Statistical Effective Fault Attacks: The other Side of the Coin

Navid Vafaei, Sara Zarei, Nasour Bagheri, Maria Eichlseder, Robert Primas and Hadi Soleimany

***Abstract*— The introduction of Statistical Ineffective Fault Attacks (SIFA) has led to a renewed interest in fault attacks. SIFA requires minimal knowledge of the concrete implementation and is effective even in the presence of common fault or power analysis countermeasures. However, further investigations reveal that undesired and frequent ineffective events, which we refer to as the noise phenomenon, are the bottleneck of SIFA that can considerably diminish its strength. This includes noise associated with the attack's setup and caused by the countermeasures utilized in the implementation. This research aims to address this significant drawback. We present two novel statistical fault attack variants that are far more successful in dealing with these noisy conditions. The first variant is the Statistical Effective Fault Attack (SEFA), which exploits the non-uniform distribution of intermediate variables in circumstances when the induced faults are effective. The idea behind the second proposed method, dubbed Statistical Hybrid Fault Attacks (SHFA), is to take advantage of the biased distributions of both effective and ineffective cases simultaneously. Our experimental results in various case studies, including noise-free and noisy setups, back up our reasoning that SEFA surpasses SIFA in several instances and that SHFA outperforms both or is at least as efficient as the best of them. For example, in the case of a 4-bits random-AND fault injected into the AES with a 35 % missed fault rate, utilizing SEFA reduces the number of needed ciphertexts by 50 %. In the same case study, SHFA can yield 10 % and 55 % reductions compared to SEFA and SIFA.**

***Index Terms*— Statistical Fault Attack, SIFA, SEFA, AES**

I. Introduction

Starting with the seminal work by Boneh et al. [2] on RSA, numerous studies have assessed the security of cryptographic implementations against a wide range of fault attacks. The earliest fault attack on symmetric-key cryptosystems is *Differential Fault Analysis* (DFA) [3], which requires concurrent knowledge of a faulty ciphertext and its correct counterpart ciphertext under the same key. A variety of differential fault attack techniques have been applied to symmetric primitives by exploiting the difference between the correct and faulty ciphertexts. Li et al. [4] proposed *Fault Sensitivity Analysis* (FSA) which does not use the value of the faulty ciphertexts in the key retrieval procedure. This works by raising the fault intensity until a distinct feature may be observed that can be used as leakage information. FSA requires a profiling phase and assumes that the adversary has control over the input in the chosen-plaintext scenario. Fuhr et al. introduced a novel technique called *Statistical Fault Attack (SFA)* [5], which exploits the statistical bias introduced by injected faults. Contrary to previous attacks, SFA only requires faulty ciphertexts and is thus applicable in several scenarios where previous attacks are not [6], [7]. For instance, SFA can target the final rounds of AES to practically recover the key with a small number of faulty ciphertexts. In contrast to the aforementioned attacks, *Safe Error Attacks* (SEA) [8]–[10] and *Ineffective Fault Attacks* (IFA) exclusively rely on cases where the injected fault has no effect on the output. Dobraunig et al. [11] recently proposed a new type of fault attack called *Statistical Ineffective Fault Attacks* (SIFA), which combines the concepts of IFA and SFA. A method to accelerate SIFA on AES is proposed in [12].

Amongst all active physical attacks, SIFA seems to be the most simple and effective one from the attacker's point of view but the most intricate one to counteract from the designer's point of view. SIFA and IFA are similar in that the attacker exploits the information whether the fault affected the ciphertext or not, and requires only the correct, unchanged ciphertexts. As with SFA, SIFA exploits the statistical distribution of an intermediate value impacted by the injected fault. The bias ensures that the probabilities of changing/not changing intermediate variables in fault inductions are not uniform. The attack does not require knowledge of the dependency or a specific fault model. The second advantage of SIFA is the result of focusing on ineffective faults, which enables it to bypass several fault countermeasures, including detection-based [13] and infection-based [14]–[16], as well as side-channel countermeasures like masking. In the presence of certain other countermeasures like error correction or dummy rounds, SIFA still works, but becomes less efficient. After the launch of SIFA, one of the notable introduced works was [17]. It exploits the leakages of both faulty and correct ciphertexts to conduct a template attack, and therefore, belongs to the category of Fault Template

N. Vafaei and N.Bagheri are with CPS^2 Lab., Electrical Engineering Dept. of Shahid Rajaee Teacher Training University (SRTTU), Tehran, Iran, e-mail: n.vafaei@sru.ac.ir and Nbagheri@sru.ac.ir, N.Bagheri is also with School of Computer Science (SCS), Institute for Research in Fundamental Sciences (IPM), Farmanieh Campus

S. Zarei is with imec-COSIC, KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium e-mail: firstname.lastname@esat.kuleuven.be

H. Soleimany is with Shahid Beheshti University, Tehran, Iran e-mail: h_soleimany@sbu.ac.ir

M. Eichlseder and R. Primas are with Graz University of Technology, Graz, Austria, e-mail: maria.eichlseder@iaik.tugraz.at and robert.primas@iaik.tugraz.at

This paper has been accepted in IEEE Transactions on Information Forensics and Security (DOI: 10.1109/TIFS.2022.3172634) [1].

TABLE I: Comparison between different fault attacks. CP: Chosen plaintext, CO: Ciphertext only, KP: Known plaintext, ✓*: Correct and faulty ciphertexts are identical for ineffective faults; both can be obtained from a single computation for SIFA and SHFA.

Attack	Data Used for Key Recovery				Bypass Detection Countermeasures	Attack Scenario
	Correct Ciphertexts		Faulty Ciphertexts			
	Effective	Ineffective	Ineffective	Effective		
DFA	✓		✓		✗	CP
FSA	✓		✗		✗	CP
SFA	✗		✓		✗	CO
SIFA	✗	✓*		✗	✓	CO or CP
SEFA	✓	✗	✗	✗	✓	KP or CP
SHFA	✓	✓*		✗	✓	CP

Attacks (FTA). These types of attacks make use of yes/no distributions and have their own appeals and applications, but because they operate in a different scenario compared to SIFA (non-profiled), we leave them for their related scope.

Our Contribution: We introduce a novel technique for fault analysis: Statistical Effective Fault Attacks (SEFA). SEFA makes use of the abandoned portion of the leakage that the attacker has access to when performing SIFA. Surprisingly, this information has been completely overlooked so far. Assume that the correct and faulty encryption of a plaintext P is C and $C^{\mathfrak{f}}$, respectively. When C equals $C^{\mathfrak{f}}$, the output is referred to as *ineffective ciphertext*; otherwise, it is referred to as *effective ciphertext*. In the presence of detection- or infection-based countermeasures, the attacker cannot observe the faulty ciphertexts when the fault is effective. However, the attacker typically retains access to the *correct*, non-faulty ciphertexts in which the fault was effective, even in the presence of these countermeasures. To the best of our knowledge, no comparable fault attack has exploited leakage from such correct ciphertexts (see Table I). SEFA employs non-faulty effective ciphertexts, whereas SIFA employs the set of ineffective ciphertexts. Both SIFA and SEFA exploit the fact that the distribution of an intermediate value over the selected ciphertexts may be non-uniform. Our main observation is that SEFA outperforms SIFA in many practical scenarios, particularly in noisy situations. To get the best of both worlds, we introduce SHFA, a combined SEFA/SIFA attack that outperforms the individual attacks.

Our main contributions in this paper are the following:

1) We propose Statistical Effective Fault Attacks. SEFA is a novel fault attack strategy that is more efficient than SIFA in many circumstances, like noise or certain countermeasures. At the same time, it shares many of the advantages of SIFA, such as the power to bypass state-of-the-art detection and infection countermeasures.
2) We detail how SEFA can circumvent countermeasures, including combined countermeasures and noisy setups.
3) We propose Statistical Hybrid Fault Attacks (SHFA) to combine the advantages of SEFA and SIFA by collecting data for both concurrently and yield the best results.
4) We practically evaluate SEFA, SHFA, and SIFA in different scenarios in both simulations and real experiments on implementations of Keccak/SHA-3 and AES, confirming that SEFA outperforms SIFA in many scenarios, particularly under high missed-fault rates or under countermeasures like error correction and dummy rounds, and that SHFA is generally the most efficient.

The simulations and experiments we conducted to validate our theoretical achievements are all accessible through the GitHub repository `https://github.com/Navidvafaei/SEFA`.

Outline: Section II gives a brief introduction to SIFA and its countermeasures. In Section III, we introduce the novel SEFA approach. Section IV demonstrates its application in different scenarios. Section V introduces the hybrid SHFA attack. In Section VI, we present our experiments and results.

II. Preliminaries

In this section, we first introduce the notation used in this paper. Then, we give a brief overview of SIFA's core ideas and techniques, including fault distributions and statistical hypothesis tests. We also discuss how SIFA interacts with different countermeasures.

A. Statistical Scoring Functions

In statistical attacks like SFA, we evaluate all key candidates for a partial last round key based on the distribution of some intermediate value in the cipher evaluation. This distribution is measured by partially decrypting a given set of ciphertexts under this key candidate. For this purpose, we associate a statistical scoring function $\mathcal{S}(\hat{p})$ with each key candidate, where $\hat{p}$ is the corresponding probability distribution in the intermediate value, e.g., one byte in AES. This scoring function indicates how close the measured distribution is to the expected behaviour under the correct key. Then, the key candidates are ranked according to the metric $\mathcal{S}(\hat{p})$.

We consider two fault scenarios: the attacker may or may not know p, the true statistical distribution of intermediate values corresponding to the correct key. In case p is known, to rank the keys depending on their distribution $\hat{p}$, one can follow [11], [18], [19] and use the log-likelihood ratio (LLR) statistic, which is defined as follows, where θ denotes the uniform distribution and N is the number of samples:

$$\mathrm{LLR}(k) = N \sum_{x \in \mathcal{X}} \hat{p}_k(x) \cdot \log_2 \frac{p(x)}{\theta(x)} \,. \tag{1}$$

In the second scenario, it is assumed that the attacker does not know the details of the faulty distribution except that it is biased when compared to a uniform distribution. In this circumstance, a good metric for recovering the key is Squared Euclidean Imbalance (SEI), which is defined as follows [11], [19], [20]:

$$\mathrm{SEI}(k) = \sum_{x \in \mathcal{X}} (\hat{p}_k(x) - \theta(x))^2 \,. \tag{2}$$

As a result, we use $\mathcal{S}(\hat{p}) = \mathrm{LLR}(k)$ or $\mathcal{S}(\hat{p}) = \mathrm{SEI}(k)$ to rank the key candidates. In both cases, for large N, the distribution of $\mathcal{S}(\hat{p})$ is independently normally distributed for

samples from either p or θ [11], [21]:

$$\mathcal{S}(\hat{p}) \sim \begin{cases} \mathcal{N}(\mu_p, \sigma_p^2) & \text{if } \hat{p} \text{ was produced by } p, \\ \mathcal{N}(\mu_\theta, \sigma_\theta^2) & \text{if } \hat{p} \text{ was produced by } \theta. \end{cases}$$

The difference Δ_a between the score of target key k and the score of the wrong key with rank $2^{\kappa-a}$ of 2^κ possible keys is also normally distributed:

$$\Delta_a \sim \mathcal{N}(\mu_\Delta, \sigma_\Delta^2)\,,$$

where $\mu_\Delta = \mu_p - \mu_\theta - \sigma_\theta \cdot \Phi_{0,1}^{-1}(1-2^{-a})$ and $\sigma_\Delta \approx \sigma_p$. Hence, the success probability of getting an a-bit advantage, that is, limiting the maximum number of candidates examined in the final search phase to $2^{\kappa-a}$ when the number of possible key candidates is 2^κ, is computed as follows [21], [22]:

$$\mathbb{P}(\Delta_a > 0) = \Phi_{0,1}\left(\frac{\mu_p - \mu_\theta - \sigma_\theta \cdot \Phi_{0,1}^{-1}(1-2^a)}{\sigma_p}\right).$$

Assuming that p is very close to θ, the capacity $C(p,\theta)$ is defined as follows:

$$C(p,\theta) = \sum_{x \in \mathcal{X}} \frac{(p(x)-\theta(x))^2}{\theta(x)}\,.$$

Then, the number of ciphertexts required to obtain the success probability $P_\mathrm{S} = \mathbb{P}(\Delta_a > 0)$ can be estimated as follows [11], [22] for LLR and SEI:

$$N_\mathrm{LLR} \approx \frac{2 \cdot [\Phi_{0,1}^{-1}(P_\mathrm{S}) + \Phi_{0,1}^{-1}(\alpha)]^2}{C(p,\theta)} \tag{3}$$

$$N_\mathrm{SEI} \approx \frac{\beta \cdot \Phi_{0,1}^{-1}(\alpha)}{C(p,\theta)} \qquad \text{for } P_\mathrm{S} = 0.5. \tag{4}$$

These estimates assume sufficiently large N, i.e., that p and θ are not too far apart.

B. Statistical Ineffective Fault Attacks (SIFA)

Fuhr et al. [5] introduced the Statistical Fault Attack (SFA), which takes advantage of a bias in an intermediate target value caused by fault induction. Although SFA is more powerful than its preceding generation of fault attack, e.g., DFA, and works on ciphertext-only scenarios, it fails when confronting detection and infection-based countermeasures due to its reliance on a faulty ciphertext. Statistical Ineffective Fault Attack (SIFA) relaxes this reliance. [11] shows that almost all practical fault inductions, whether effective or not, change the targeted intermediate variable to a new value with a non-uniform probability distribution. SIFA then works the same way as SFA, with one exception: SIFA gathers and decrypts ineffective, correct ciphertexts, whereas SFA requires faulty ones.

The efficiency of SIFA attacks is characterized by several metrics, which we introduce in our notation in the remainder of this section. We use the i superscript to indicate that the formula describes an ineffective-based statistical attack (SIFA). The first relevant characteristic is the probability of an ineffective fault when the faulted intermediate variable X equals value j. Similarly, we define a second probability parameter that calculates the probability of $X = j$ if the injected fault is ineffective. We call it q_j^i and compute it using the Bayesian relation:

$$p_j^\mathsf{i} = \Pr(\mathsf{i} \mid X = j) = \Pr(X = X' = j) \tag{5}$$

$$q_j^\mathsf{i} = \Pr(X = j \mid \mathsf{i}) = \frac{\Pr(\mathsf{i} \mid X = j) \cdot \Pr(X = j)}{p^\mathsf{i}}\,.$$

Assuming that X is uniformly distributed, $\Pr(X = j) = 2^{-m}$, we can estimate q_j^i as

$$q_j^\mathsf{i} = \Pr(X = j \mid \mathsf{i}) = \frac{p_j^\mathsf{i}}{\sum p_j^\mathsf{i}}\,. \tag{6}$$

Given N ciphertexts, we assume there exist N_i ineffective ciphertexts. For a key candidate k, one can partially decrypt the ineffective ciphertexts to compute the m-bit intermediate value X. The number of times that value j is observed in the fault location (intermediate value X) for the key guess k is

$$r_j^\mathsf{i}(k) = \frac{\#\{X = j \mid \mathsf{i}, \text{key} = k\}}{N_\mathsf{i}}\,.$$

As with SFA, the attacker can recover the correct key using statistical tests such as LLR or SEI. The required N_i is inversely proportional to the capacity as we discuss in Section II-A. The probability of an ineffective event is referred to as the ineffectivity rate Π_i:

$$\Pi_\mathsf{i} = \frac{\sum_{j=0}^{2^m-1} p_j^\mathsf{i}}{2^m}\,.$$

The total number N of ciphertexts required to get N_i ineffective ciphertexts can be calculated as follows based on the ineffectivity rate:

$$N = \frac{N_\mathsf{i}}{\Pi_\mathsf{i}}\,. \tag{7}$$

C. SIFA on Masked Implementations with Fault Countermeasures

Consider a d^{th}-order masked redundant implementation that executes each operation d times and outputs only if all redundant computations are identical. Such an implementation is generally assumed to be secure against up to d fault inductions, as the redundant computations identify faults, and against up to d^{th}-order side-channel attacks thanks to the masking approach. However, it is shown in [23] that this assumption holds only for fault attacks that make use of faulty ciphertexts, and SIFA is able to invalidate the statement's generality. In a circuit with masking and detection countermeasures, a single-bit fault induction to only one out of d shares of a variable is sufficient for the attacker to attain the original (unmasked) value of another variable. The observation is not confined to any specific masking scheme. There is also no limitation for the fault model (bit flip, instruction skip, stuck-at, etc.). The only existing complication is determining which fault model is best for each scheme.

D. Countermeasures Against SIFA

The purpose of any SIFA countermeasures should be to either reduce the distribution's capacity or make the fault effective when injected into the sensitive value.

Regarding the first direction, adding noise by introducing ineffective events that do not originate from ineffective faults can help in the establishment of an unbiased distribution over ineffective events, or at least in reduction of the capacity C_{i}. For example, in the case of hiding, the attacker is prevented from easily injecting the fault into the intended intermediate value with a high success rate. Dummy rounds and shuffling are two common approaches to hiding. Other alternatives include error-correction methods, which correct specific faulty bits. As a result, the attacker is unable to distinguish between corrected and ineffective faults [24], [25]. The use of error-correction techniques in the SIFA countermeasures can be effective, but it comes with significant overhead, especially when combined with masking techniques.

The second direction was proposed in [26], where a sophisticated mechanism is introduced for circuits with simultaneous masking and detection redundancies to counteract SIFA. For this purpose, the authors recommend employing Toffoli gates [27] to construct circuits. When the sensitive intermediate variable is dependent on a complete set of shares, such a circuit ensures that faults are effective and thus detected; but when it is only dependent on an incomplete set of shares, they can be ineffective without providing exploitable information to the attacker.

III. Statistical Effective Fault Attacks (SEFA)

In this section, we first introduce the concept of Statistical Effective Fault Attacks. Then, we study its application on circuits with simultaneous side-channel and fault countermeasures.

A. SEFA

Critical information can be exploited utilizing ineffective faults, as previously discussed. We now show that effective faults, on the other side of the coin, can also lead to the retrieval of the key. We use a 2-bit random-AND fault model as an example to describe the outline of the idea of SEFA. The probability distributions for various transitions that are likely to occur due to the fault injection are represented in Table II. The red diagonal of the table denotes the transitions in which the fault has not affected the intermediate variable, i.e., an ineffective fault. The higher and lower triangles (in blue), on the other hand, indicate effective faults. We observe that the probability distribution of the intermediate variable x is not only non-uniform when restricted to the ineffective faults, but is also biased over the effective faults (see Table IIIa). SIFA uses the former; for SEFA, we want to take advantage of the latter with the same presumptions and in the same manner.

In the following, we formulate SEFA analogous to the notation presented in Section II-B for SIFA. Then, we use these formulas to evaluate two different examples: the 2-bit random-AND model, as illustrated in Table II, and its 4-bit version.

The probability of an effective fault (p_j^{e}) when $X = j$ is denoted as follows, where p_j^{i} is the probability of an ineffective fault as defined in Equation (5):

$$p_j^{\mathsf{e}} = \Pr(\mathsf{e} \mid X = j) = 1 - p_j^{\mathsf{i}} \,. \tag{8}$$

TABLE II: Fault distribution for a 2-bit random-AND model

		x'			
		00	01	10	11
x	00	1	0	0	0
	01	$\frac{1}{2}$	$\frac{1}{2}$	0	0
	10	$\frac{1}{2}$	0	$\frac{1}{2}$	0
	11	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

Conversely, the probability of $X = j$ when the injected fault is effective is then

$$q_j^{\mathsf{e}} = \Pr(X = j \mid \mathsf{e}) = \frac{p_j^{\mathsf{e}}}{\sum p_j^{\mathsf{e}}} \,. \tag{9}$$

Given N ciphertexts, we assume that there are $N_{\mathsf{e}} = N - N_{\mathsf{i}}$ ineffective ciphertexts. One can partially decrypt the ineffective ciphertexts to obtain the m-bit intermediate value X for a key candidate k. $r_j^{\mathsf{e}}(k)$ denotes how frequently the value j is observed in the fault location (intermediate value X) for the key guess k over the effective ciphertexts:

$$r_j^{\mathsf{e}}(k) = \frac{\#\{X = j \mid \mathsf{e}, \mathrm{key} = k\}}{N_{\mathsf{e}}} \,.$$

The number of necessary effective ciphertexts N_{e} for successful key-recovery can be estimated based on the capacity of the q-distribution. The total number of required ciphertexts N also depends on the probability of an effective event, the effectivity rate Π_{e}:

$$\Pi_{\mathsf{e}} = \frac{\sum_{j=0}^{2^m-1} p_j^{\mathsf{e}}}{2^m} = 1 - \Pi_{\mathsf{i}} \tag{10}$$

$$N = \frac{N_{\mathsf{e}}}{\Pi_{\mathsf{e}}} \,. \tag{11}$$

a) Example: 2-bit and 4-bit AND*:* The probability distributions of an intermediate variable under both the effective and ineffective scenarios of 2-bit random-AND and 4-bit random-AND faults are illustrated in Table III. The amount of data required for key recovery is determined by the capacity and rate, different for effective and ineffective events. The required number of ciphertexts N is proportional to $\frac{1}{C \cdot \Pi}$. For 2-bit random-AND faults, $\frac{1}{C \cdot \Pi} \approx 7.6$ for ineffective faults and ≈ 5.9 for effective faults, so we expect SEFA to require less data than SIFA. Our experiments, discussed in detail later in Section VI, confirm this expectation. All in all, depending on the application, SEFA may require less data than SIFA or vice versa since effective and ineffective faults have distinct capacities and rates.

B. SEFA on Masked Implementations with Fault Countermeasures

We are interested in investigating the performance of SEFA when applied to implementations providing masking and fault detection countermeasures.

Consider a simple masked AND gate, as found in most S-boxes. A first-order implementation of this gate with Domain-

TABLE III: Probability distribution of effective/ineffective fault with capacity C and rate Π.

(a) 2-bit random-AND model

j	00	01	10	11	C	Π
q_j^{i}	0.44	0.22	0.22	0.11	0.234	0.562
q_j^{e}	0	0.29	0.29	0.43	0.387	0.437

(b) 4-bit random-AND model

j	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	C	Π
q_j^{i}	0.20	0.10	0.10	0.05	0.10	0.05	0.05	0.02	0.10	0.05	0.05	0.02	0.05	0.02	0.02	0.01	0.52	0.32
q_j^{e}	0.00	0.05	0.05	0.07	0.05	0.07	0.07	0.08	0.05	0.07	0.07	0.08	0.07	0.08	0.08	0.09	0.11	0.68

Oriented Masking (DOM) [28] is given by

$$z_0 = x_0y_0 \oplus (x_0y_1 \oplus r)$$
$$z_1 = x_1y_1 \oplus (x_1y_0 \oplus r)\,.$$

If we examine the gate as a single masked circuit, the output shares z_0 and z_1 are joined together to produce the z output. However, they are treated as two distinct outputs when we suppose the gate is a subcircuit. A bit-flip fault induced into each of the x input shares leads to effective or ineffective faults at the outputs of this implementation (in either of the joint or distinct output cases) with a probability of $50\,\%$. This is illustrated in the bit-flip columns in the DOM Implementation segment in Table IV; the effectivity and ineffectivity cases are represented in blue and red, respectively, in the table. An attacker who fails to obtain the desired output after inducing a bit-flip fault on the x_0 share can conclude that the fault was effective as the detection system filtered it out. Then, they can translate the effectivity of the fault to the unmasked value of y input by a closer look at Table IV. Effective cases are possible only if the original value of y equals 1.

The observation is not confined to the bit-flip fault model or the DOM implementation of S-boxes. Other schemes, such as CMS [29], suffer from similar exploitable leaks if the proper fault model is utilized. We consider some further examples for SEFA applications:

1) The first-order masked AND gate implementation with the stuck-at-0 fault model,
2) Masked implementations within a broader scope, considering both bit-flip and stuck-at-0 fault models. By broader scope, we mean every realization that generates component functions x_0y_0, x_0y_1, …, and uses them as components that together are expected to construct the output in a more complicated manner.

The stuck-at-0 columns of the DOM Implementation segment and the last six columns of Table IV illustrate both the effective and ineffective cases of the first two examples. A cursory examination of the effective/ineffective cases may result in the conclusion that one cannot exploit them to retrieve the unmasked value of y input with the same certainty as to the preliminary example. In other words, they cannot provide us with a definitive answer concerning the y input. However, that is not all, and both SEFA and SIFA still work. They can be used to recover the original value of the y input, although not at first glance, but by employing some computations. We will give some numerical criteria later in Section IV-C to assess and compare SEFA and SIFA in the mentioned examples. Our comparison methodology works for all masked implementations with fault detection countermeasures in general. We leave the impact of inducing multiple faults per execution for future research, as did the authors of [23].

TABLE IV: Revealing the unmasked value of y input by inducing stuck-at-0 (S0) or bit-flip (BF) faults on input share x_0 in a masked AND-gate for two types of masking.

					DOM Implementation							General masked implementations					
x_0	x_0' S0	x_0' BF	y_0	y_1	z_0	z_0' S0	z_0' BF	z_1	z	z' S0	z' BF	x_0y_0	x_0y_1	$x_0'y_0$ S0	$x_0'y_1$ S0	$x_0'y_0$ BF	$x_0'y_1$ BF
0	0	1	0	0	r	r	r	r	0	0	0	0	0	0	0	0	0
0	0	1	0	1	r	r	$\bar{r}$	$x_1 \oplus r$	x_1	x_1	$\bar{x}_1$	0	0	0	0	0	1
0	0	1	1	0	r	r	$\bar{r}$	$x_1 \oplus r$	x_1	x_1	$\bar{x}_1$	0	0	0	0	1	0
0	0	1	1	1	r	r	r	r	0	0	0	0	0	0	0	1	1
1	0	0	0	0	r	r	r	r	0	0	0	0	0	0	0	0	0
1	0	0	0	1	$\bar{r}$	r	r	$x_1 \oplus r$	$\bar{x}_1$	x_1	x_1	0	1	0	0	0	0
1	0	0	1	0	$\bar{r}$	r	r	$x_1 \oplus r$	$\bar{x}_1$	x_1	x_1	1	0	0	0	0	0
1	0	0	1	1	r	r	r	r	0	0	0	1	1	0	0	0	0
						distinct outputs				joint output							

IV. SEFA vs. SIFA

Comparing the performance of SIFA and SEFA entails assessing the adversary model and the amount of data required. To begin, we will explore how the attack model is dependent on the countermeasure employed. We will then inspect various parameters that affect the data required for SIFA and SEFA in practice.

A. Attack Model in Detection- and Infection-based Countermeasures

To mount SIFA or SEFA, the attacker must possess ineffective ciphertexts and non-faulty effective ciphertexts, respectively. Various scenarios for acquiring the relevant data may be considered depending on the countermeasures used in the implementation. Different fault attacks, such as SFA and DFA, are applicable to unprotected implementations. As a result, we concentrate on protected implementations that employ detection- or infection-based countermeasures, where classical fault assaults are inapplicable.

In the presence of infection-based countermeasures, the attacker is unable to determine ineffective ciphertexts directly. SIFA can be mounted in one of two ways. First, the attacker

can determine which ciphertexts are ineffective by encrypting specific inputs in the chosen-plaintext model using both normal and faulty encryption. As a result, the required data is multiplied by two, and input control is assumed to be in the attacker's hands. Alternatively, the attacker can perform SIFA on all available faulty ciphertexts. This attack has the advantage of remaining within the known-ciphertext model. On the other hand, the set of ineffective ciphertexts is not distinguishable. As a result, the amount of data required significantly increases because unknown effective ciphertexts act as noise during the key recovery process. The attacker must execute the encryption twice to obtain non-faulty effective ciphertexts for mounting the SEFA: once to compute the correct ciphertexts and once to compute the faulty ciphertext. This fact implies that the required data doubles, and the attack is defined in the chosen-plaintext scenario.

If a detection-based countermeasure is used, it is possible to obtain ineffective ciphertexts from the faulted encryption. As a result, SIFA can be performed without incurring additional overhead in the known ciphertext scenario. In the presence of a detection-based countermeasure, there are two approaches to perform SEFA. The first is a chosen-plaintext technique, as previously discussed for infection-based countermeasures. However, achieving both correct and faulty encryption doubles the amount of required data. Alternatively, the attacker can target an intermediate value of the first rounds. Thus, SEFA can be mounted in the known-plaintext scenario without requiring additional data. It is worth mentioning that SEFA might also be performed in a ciphertext-only scenario, if the adversary can determine whether the injected fault was effective or not. For example, if the attacker can identify the tag verification procedure in an authenticated encryption system, it is possible to follow the strategy described by [30] but for SEFA. Any fault injection for which the tag has been checked is deemed an ineffective fault injection in this situation, and vice versa. As a result, in a ciphertext-only scenario, the adversary can obtain the necessary data to carry out the attack.

B. Effect of Undesired Ineffective Events

Apart from ineffective faults, there are several other possible causes of ineffective ciphertexts. The attacker cannot distinguish between ineffective faults and these events, as neither modifies the ciphertexts. As a result of these occurrences, the performance of SIFA and even SEFA may be impacted. In this part, we examine the consequences of these occurrences.

1) Missed Faults: In practice, despite the attacker's efforts, the injected fault may not be successful. When missed faults occur, the observable ineffective rate is higher than expected, whereas the observable effective rate is lower than expected. The observable ineffective and effective rates are proportional to the probability of missed fault injection $\Pi_{\text{miss}} = \Pr[\text{missed fault}]$. $\Pi_{\text{i,miss}}$ and $\Pi_{\text{e,miss}}$ denote ineffective and effective rates, respectively:

$$\Pi_{\text{i,miss}} = \Pi_{\text{miss}} + (1 - \Pi_{\text{miss}})\Pi_{\text{i}}$$
$$\Pi_{\text{e,miss}} = (1 - \Pi_{\text{miss}})\Pi_{\text{e}} .$$

When faults are missed, the probability of $X = j$ over ineffective ciphertexts is

$$q_j^{\text{i,miss}} = (1 - \Pi_{\text{miss}})q_j^{\text{i}} + 2^{-m}\Pi_{\text{miss}} .$$

The missed fault rate varies depending on the attack's setup and equipment. The probability distribution of the target intermediate value over ineffective ciphertexts approaches uniform distribution as the probability of missing a fault increases.

In comparison, missed faults do not affect on the probability distribution of the target intermediate value for effective ciphertexts. However, SEFA requires slightly more data to get enough effective ciphertexts, as the effective rate $\Pi_{\text{e,miss}}$ decreases with missed faults.

2) Error Correction: Assume that an error-correction code-based countermeasure is used to protect the implementation against fault attacks. Consider a simple correcting code such as [31] that detects d bit faults and corrects $d' = \frac{d}{2}$ bits faults. SIFA and SEFA can both be applied by injecting $d'+1$ biased bit faults. However, restoring d' faulty bits increases the number of observable ineffective ciphertexts. In other words, not all ineffective ciphertexts result from ineffective faults, but error correction may also cause ineffective ciphertexts. However, it is impossible to distinguish between corrected errors and ineffective faults. The probability of a fault being ineffective or corrected when the faulty intermediate variable X equals the value j increases, where HW denotes the Hamming weight:

$$\begin{aligned} p_j^{\text{i,cor}} &= \Pr(\text{i} \mid X = j) + \Pr(\text{cor} \mid X = j) \\ &= \Pr(X = j, \text{HW}(X \oplus X') \leq d')r . \end{aligned}$$

The probability of $X = j$ for a correct ciphertext approaches the uniform distribution:

$$q_j^{\text{i,cor}} = \Pr(X = j \mid \text{i}, \text{cor}) = \frac{p_j^{\text{i,cor}}}{\sum p_j^{\text{i,cor}}} .$$

This fact significantly increases the amount of data required for SIFA, as it results in a less biased probability distribution for SIFA's target.

By contrast, the probability of a fault being effective when the faulty intermediate variable X equals the value j decreases in this scenario:

$$\begin{aligned} p_j^{\text{e,cor}} &= \Pr(\text{e} \mid X = j) + \Pr(\text{cor} \mid X = j) \\ &= \Pr(X = j, \text{HW}(X \oplus X') > d') . \end{aligned}$$

Thus, the probability distribution of effective faults in SEFA becomes more biased:

$$q_j^{\text{e,cor}} = \Pr(X = j \mid \text{e}, \text{cor}) = \frac{p_j^{\text{e,cor}}}{\sum p_j^{\text{e,cor}}} .$$

a) Example: To demonstrate this point, consider a 2-bit random-AND fault corrected using the 1-bit correction method. Table V illustrates the effective and ineffective probability distributions for this scenario. The attacker can retrieve the intermediate value with probability one since $11 \rightarrow 00$ is the only possible effective event.

TABLE V: Fault distribution for a 2-bit random-AND model with the assumption of 1-bit correction. Red: corrected and ineffective events. Blue: effective events.

x \ x'	00	01	10	11
00	1	0	0	0
01	$\frac{1}{2}$	$\frac{1}{2}$	0	0
10	$\frac{1}{2}$	0	$\frac{1}{2}$	0
11	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

However, increasing effective capacity does not always imply that SEFA can be enhanced by implementing error-correction mechanisms as the effective rate decreases. Nevertheless, SEFA can significantly outperform SIFA when an error-correction technique is used, as we will demonstrate in Section VI-C.

b) Majority Voting: As already discussed in [11], error correction can also be implemented on the cipher level. For example, consider an implementation that performs two additional redundant computations and only releases an output if it was produced by at least two of the redundant computations. This can hamper both SIFA and SEFA since the attacker, performing only one fault injection during the entire computation, cannot observe a different behavior between effective/ineffective fault injections anymore. In such a case, the attacker can adopt the attack strategy by using an additional (possibly random) fault somewhere within one of the additional computations such that the criteria if the output can be observed again relies on the effect of the first fault injection.

3) Hiding: Typically, fault attacks use a biased distribution of intermediate values. By limiting the attacker's knowledge of the computed values at various points during the execution, the hiding countermeasures make it more difficult for an attacker to inject faults into a specific intermediate value. The ideal goal of preventing fault attacks through the use of hiding countermeasures appears challenging to achieve. However, as the amount of data required for fault attacks grows, hiding countermeasures can provide additional security.

We now discuss two common strategies for hiding countermeasures and their implications for SIFA and SEFA. The first method involves inserting dummy rounds between the real rounds at random. The second method is shuffling, which randomly orders the operations.

a) Dummy Rounds: The basic idea behind this technique is to perform ineffective dummy rounds preceding and following real rounds at random. As a result, the position of each round varies from execution to execution. Injecting a fault into the dummy rounds always results in ineffective ciphertexts that are indistinguishable from those produced by the target round's ineffective fault. Similar to [23], we consider a protected AES implementation in which each execution includes 10 real AES rounds and $10(k-1)$ ineffective dummy rounds. As demonstrated in [23], this protection increases the amount of data required for the SIFA by a factor of $6.5 \cdot k^2$.

TABLE VI: Approximation data complexity of SEFA and SIFA with masking countermeasure.

Implementation	Fault model	SEFA Π_e	SEFA C_e	SEFA $\frac{1}{C_e \cdot \Pi_e}$	SIFA Π_i	SIFA C_i	SIFA $\frac{1}{C_i \cdot \Pi_i}$
DOM-AND gate	Bit-flip	$\frac{1}{2}$	1	2	$\frac{1}{2}$	1	2
	Stuck-at-0	$\frac{1}{4}$	1	4	$\frac{3}{4}$	$\frac{1}{9}$	12
	Random-AND	$\frac{1}{8}$	1	8	$\frac{7}{8}$	$\frac{1}{49}$	56
General masked implementations	Bit-flip	$\frac{3}{4}$	$\frac{1}{9}$	12	$\frac{1}{4}$	1	4
	Stuck-at-0	$\frac{3}{8}$	$\frac{1}{9}$	24	$\frac{5}{8}$	$\frac{4}{100}$	40
	Random-AND	$\frac{3}{16}$	$\frac{1}{9}$	48	$\frac{13}{16}$	$\frac{1}{169}$	208

Dummy rounds do not induce any noise in the probability distribution of the intermediate value over the non-faulty effective ciphertexts. Consequently, in the case of SEFA, this countermeasure has no effect on capacity. The dummy rounds only reduce the effective rate by a factor of at most k, resulting in an increase in data complexity by a factor of k. As a result, dummy rounds are significantly less efficient against SEFA than they are against SIFA.

b) Shuffling: As with dummy rounds, shuffling is intended to make it more difficult for an attacker to inject the desired fault into a precise intermediate value. Ineffective ciphertexts can be generated due to an untargeted intermediate value being injected with an ineffective fault. Similarly, the effective fault on non-target intermediate values can generate effective ciphertexts. Due to this impact, this approach is equally efficient against SIFA and SEFA.

C. Masking

As discussed in Section III-B, both SEFA and SIFA can be mounted on implementations that leverage masking and detection as their underlying defenses against power analysis and fault attacks. We also showed through various cases that depending on the masking strategy and the fault model, SEFA and SIFA can yield similar or better outcomes. A reasonable benchmark to compare their efficiency would be an approximation of their data complexity using the $\frac{1}{C \cdot \Pi}$ parameter. Hence, in Table VI, we provide the capacity C, rate Π, and $\frac{1}{C \cdot \Pi}$ values for each case studied in Section III-B. The results show that when the fault model is stuck-at-0 or random-AND, SEFA has a much lower data complexity for each of the examined masking strategies. This superiority is also valid for the stuck-at-1 case; we give no details because the analysis is very similar. On the other hand, when the DOM approach is employed as the underlying masking in the bit-flip fault model, the two attacks have the same data complexities, while SIFA outperforms when we move to a more general masked implementation. We also present experimental results for concrete instances in Section VI.

D. Inapplicable Cases

So far, we have explored the events and countermeasures that, while not preventing SIFA, make it more difficult to perform in practice. However, there are specific circumstances in which SIFA cannot be applied even if the adversary has access to an unlimited amount of data. A natural question arises regarding the applicability of SEFA in these instances.

In this part, we analyze these scenarios in detail below and explain that if SIFA is not applicable, SEFA will not work as well. Using the same argument, one can demonstrate that if SEFA is not applicable, SIFA will also be inapplicable.

1) Unbiased Ineffective Faults: Let us assume that the probability distribution of a faulty value is uniform over all ineffective ciphertexts, i.e., $q^{\mathsf{i}}_j = 2^{-m}$ for all $0 \leq j \leq 2^m - 1$. It is possible to demonstrate that in this case, the probability distribution of the same intermediate value over effective events is uniform as well. The probability of $X = j$ in case of effective injection can be determined using Equation (9). Since $p^{\mathsf{e}}_j = \Pr(\mathsf{e} \mid X = j) = 1 - p^{\mathsf{i}}_j$ (refer to Equation (5)), and $q^{\mathsf{i}}_j = 2^{-m}$ implies $\sum_{j'=0}^{2^m-1} p^{\mathsf{i}}_{j'} = 2^m p^{\mathsf{i}}_j$ (see Equation (6)), we get

$$q^{\mathsf{e}}_j = \frac{1 - p^{\mathsf{i}}_j}{\sum_{j'=0}^{2^m-1}(1 - p^{\mathsf{i}}_{j'})} = \frac{1 - p^{\mathsf{i}}_j}{2^m - \sum_{j'=0}^{2^m-1} p^{\mathsf{i}}_{j'}} = \frac{1 - p^{\mathsf{i}}_j}{2^m - 2^m p^{\mathsf{i}}_j} = 2^{-m}\,.$$

Thus, the probability distribution over effective ciphertexts is also uniform. As a consequence, if SIFA is inapplicable due to an unbiased ineffective fault, SEFA will also be inapplicable.

2) Zero Ineffective Rate: Let us assume that the ineffective rate Π_{i} for the injected fault over an intermediate value is zero. In other words, $q^{\mathsf{i}}_j = 0$ for all $0 \leq j \leq 2^m - 1$, or equivalently $p^{\mathsf{i}}_j = 0$. Naturally, SIFA cannot be successful in this case. The probability distribution of the intermediate value over the effective ciphertexts should be taken into account to determine the applicability of SEFA. Using Equation (8), it is possible to deduce that $p^{\mathsf{e}}_j = 1$ for all $0 \leq j \leq 2^m - 1$. Thus, the probability distribution of the intermediate value over the effective ciphertexts is unbiased; as a result, neither SEFA nor SIFA are applicable in this scenario:

$$q^{\mathsf{e}}_j = \frac{p^{\mathsf{e}}_j}{\sum p^{\mathsf{e}}_j} = \frac{p_j}{2^m p_j} = 2^{-m}\,.$$

3) Reversible Computations: A recent paper in CHES 2020 [26] describes how to combine a typical combination of masking and redundant computations with principles from reversible computing to build SIFA-resistant S-box implementations. This construction is efficient against SIFA since a fault based on the intermediate value either (1) is effective (with $\Pi_{\mathsf{i}} = 0$) or (2) becomes ineffective but only depending on an incomplete set of shares. This method should also help against SEFA since the fault is either (1) effective but with uniform distribution over the faulty value, or (2) ineffective but an insufficient number of shares are dependent on the faulty value.

E. SEFA's Applicability in Comparison to Other Fault Attacks

In summary, compared to other types of fault attacks and specifically to SIFA, SEFA has the following advantages and disadvantages:

a) Disadvantages of SEFA:

- Compared to SIFA and SFA, one downside of SEFA is the demand for input control which is, for example, not typically available in the case of nonce-based encryption. However, SEFA is applicable in authenticated decryption if the attacker also has access to a decryption device that essentially signals effective/ineffective faults via the successful/unsuccessful tag verification.
- For an infection-based countermeasure, SEFA requires both correct and faulty encryption of the same plaintext: the number of necessary encryptions is doubled.
- As with other fault attacks, SEFA is also somewhat impacted by misplaced faults caused by shuffling countermeasures or fault inductions with an unintended effect.

b) Advantages of SEFA:

- Like SIFA, SEFA can bypass typical detection- and infection-based countermeasures. Similarly, SEFA is often applicable to implementations that include both masking and redundancy-based fault countermeasures.
- Compared to SIFA, SEFA is much less affected by noise from missed faults. Error-correction approaches against SEFA are significantly less effective than against SIFA. Dummy rounds are substantially less effective against SEFA than against SIFA.
- The data required for both SEFA and SIFA can be obtained concurrently. This enables us to propose our second technique, SHFA, a combination of SEFA and SIFA that usually outperforms either of the techniques alone. We introduce SHFA next.

V. Statistical Hybrid Fault Attacks (SHFA)

In this section, we define the Statistical Hybrid Fault Attack (SHFA) by introducing an appropriate statistical model to take benefit of both SIFA and SEFA and all available ciphertexts simultaneously.

The sets of effective and ineffective ciphertexts are denoted by $\mathcal{E}$ and $\mathcal{I}$, respectively. Clearly, $\mathcal{E}$ and $\mathcal{I}$ may be used to determine the distribution of SEFA and SIFA, respectively. The ciphertexts that yield the SEFA and SIFA distributions, on the other hand, are disjoint, i.e., $\mathcal{E} \cap \mathcal{I} = \emptyset$. As a result, the outcome distributions, q^{e}_j and q^{i}_j, will be independent.

Therefore, the joint distribution of SEFA and SIFA, can be determined easily and used as a score statistic in SHFA to rank the key candidates. More precisely, since $\mathcal{E} \cap \mathcal{I} = \emptyset$:

$$q^{\mathsf{h}}(x, y) = \Pr(X = x \mid \mathsf{e}, Y = y \mid \mathsf{i}) = q^{\mathsf{e}}_x \cdot q^{\mathsf{i}}_y\,.$$

Given the $q^{\mathsf{h}}(x, y)$ and following the defined statistical scoring functions in Section II-A, we adapt Equation (1) to define $\mathrm{LLR}^{\mathsf{h}}$ as follows:

$$\mathrm{LLR}^{\mathsf{h}}(k) = N \sum_{x=0}^{2^m-1} \sum_{y=0}^{2^m-1} \hat{q}^{\mathsf{e}}_x \cdot \hat{q}^{\mathsf{i}}_y \cdot \log_2 \frac{q^{\mathsf{e}}_x \cdot q^{\mathsf{i}}_y}{\theta_{x,y}}\,.$$

Similarly, we adapt Equation (2) to define $\mathrm{SEI}^{\mathsf{h}}$ as follows:

$$\mathrm{SEI}^{\mathsf{h}}(k) = \sum_{x=0}^{2^m-1} \sum_{y=0}^{2^m-1} (\hat{q}^{\mathsf{e}}_x \cdot \hat{q}^{\mathsf{i}}_x - \theta_{x,y})^2\,.$$

Given q^{h}, its capacity is also defined as follows:

$$C(q^{\mathsf{h}}, \theta^{\mathsf{h}}) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \frac{(q_x^{\mathsf{e}} \cdot q_y^{\mathsf{i}} - \theta_{x,y})^2}{\theta_{x,y}} .$$

For an instant, assuming that q^{i} is uniformly distributed, then:

$$\begin{aligned} C(q^{\mathsf{h}}, \theta^{\mathsf{h}}) &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \frac{(q_x^{\mathsf{e}} \cdot \theta_y - \theta_x \cdot \theta_y)^2}{\theta_x \cdot \theta_y} \\ &= \sum_{x \in \mathcal{X}} \frac{(q_x^{\mathsf{e}} - \theta_x)^2}{\theta_x} = C(q^{\mathsf{e}}, \theta_{\mathsf{e}}) . \end{aligned}$$

When q^{e} is uniformly distributed, a similar argument may be made, demonstrating that SHFA is at least as good as the best of SEFA and SIFA.

It is also possible to follow Equations (3) and (4) respectively to determine $N_{\mathrm{LLR}}^{\mathsf{h}}$ and $N_{\mathrm{SEI}}^{\mathsf{h}}$ for the success probability of getting an a-bit advantage, that are the number of ciphertexts required to obtain the gain $P_{\mathrm{S}} = \mathbb{P}(\Delta_a > 0)$ for LLR and SEI respectively.

It is worth noting that, in addition to the joint distribution, we investigated alternative options, including $\mathrm{SEI}^{\mathsf{i}} + \mathrm{SEI}^{\mathsf{e}}$ and $\mathrm{SEI}^{\mathsf{i}} \times \mathrm{SEI}^{\mathsf{e}}$. In most circumstances, though, joint distribution surpasses them. As a result, we used it in this research.

VI. Results

To compare SIFA, SEFA, and SHFA experimentally, we performed these attacks in various circumstances, taking into account the noise associated with the attack's setup and countermeasures. The simulations and experimental testing that we employed are archived in the GitHub repository[1]. We would also like to mention that all percentage comparisons of this section are based on the required number of ciphertexts and are taken from the stated figures.

A. Noise-Free Setup

We first consider a scenario in which the fault can be injected with a success probability of 1 and no missing fault events occur. Our target is AES. At the start of the last round, we inject 2-bit and 4-bit random-AND faults into the least significant bits of the first byte of AES. By guessing a portion of the last subkey, the ciphertexts are decrypted in order to obtain the desired intermediate value. The key candidates are then ranked using the SEI statistic. We repeated the experiment with 100 randomly generated keys and calculated the average rank of the correct key based on the number of ciphertexts available. The output of our simulations is depicted in Figure 1. SEFA and SIFA outperform each other in the presence of 2- and 4-bit faults, respectively, whereas SHFA consistently outperforms both.

B. Noisy Setup with Possible Missed Faults

To investigate the effect of missed faults in practice, we simulate a fault attack on AES with a 4-bit random-AND fault, identical to the one described in Section VI-A, considering different missed fault rates of 10%, 25%, 35%, and 90%. It means that faults are not injected into the device for these ratios of encryptions. The obtained results are reflected in Figure 2 and Table VII. As the missed fault rate increases, SEFA significantly outperforms SIFA. For the 10% missed rate, the superiority is roughly 16% in favor of SIFA. Then, the tide turns already for a relatively low fault miss rate, with 17%, 50%, and 100% in favor of SEFA for 25%, 35%, and 90% missed rates, respectively. This outcome is hardly surprising, as explained in Section IV-B.1: the miss rate has a much smaller effect on SEFA. On the other hand, SHFA is superior to both SIFA and SEFA according to the same expectation, as it exploits information from both effective and ineffective leaked information. Compared to SEFA, the required number of ciphertexts for SHFA is reduced by about 37%, 20%, and 10% for 10%, 25%, and 35% missed rates, respectively (they require the same number of ciphertexts for 90% missed rate). When compared to SIFA, this reduction is about 25%, 33%, and 55% for 10%, 25%, and 35% missed rates, respectively (the rank of the correct key for 90% missed rate is around 100 in 5000 ciphertexts in the SIFA case, while in the SHFA case it converges to the first rank in 3200 ciphertexts).

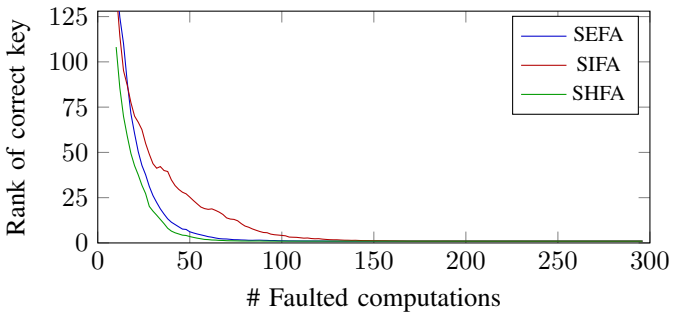
C. Protected Implementation with Error Correction

As in previous tests, we use AES as the target cipher. However, in this test, AES is protected by error-correction mechanisms capable of detecting d error bits and correcting $d' = d/2$ of them. The results for $d = 2$ and $d = 4$ are illustrated in Figure 3 and also reported in Table VII. They nicely illustrate that SEFA outperforms SIFA as the number of recovered faulty bits increases. This observation verifies the explanations presented in Section IV-B.2 that SEFA acts more effectively in the presence of error-correction techniques since the probability distribution of effective fault becomes more biased. Besides, it can be observed from the results that SHFA cannot considerably improve SEFA in this scenario, as information leakage via ineffective events is negligible. For 1-bit correction, SEFA acts 11.6% better than SIFA, and SHFA works 43.4% and 35.8% better than SIFA and SEFA, respectively. When it comes to 2-bit correction, SIFA does not yield a low-ranked key even in 3000 ciphertexts. SEFA and SHFA, on the other hand, produce nearly identical outcomes, with SHFA showing only a 7.7% reduction.

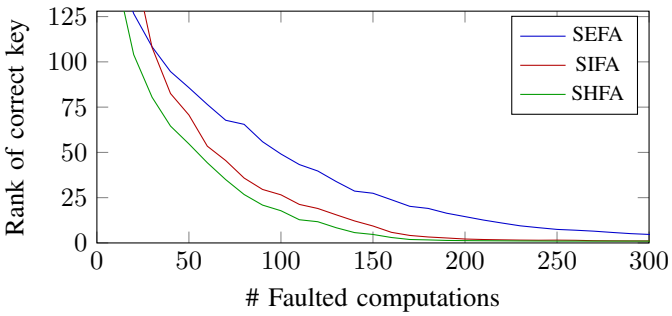
D. Protected Implementation with Dummy Rounds

Assuming $10(k-1)$ ineffective dummy rounds are performed at random during the encryption process, Figure 4 represents the effect of this hiding countermeasure against SIFA, SEFA, and SHFA. We expect SEFA to be less affected by this technique, based on what was explained in Section IV-B.3. This assertion has come true for our two examinations where $k = 2$ and $k = 5$ (see also Table VII). When comparing SEFA to SIFA in the $k = 2$ situation, there is a 59% reduction in needed ciphertexts. With 5000 ciphertexts, the rank of the correct key in SIFA does not converge to low positions for $k = 5$; nevertheless, in SEFA, it converges in roughly 1700 ciphertexts. For the same reason as Section VI-C, SHFA

[1] https://github.com/Navidvafaei/SEFA

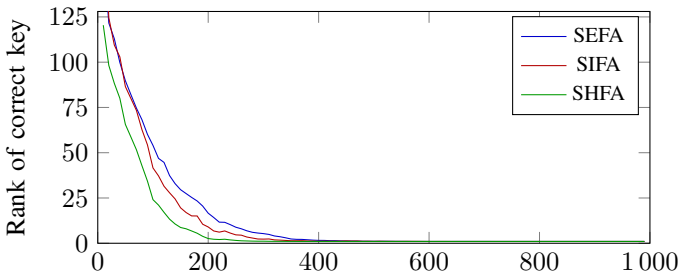


(a) SEI, 2-bit random-AND fault

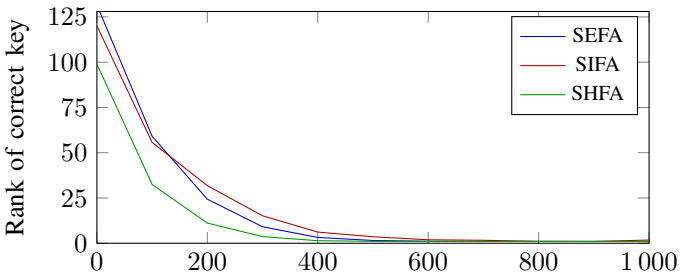


(b) SEI, 4-bit random-AND fault

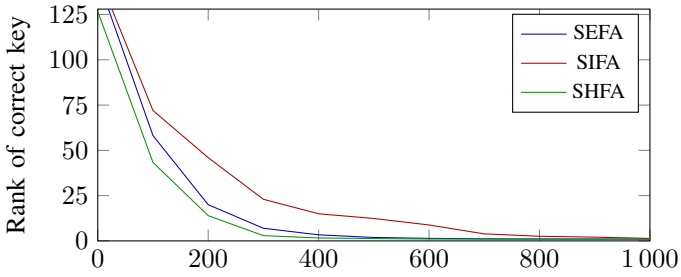
Fig. 1: The rank of the correct key in deterministic fault injection model on unprotected AES, for SIFA, SEFA, and SHFA, using SEI as the statistical scoring function.



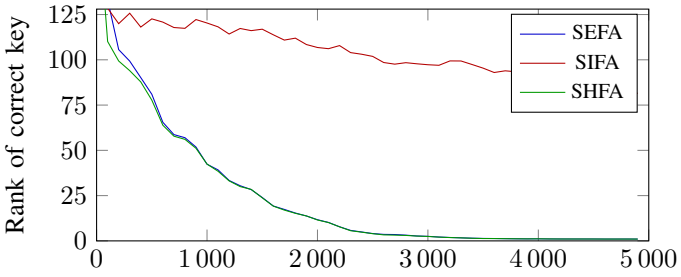
(a) Missed faults, 10 % miss rate



(b) Missed faults, 25 % miss rate



(c) Missed faults, 35 % miss rate



(d) Missed faults, 90 % miss rate

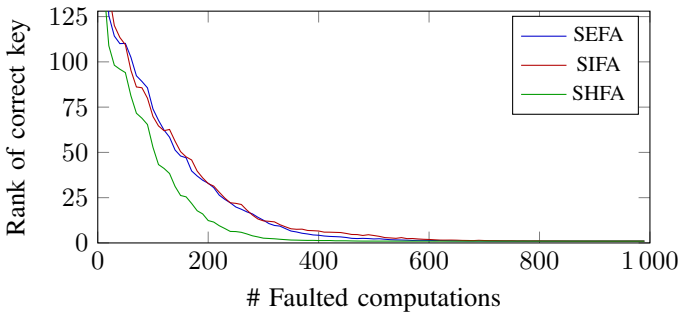
Fig. 2: The rank of the correct key in the non-deterministic fault injections model on unprotected AES, for SIFA, SEFA, and SHFA, using SEI as the statistical scoring function.

does not enhance the results of SEFA significantly as the k parameter grows. In our experiments, SHFA reduces the number of ciphertext by 38% versus SEFA (75% versus SIFA) in the $k = 2$ case. In $k = 5$, SHFA and SEFA demand similar amounts.

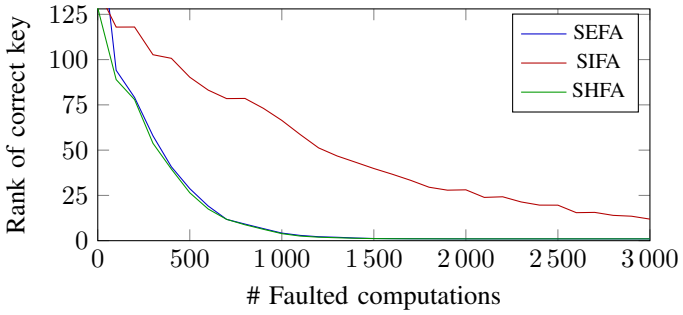
E. Implementations with Combined Countermeasures

In this section, we evaluate the performance of SIFA/SEFA/SHFA against implementations featuring combined countermeasures against fault attacks and power analysis. First, we perform a comprehensive analysis of simulated attacks against a bit-sliced, masked, redundant implementation of AES-128. We then describe practical attacks on different variants of the Keccak S-box with/without implementation-level protection against SIFA from [26] and with/without noise in the fault injection setup.

a) Emulated Attacks on AES: We first consider emulated fault injections on an assembler-optimized, masked AES implementation for the 32-bit Cortex-M4 platform from [32]. This implementation can encrypt two 128-bit inputs per block cipher call in CTR mode and is fully unrolled. For our purposes, and similarly as done in [23], we add temporal redundancy but only encrypt one 128-bit input per block cipher call in ECB mode to keep the evaluation scenario as simple as possible. In our experiments, we consider faults that flip a single bit in the output of one specific instruction during the execution of the S-box layer in round 9 of the encryption operation (which is the usual location when mounting SIFA on AES-128 [11]). We then call the AES encryption operation 1000/2000/3000/4000 times and collect the received outputs. We use correct outputs for evaluation of SIFA, correct re-encryptions of inputs that resulted in faulty outputs for the evaluation of SEFA, or both faulty and correct outputs for the evaluation of SHFA. Hereby, we consider an attack to be successful if the rank of the correct key is the lowest after having observed a certain amount of faulted computations.

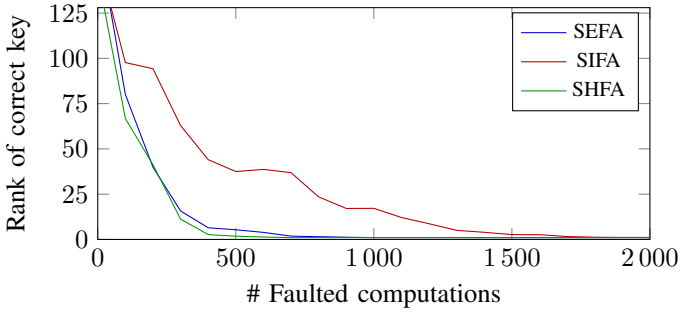


(a) 1-bit error correction

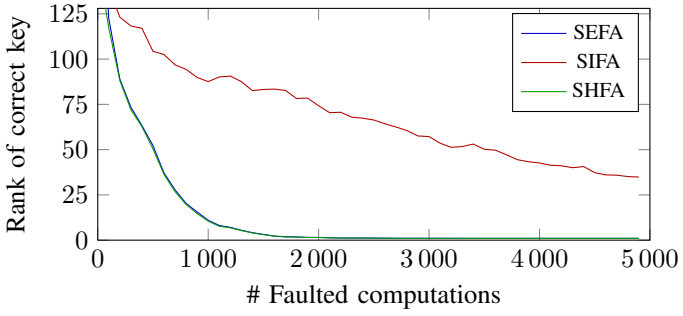


(b) 2-bit error correction

Fig. 3: The rank of the correct key in the fault injections model on protected AES with error correction for SIFA, SEFA, and SHFA, using SEI as the statistical scoring function.



(a) Protected by dummy rounds when $k = 2$



(b) Protected by dummy rounds when $k = 5$

Fig. 4: The rank of the correct key in the fault injections model on protected AES with dummy rounds for SIFA, SEFA, and SHFA, using SEI as the statistical scoring function.

This experiment is repeated for all 688 instructions that make up the masked S-box computation on the entire state in round 9. Hence, we can see which type of attack is successful after a certain amount of faulted encryptions and at which fault locations.

The results of our analysis are presented in Figure 5. Each vertical line indicates if an attack is successful when a bit-flip fault is injected in the corresponding instruction of the S-box computation. We additionally color code the strength of the resulting bias by using a dark color if the attack was already successful after observing 1000 faulted computations, a light color if the attack was only successful after 4000 faulted computations, and two steps in between. Conversely, a white line indicates that a bit-flip fault at the corresponding instruction does not lead to a successful attack after 4000 faulted computations.

Overall, we can see that, when considering this concrete experiment, SIFA seems to be applicable in more locations than SEFA after having performed at most 4000 faulted computations. While we have shown before that SEFA is applicable whenever SIFA is applicable, the resulting bias, and thus the necessary amount of observed faulted computations, can still be different for both attacks. We have confirmed this theory by observing that SEFA does sometimes require more than 100 000 faulted computations to be successful in this specific experiment. Besides that, we want to note that the observed biases of SEFA seem to be generally more robust than in the case of SIFA if fault injections in the latter half of the computations are considered. In any case, SHFA works well in both halves of the computation when compared to performing SIFA and SEFA individually.

b) Practical Attacks: For our practical evaluation, we target Keccak S-Box implementations using a traditional combination of DOM masking with redundancy and the SIFA-hardened Keccak S-Box implementation from [26]. The practical evaluation of our fault attack was done on an 8-bit Xmega 128D4 microprocessor using clock glitches on a Chipwhisperer-Lite SCA evaluation board [33]. The choice of using an 8-bit platform for practical attacks is mostly motivated by the fact that we do not have easy access to a fault setup suitable for 32-bit devices. The word size of the target implementation can have an impact on the attack performance, however, this strongly depends on the used fault injection method. When considering simple clock glitches that corrupt the result of one computation, smaller word sizes help by limiting the effect of a fault to only 8-bit of an intermediate value which gives very usable ratios between effective and ineffective faults. However, a more localized fault injection method based on e.g. Lasers or BBI can achieve similar localized fault effects, no matter the processor word size.

In the attacks, we insert an additional fast clock edge onto the device's original clock signal to violate a critical path guarantee and force a temporary undefined behavior of the microprocessor. Similarly as before, we evaluate the performance of SIFA, SEFA, and SHFA by performing 1000/2000/3000/4000 faulted S-box computations. The Keccak S-box can be used in schemes with different state sizes. This fact influences the number of key bits that need to be guessed at once [6]. Therefore, we now consider an attack to be successful if the observed SEI of an input byte of the

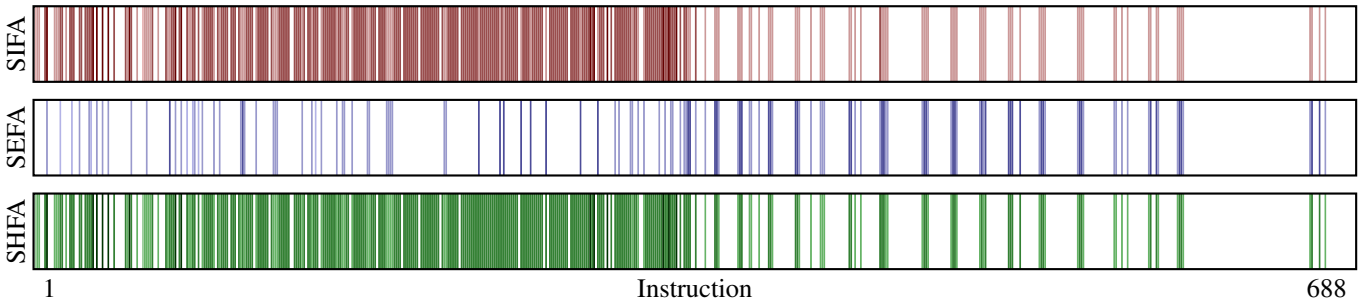


Fig. 5: Evaluation of SIFA/SEFA/SHFA using an ARM assembler optimized masked AES implementation with temporal redundancy. Vertical lines indicate if injecting a bit-flip fault into the corresponding instruction of the AES S-box computation in round 9 leads to a successful attack. Darker/lighter colors indicate stronger/weaker observed biases, while white lines indicate that an attack was unsuccessful after 4000 faulted computations.

S-box is higher than the maximum SEI that one would expect to observe when drawing the same amount of samples from a uniform distribution 2^{32} times. This is equivalent to checking if the observed bias is stronger than the highest bias of testing 2^{32} wrong key guesses during key recovery.

As a first experiment, we performed attacks on a DOM masked Keccak S-box implementation with redundancy using reliable clock glitches that corrupt one value (byte) of the computation at a specified time. The results are depicted in Figure 6a and essentially show that SIFA, SEFA, and SHFA perform about the same. If we, however, modify the parameter of the injected glitch (in terms of width and offset during the targeted clock cycle) such that it only affects the executed instruction about 90% of the time, thereby simulating the effect of imperfect fault injection setups, we can see that the performance of SIFA is impacted more than for SEFA and SHFA (Figure 6b). Finally, we also test the SIFA-hardened Keccak S-box implementation from [26] and confirm that neither attack is applicable, even if perfectly reliable fault injections are used (Figure 6c).

F. Discussion

The results of our experiments are summarized in Table VII. In general, SEFA performs better than SIFA when undesired ineffective events occur due to setup noise or the countermeasures used. Considering Equations (7) and (11), it seems surprising that SEFA (SIFA) is superior when the effective (ineffective) rate is less. It is worth noting that the rate is not the only aspect that affects the complexity of the data; capacity also plays a central role. As a rule of thumb, SIFA and SEFA require around $O(1/C_{\mathsf{i}}\Pi_{\mathsf{i}})$ and $O(1/C_{\mathsf{e}}\Pi_{\mathsf{e}})$ data. In most situations, we observed a substantial increase in the effective capacity C_{e} when the effective rate Π_{e} decreases. Since capacity usually varies substantially more than the rate, capacity has a greater impact on data complexity. As a result, SEFA requires less data in these instances. The relationship between SIFA and ineffective rata is analogous. As the effective rate and ineffective rate decrease, SEFA and SIFA perform better, respectively. When the results of SIFA and SEFA are comparable, SHFA surpasses both attacks. In other instances when SIFA (SEFA) outperforms SEFA (SIFA), the performance of SHFA is similar to SIFA (SEFA).

TABLE VII: Required number of faulted ciphertexts to reduce the correct key's rank below 2.

	random-AND		Miss rate				Dummy rounds		Error-Correction	
	2 bits SEI	4 bits SEI	10%	25%	35%	90%	$k=2$	$k=5$	$d'=1$	$d'=2$
SIFA	118	232	320	600	1000	>5000	1600	>5000	600	>3000
SEFA	68	292	380	500	500	3200	650	1700	530	1300
SHFA	60	168	240	400	450	3200	400	1700	340	1200

VII. Conclusion

We proposed Statistical Effective Fault Attacks (SEFA), a novel fault attack strategy that shares many of the advantages of SIFA, such as the power to bypass many combined countermeasure techniques or applicability to AEAD modes. In many circumstances, SEFA is more efficient than SIFA, for example when fault setups are not perfectly reliable or under countermeasures such as error-correction or dummy rounds. We also proposed Statistical Hybrid Fault Attacks (SHFA), which combine the advantages of SIFA and SEFA by collecting data for both attack strategies concurrently, thus yielding the best results in most scenarios. We discuss in detail how SIFA, SEFA, and SHFA perform in various scenarios, including combined countermeasures and noisy setups, both by means of simulations and real experiments on implementations of Keccak and AES.

Acknowledgements

This project has received funding from the European Research Council (ERC) under EU's Horizon 2020 research and innovation programme (grant agreement 681402).

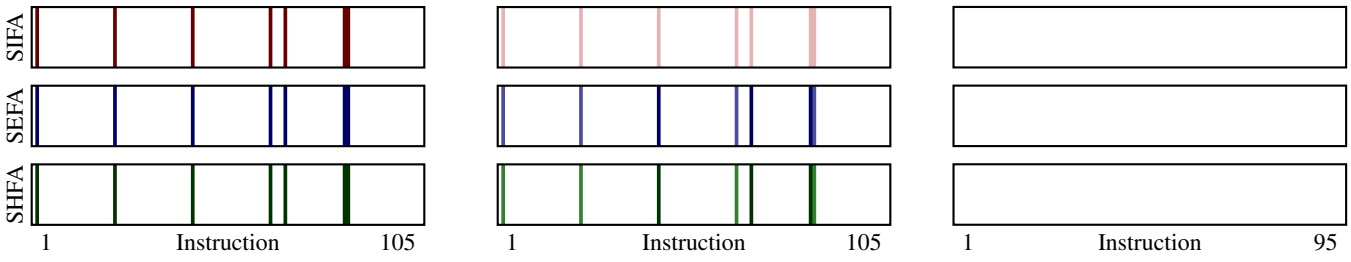
References

[1] N. Vafaei, S. Zarei, N. Bagheri, M. Eichlseder, R. Primas, and H. Soleimany, "Statistical effective fault attacks: The other side of the coin," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 1855–1867, 2022.

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

[3] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology – CRYPTO '97*, ser. LNCS, B. S. K. Jr., Ed., vol. 1294. Springer, 1997, pp. 513–525.

[4] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis," in *Cryptographic Hardware and Embedded Systems – CHES 2010*, ser. LNCS, S. Mangard and F.-X. Standaert, Eds., vol. 6225. Springer, 2010, pp. 320–334.



(a) DOM, reliable faults (b) DOM, unreliable faults (c) [26], reliable faults

Fig. 6: Practical evaluation of SIFA/SEFA/SHFA on different Keccak S-box implementations with combined countermeasures. Vertical lines indicate if injecting a fault in an instruction leads to a successful attack. Darker/lighter colors indicate stronger/weaker observed biases; white lines indicate no successful attack after 4000 faulted computations.

[5] T. Fuhr, É. Jaulmes, V. Lomné, and A. Thillard, "Fault attacks on AES with faulty ciphertexts only," in *Fault Diagnosis and Tolerance in Cryptography – FDTC 2013*, W. Fischer and J.-M. Schmidt, Eds. IEEE Computer Society, 2013, pp. 108–118.

[6] C. Dobraunig, M. Eichlseder, T. Korak, V. Lomné, and F. Mendel, "Statistical fault attacks on nonce-based authenticated encryption schemes," in *Advances in Cryptology – ASIACRYPT 2016*, ser. LNCS, J. H. Cheon and T. Takagi, Eds., vol. 10031, 2016, pp. 369–395.

[7] W. Li, J. Li, D. Gu, C. Li, and T. Cai, "Statistical fault analysis of the simeck lightweight cipher in the ubiquitous sensor networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4224–4233, 2021. [Online]. Available: https://doi.org/10.1109/TIFS.2021.3102485

[8] M. Joye, J.-J. Quisquater, S.-M. Yen, and M. Yung, "Observability analysis – detecting when improved cryptosystems fail," in *Topics in Cryptology – CT-RSA 2002*, ser. LNCS, B. Preneel, Ed., vol. 2271. Springer, 2002, pp. 17–29.

[9] S.-M. Yen and M. Joye, "Checking before output may not be enough against fault-based cryptanalysis," *IEEE Transactions on Computers*, vol. 49, no. 9, pp. 967–970, 2000.

[10] S.-M. Yen, S. Kim, S. Lim, and S.-J. Moon, "A countermeasure against one physical cryptanalysis may benefit another attack," in *Information Security and Cryptology – ICISC 2001*, ser. LNCS, K. Kim, Ed., vol. 2288. Springer, 2001, pp. 414–427.

[11] C. Dobraunig, M. Eichlseder, T. Korak, S. Mangard, F. Mendel, and R. Primas, "SIFA: Exploiting ineffective fault inductions on symmetric cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, pp. 547–572, 2018.

[12] G. Barbu, L. Castelnovi, and T. Chabrier, "Generalizing statistical ineffective fault attacks in the spirit of side-channel attacks," in *COSADE*, ser. Lecture Notes in Computer Science, vol. 12910. Springer, 2021, pp. 105–125.

[13] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proc. IEEE*, vol. 94, no. 2, pp. 370–382, 2006.

[14] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay, "Destroying fault invariant with randomization – A countermeasure for AES against differential fault attacks," in *Cryptographic Hardware and Embedded Systems – CHES 2014*, ser. LNCS, L. Batina and M. Robshaw, Eds., vol. 8731. Springer, 2014, pp. 93–111.

[15] B. Gierlichs, J.-M. Schmidt, and M. Tunstall, "Infective computation and dummy rounds: Fault protection for block ciphers without check-before-output," in *Progress in Cryptology – LATINCRYPT 2012*, ser. LNCS, A. Hevia and G. Neven, Eds., vol. 7533. Springer, 2012, pp. 305–321.

[16] J. Feng, H. Chen, Y. Li, Z. Jiao, and W. Xi, "A framework for evaluation and analysis on infection countermeasures against fault attacks," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 391–406, 2020. [Online]. Available: https://doi.org/10.1109/TIFS.2019.2903653

[17] S. Saha, A. Bag, D. Basu Roy, S. Patranabis, and D. Mukhopadhyay, "Fault template attacks on block ciphers exploiting fault propagation," *Advances in Cryptology–EUROCRYPT 2020*, vol. 12105, pp. 612–643, 2020.

[18] T. Baignères, P. Junod, and S. Vaudenay, "How far can we go beyond linear cryptanalysis?" in *Advances in Cryptology – ASIACRYPT 2004*, ser. LNCS, P. J. Lee, Ed., vol. 3329. Springer, 2004, pp. 432–450.

[19] M. Rivain, "Differential fault analysis on DES middle rounds," in *Cryptographic Hardware and Embedded Systems – CHES 2009*, ser. LNCS, C. Clavier and K. Gaj, Eds., vol. 5747. Springer, 2009, pp. 457–469.

[20] T. Fuhr, É. Jaulmes, V. Lomné, and A. Thillard, "Fault attacks on AES with faulty ciphertexts only," in *Fault Diagnosis and Tolerance in Cryptography – FDTC 2013*, W. Fischer and J.-M. Schmidt, Eds. IEEE Computer Society, 2013, pp. 108–118.

[21] A. A. Selçuk, "On probability of success in linear and differential cryptanalysis," *Journal of Cryptology*, vol. 21, no. 1, pp. 131–147, 2008.

[22] C. Blondeau, B. Gérard, and K. Nyberg, "Multiple differential cryptanalysis using LLR and χ^2 statistics," in *Security and Cryptography for Networks – SCN 2012*, ser. LNCS, I. Visconti and R. D. Prisco, Eds., vol. 7485. Springer, 2012, pp. 343–360.

[23] C. Dobraunig, M. Eichlseder, H. Groß, S. Mangard, F. Mendel, and R. Primas, "Statistical ineffective fault attacks on masked AES with fault countermeasures," in *Advances in Cryptology – ASIACRYPT 2018*, ser. LNCS, T. Peyrin and S. D. Galbraith, Eds., vol. 11273. Springer, 2018, pp. 315–342.

[24] S. Saha, D. Jap, D. B. Roy, A. Chakraborty, S. Bhasin, and D. Mukhopadhyay, "A framework to counter statistical ineffective fault analysis of block ciphers using domain transformation and error correction," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1905–1919, 2020. [Online]. Available: https://doi.org/10.1109/TIFS.2019.2952262

[25] M. Gruber, M. Probst, P. Karl, T. Schamberger, L. Tebelmann, M. Tempelmeier, and G. Sigl, "Domrep–an orthogonal countermeasure for arbitrary order side-channel and fault attack protection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4321–4335, 2021.

[26] J. Daemen, C. Dobraunig, M. Eichlseder, H. Groß, F. Mendel, and R. Primas, "Protecting against statistical ineffective fault attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, pp. 508–543, 2020.

[27] T. Toffoli, "Reversible computing," in *Automata, Languages and Programming – ICALP 1980*, ser. LNCS, J. W. de Bakker and J. van Leeuwen, Eds., vol. 85. Springer, 1980, pp. 632–644.

[28] H. Groß, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," in *Theory of Implementation Security – TIS@CCS 2016*, B. Bilgin, S. Nikova, and V. Rijmen, Eds. ACM, 2016, p. 3.

[29] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *CRYPTO (1)*, ser. Lecture Notes in Computer Science, vol. 9215. Springer, 2015, pp. 764–783.

[30] M. Gruber, M. Probst, and M. Tempelmeier, "Statistical ineffective fault analysis of GIMLI," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2020, San Jose, CA, USA, December 7-11, 2020*. IEEE, 2020, pp. 252–261. [Online]. Available: https://doi.org/10.1109/HOST45689.2020.9300260

[31] A. R. Shahmirzadi, S. Rasoolzadeh, and A. Moradi, "Impeccable circuits II," in *Design Automation Conference – DAC 2020*. IEEE, 2020, pp. 1–6.

[32] P. Schwabe and K. Stoffelen, "All the AES you need on cortex-m3 and M4," in *SAC*, ser. LNCS, vol. 10532. Springer, 2016, pp. 180–194.

[33] C. O'Flynn and Z. D. Chen, "Chipwhisperer: An open-source platform for hardware embedded security research," in *COSADE*, ser. LNCS, vol. 8622. Springer, 2014, pp. 243–260.