists x \in \mathbb{F}_2^b, \right. \\ & \mathcal{HW}(x) = h^i_r, \\ & \left. \mathcal{HW}(\mathbf{S}_{r+1,j} \circ \mathbf{A} \mid_k (x)) = h^i_{r+1} \right\} \end{aligned} \tag{23}$$

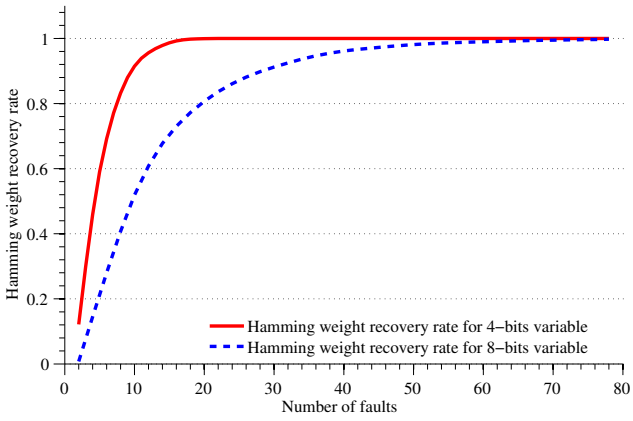


Figure 4: Results of Hamming weight computation by fault injection.

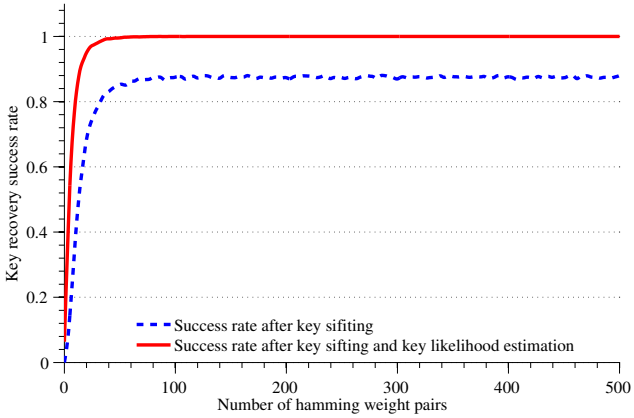
Reducing $|\mathcal{L}|$ requires a significantly higher number of pairs than ciphertext- or plaintext-based attacks.

To perform second step, called *key likelihood estimation*, the HWPD $\mathbf{Pr}_k\left[\mathcal{HW}(x), \mathcal{HW}\left(\mathbf{S}_{r+1,j} \circ \mathbf{A} \mid_k (x)\right)\right]$ is precomputed for each key value $k \in \mathbb{F}_2^b$ and uniformly distributed $x \in \mathbb{F}_2^b$. During this step the probability distribution function is computed for the list of obtained Hamming weight pairs $\mathbf{Pr}_r\left[h^j_r, h^j_{r+1}\right]$. Then the Euclidean distance between the $\mathbf{Pr}_r$ and $\mathbf{Pr}_k$ is computed for each key from the list $k \in \mathcal{L}$:

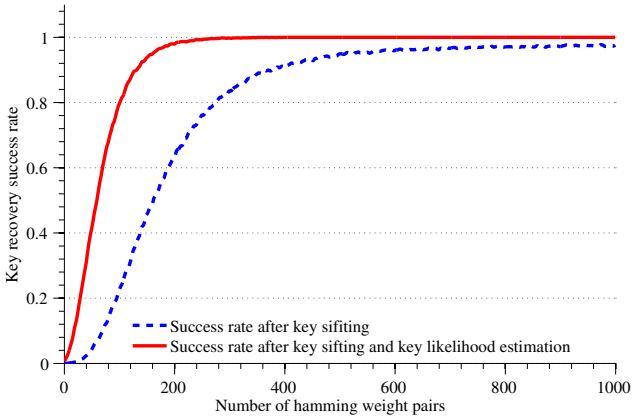
$$D(\mathbf{Pr}_r, \mathbf{Pr}_k) = \sqrt{\sum_{\forall h_i, h_j} \left(\mathbf{Pr}_r\left[h_i, h_j\right] - \mathbf{Pr}_k\left[h_i, h_j\right]\right)^2} \tag{24}$$

and the key candidate with the minimum distance is betted upon as correct:

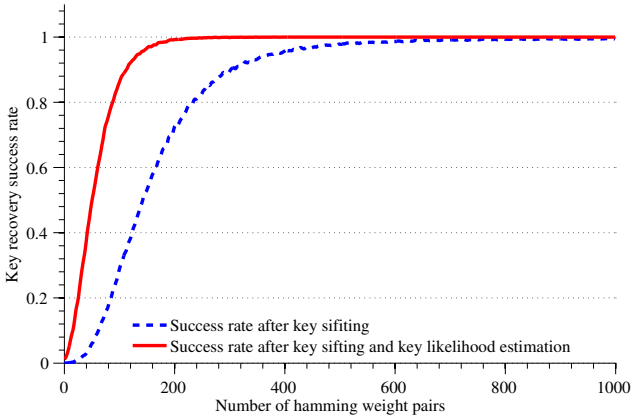
$$\hat{k} = \arg\min_k D(\mathbf{Pr}_r, \mathbf{Pr}_k) \tag{25}$$



(a) LED



(b) AES



(c) SAFER++

Figure 5: Key recovery success rate for different S-boxes $\mathbf{S}_{r+1,j} \circ \mathbf{A} \mid_{K_{r,j}} \left(X_{r,j}^{SP}\right)$.

D. Simulation

The key search algorithm was simulated for LED, AES and SAFER++. Operation (15) of each cipher is described in the Table I. Note that LED and AES uses bit-wise exclusive or as key mixing operation, while SAFER++ applies byte addition. SAFER++ uses two kind of S-boxes based on discrete logarithm and exponentiation. In our test we used discrete logarithm-based S-box with 256 elements.

The key search was performed with known Hamming weights $\left(\mathcal{HW}(X_r^{SP,i}), \mathcal{HW}(X_{r+1}^{S,i})\right)$ that could have been recovered at an earlier step by fault injection. The successful key recovery was recorded after key sifting and key likelihood estimation and shown in the Figure 5 for the various ciphers considered.

As illustrated on Figure 5, key likelihood estimation significantly improves key recovery success rate. Moreover, key sifting does not converge to 1, which justifies the usage of the key likelihood estimation step. The average number of Hamming weight pairs needed to recover the correct key with 99% confidence is 50 for LED, 250 for AES and 200 for SAFER++.

Simulations reveal that our attack can be used to recover round keys. The total number of required faults, given in Table II, depends on the cipher's S-box input size, key mixing operation and the number of elements in S-box. For example, to recover all the sixteen AES key bytes, we would need 480,000 faults.

IV. Feasibility of the Attack

This section discusses practical aspects of the proposed fault attack, namely multiple bit-set or bit-reset fault models, precise fault injection time and the required number of faulty ciphertexts.

1) Multiple bit-set or bit-reset fault model: One of the attack's main assumptions is that a multiple bit-reset (or multiple bit-set) can be invoked by fault injection. Previously, these fault models have been applied by various papers [12], [15]. The practical feasibility of bit-reset (or bit-set) fault injection was shown in a set of experiments. The multiple bit-set fault model was observed during EM-glitch fault injection as described in [30]. [31] reports that during laser fault injection to SRAM, bit-flip fault model is irrelevant, only bit-set (or bit-reset) errors are feasible.

2) Precise fault injection time and space: A second attack requirement is precise fault injection, *i.e.* the time and the location of a fault must be well specified. This is a single fault attack since only one fault has to be injected during an encryption. The identification of fault injection time and place can be done during the characterization phase when the adversary has the full control over the device. To identify the processing time of the state value $X_{r,j}^{\Omega}$ an adversary may use side-channel based reverse-engineering techniques as shown in [14], [32], [33]. Once the time is identified the adversary can search a location for EM or laser fault injection. In order to speed up the identification phase the number of cipher rounds can be reduced. Note that during the characterization phase an adversary may have access to the cipher's input and output but during the evaluation this

Table I: Specification of operation (15) for different ciphers

Cipher	Exact operation	Size of $X^{SP}_{r,j}$, $X^{S}_{r+1,j}$, and $K_{r,j}$	Number of elements in the S-box
LED	$X^{S}_{r+1,j} = \mathbf{S}\left[K_{r,j} \oplus X^{SP}_{r,j}\right]$	4-bit	16
AES	$X^{S}_{r+1,j} = \mathbf{S}\left[K_{r,j} \oplus X^{SP}_{r,j}\right]$	8-bit	256
SAFER++	$X^{S}_{r+1,j} = \mathbf{S}\left[K_{r,j} + X^{SP}_{r,j}\right]$	8-bit	256

Table II: Number of faults used to recover a key from operation (15) for different ciphers

Cipher	Number of plaintexts	Number of faults per plaintext	Total number of faults
LED	50	40	2,000
AES	250	120	30,000
SAFER++	200	120	24,000

information is not accessible; hence, standard fault injection or side-channel attacks cannot be applied.

3) The number of faults: Simulation shows that approximately 120 fault injections are needed before the Hamming weights of the 8-bit input and output state values can be identified. This number of faults has to be multiplied by the number of plaintexts required to recover the key value, *i.e.* the number of Hamming weight pairs needed for key byte recovery. In total approximately 30,000 faults have to be injected before the correct key byte value can be found for the AES and 24,000 faults for SAFER++. This number of faults is significantly higher than for other FAs. However, our attack targets scenarios where other FA methods cannot be applied. The number of faults seems a reasonable price to pay. Once the time and location of fault injections are identified, it is just a matter of time to create this number of errors. In addition, our attack targets individual nibbles or bytes depending the S-box sizes. After recovering several bytes with our method, the rest of the key can be brute-forced.

4) Countermeasures: Our attack is feasible against implementations where the number of different ciphertexts can be counted but their values are inaccessible. The most straightforward countermeasures against the presented method are based on randomization. Infective countermeasures that replace the ciphertext by a random number after a fault recognition is one of them [34]. This countermeasure outputs the correct ciphertext C if no fault happens and when a fault is injected the output is masked with random data $\breve{C} = C \oplus ((C \oplus \tilde{C}) \cdot \eta)$ where η is a random number. With this countermeasure, for the same fault injected, there are multiple different faulty ciphertexts and thus Hamming weight computation cannot be applied.

Another type of countermeasures, which is often used to defeat side channel analysis, is masking [35]. In this case, the operation $F = \mathbf{S}_{r+1,j} \circ \mathbf{A} \mid_{K_{r,j}}$ is changed to $F_\eta = \mathbf{A} \mid_{f(\eta)}$ $\mathbf{S'}_{r+1,j} \circ \mathbf{A} \mid_{K_{r,j} \oplus \eta}$ where η is the randomly generated mask, **S'** is the new layer computed as a function of the mask and $\mathbf{A} \mid_{f(\eta)}$ is the unmasking operation. However, the states $X^{SP,1}_{r,j}$ and $X^{S,1}_{r+1,j}$ change masks for each encryption, thus the Hamming weight can not be determined.

V. Conclusion and further work

In this paper we have described a new fault attack on SPN ciphers that has conservative preliminary assumptions. Namely, the adversary:

- does not know plaintext and ciphertext values.
- can encrypt several times a set of unknown plaintexts.
- knows the number of different results of tampered encryptions performed for the same plaintext.
- can induce a multiple bit-reset (or bit-set) fault in a middle of a SPN round.

We show that under these assumptions the adversary can derive the Hamming weight of an internal round state. When a Hamming weight of the state before key mixing operation and a Hamming weight of the state after confusion operation are known the round key can be recovered. To the authors' best knowledge this is the first fault attack that can be used to derive any round key. Also this is the first attack that is based on Hamming weight of the internal state values.

Simulations confirm that this attack works in practice against AES, LED and SAFER++.

References

[1] T. C. May and M. H. Woods, "Alpha-particle-induced soft errors in dynamic memories," *Electron Devices, IEEE Transactions on*, vol. 26, no. 1, pp. 2–9, 1979.

[2] D. M. Fleetwood and R. D. Schrimpf, *Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices*, ser. Selected topics in electronics and systems. World Scientific Pub., 2004.

[3] C. Gossett, B. Hughlock, and A. Johnston, "Laser simulation of single-particle effects," *Nuclear Science, IEEE Transactions on*, vol. 39, no. 6, pp. 1647–1653, Dec 1992.

[4] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology EUROCRYPT 97*, ser. LNCS, W. Fumy, Ed. Springer Berlin Heidelberg, 1997, vol. 1233, pp. 37–51.

[5] C. Giraud, "DFA on AES," in *Advanced Encryption Standard AES*, ser. LNCS, H. Dobbertin, V. Rijmen, and A. Sowa, Eds. Springer Berlin Heidelberg, 2005, vol. 3373, pp. 27–41.

[6] D. R. Stinson, *Cryptography: Theory and Practice, Third Edition*, ser. Discrete Mathematics and Its Applications. Taylor & Francis, 2005.

[7] J. Katz and Y. Lindell, *Introduction to Modern Cryptography: Principles and Protocols*, ser. Chapman & Hall/CRC Cryptography and Network Security Series. Taylor & Francis, 2007. [Online]. Available: http://books.google.ch/books?id=TTtVKHdOcDoC

[8] NIST, "Advanced encryption standard," *Federal Information Processing Standard, FIPS-197*, vol. 12, 2001.

[9] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The led block cipher," in *Cryptographic Hardware and Embedded Systems CHES 2011*, ser. LNCS, B. Preneel and T. Takagi, Eds. Springer Berlin Heidelberg, 2011, vol. 6917, pp. 326–341.

[10] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ser. LNCS, P. Paillier and I. Verbauwhede, Eds. Springer Berlin Heidelberg, 2007, vol. 4727, pp. 450–466.

[11] Z. Gong, S. Nikova, and Y. Law, "KLEIN: A new family of lightweight block ciphers," in *RFID. Security and Privacy*, ser. LNCS, A. Juels and C. Paar, Eds. Springer Berlin Heidelberg, 2012, vol. 7055, pp. 1–18.

[12] J. Blömer and J.-P. Seifert, "Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)," in *Financial Cryptography*, ser. LNCS, R. N. Wright, Ed. Springer Berlin Heidelberg, 2003, vol. 2742, pp. 162–181.

[13] P. Dusart, G. Letourneux, and O. Vivolo, "Differential Fault Analysis on A.E.S," in *Applied Cryptography and Network Security*, ser. LNCS, J. Zhou, M. Yung, and Y. Han, Eds. Springer Berlin Heidelberg, 2003, vol. 2846, pp. 293–306.

[14] M. Joye and M. Tunstall, Eds., *Fault Analysis in Cryptography*, ser. Information Security and Cryptography. Springer, 2012.

[15] T. Fuhr, E. Jaulmes, V. Lomn, and A. Thillard, "Fault Attacks on AES with Faulty Ciphertexts Only," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*, W. Fischer and J.-M. Schmidt, Eds. IEEE Computer Society, 2013, pp. 108–118.

[16] L. Hemme, "A Differential Fault Attack Against Early Rounds of (Triple-)DES," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, ser. LNCS, M. Joye and J.-J. Quisquater, Eds. Springer Berlin Heidelberg, 2004, vol. 3156, pp. 254–267.

[17] J. Blömer and V. Krummel, "Fault Based Collision Attacks on AES," in *Fault Diagnosis and Tolerance in Cryptography*, ser. LNCS, L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, Eds., vol. 4236. Springer Berlin Heidelberg, 2006, pp. 106–120.

[18] V. Niemi and K. Nyberg, *UMTS Security*. Wiley, 2006.

[19] U. Maurer, "Abstract models of computation in cryptography," in *Cryptography and Coding*, ser. LNCS, N. P. Smart, Ed. Springer Berlin Heidelberg, 2005, vol. 3796, pp. 1–12.

[20] J. Massey, G. Khachatrian, and M. Kuregian, "Nomination of SAFER++ as Candidate Algorithm for the New European Schemes for Signatures, Integrity, and Encryption (NESSIE)," *First Open NESSIE Workshop*, 2000.

[21] J.-S. Coron, P. Kocher, and D. Naccache, "*Statistics and Secret Leakage*," in *Financial Cryptography*, ser. LNCS, Y. Frankel, Ed. Springer Berlin Heidelberg, 2001, vol. 1962, pp. 157–173.

[22] K. Sakiyama, Y. Li, M. Iwamoto, and K. Ohta, "Information-theoretic approach to optimal differential fault analysis," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 1, pp. 109–120, 2012.

[23] R. Lashermes, G. Reymond, J.-M. Dutertre, J. Fournier, B. Robisson, and A. Tria, "A DFA on AES based on the Entropy of Error Distributions," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*, G. Bertoni and B. Gierlichs, Eds. IEEE Computer Society, 2012, pp. 34–43.

[24] G. Piret and J.-J. Quisquater, "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad," in *Cryptographic Hardware and Embedded Systems - CHES 2003*, ser. LNCS, C. D. Walter, e. K. Ko, and C. Paar, Eds. Springer Berlin Heidelberg, 2003, vol. 2779, pp. 77–88.

[25] A. Moradi, M. Shalmani, Mohammad T., and M. Salmasizadeh, "A Generalized Method of Differential Fault Attack Against AES Cryptosystem," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, ser. LNCS, L. Goubin and M. Matsui, Eds. Springer Berlin Heidelberg, 2006, vol. 4249, pp. 91–100.

[26] D. Mukhopadhyay, "An Improved Fault Based Attack of the Advanced Encryption Standard," in *Progress in Cryptology AFRICACRYPT 2009*, ser. LNCS, B. Preneel, Ed. Springer Berlin Heidelberg, 2009, vol. 5580, pp. 421–434.

[27] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed. John Wiley & Sons, Inc., 1968, vol. 1.

[28] B. Harris, "Statistical inference in the classical occupancy problem unbiased estimation of the number of classes," *Journal of the American Statistical Association*, pp. 837–847, 1968.

[29] J. Bunge and M. Fitzpatrick., "Estimating the number of species: A review," *Journal of the American Statistical Association*, vol. 88, no. 421, pp. 364–373, 1993.

[30] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz, "Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*, W. Fischer and J.-M. Schmidt, Eds. IEEE Computer Society, 2013, pp. 77–88.

[31] C. Roscian, A. Sarafianos, J.-M. Dutertre, and A. Tria, "Fault model analysis of laser-induced faults in sram memory cells," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*, W. Fischer and J.-M. Schmidt, Eds. IEEE Computer Society, 2013, pp. 89–98.

[32] R. Novak, "Side-channel based reverse engineering of secret algorithms," in *Proceedings of the Twelfth International Electrotechnical and Computer Science Conference (ERK 2003), Ljubljana, Slovenia, September*. Citeseer, 2003, pp. 25–26.

[33] M. Goldack and I. C. Paar, "Side-channel based reverse engineering for microcontrollers," *Master's thesis, Ruhr-Universität Bochum, Germany*, 2008.

[34] V. Lomne, T. Roche, and A. Thillard, "On the Need of Randomness in Fault Attack Countermeasures - Application to AES," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*, G. Bertoni and B. Gierlichs, Eds. IEEE Computer Society, 2012, pp. 85–94.

[35] H. Kim, S. Hong, and J. Lim, "A Fast and Provably Secure Higher-Order Masking of AES S-Box," in *Cryptographic Hardware and Embedded Systems CHES 2011*, ser. LNCS, B. Preneel and T. Takagi, Eds. Springer Berlin Heidelberg, 2011, vol. 6917, pp. 95–107.