IACR Transactions
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–19. DOI:XXXXXXXX

Linked Fault Analysis

Ali Asghar Beigizad[1], Hadi Soleimany[1] and Sara Zarei[1]

Cyber Research Center, Shahid Beheshti University, Tehran, Iran
beigizad@yahoo.com, h_soleimany@sbu.ac.ir, sarazareei.94@gmail.com

Abstract. Various fault models, each with a distinct effect, have been introduced. The process of injecting a fault is not overly complicated, however it can be challenging to inject an exploitable fault. The influence of a fault model should be evaluated based on characteristics like as cost, repeatability, and practicability of desirable faults. Additionally, there must be efficient techniques for leveraging the injected fault to retrieve the key, especially in the presence of common countermeasures.
In this paper we introduce a new fault analysis technique called "linked fault analysis"(LFA) which can be interpreted as a more powerful variation of several well-known fault attacks against implementations of symmetric primitives in various scenarios particularly in software implementations. While in a traditional fault attack, the fault model is defined based on the relation between the correct value and the defective one produced by fault injection, the LFA leverages a model in which the fault involves more than one intermediate value, the target variable X, and a second variable Y. We demonstrate that LFA allows the attacker to perform fault attacks with significantly less data (relative to previously presented fault attacks in the same class) and without the input control need.

Keywords: Fault Analysis · Linked Fault

1 Introduction

Fault attacks are a sort of physical attack in which the attacker deliberately produces a fault in the targeted device in order to obtain information about its secret key by observing the device's response to the fault. According to the duration of the fault, it is possible to classify induced faults into one of three categories. The vast majority of proposed fault attacks are based on transient faults, which only have a temporary impact on the system. With their application on RSA, Boneh et al. introduced this type of fault attack for the first time [BDL97]. The second type is permanent fault where their effects are permanent for the rest of the device's lifetime. The persistent fault is the third form of fault, which persists but can be removed by resetting the target device. Schmidt first introduced the basic concept [SHP09], which was then taken by Zhang et al., who offered a more specialized framework [ZLZ+18] that has gained a great deal of recent attention [PZRB19, GPT19, CB19, SBH+21, ESP20, CGR20, ZZJ+20].

The great majority of research on transient fault attacks against symmetric primitives focuses on the ability of this category of attacks to transform an intermediate value into a faulty one. The attacker adds a fault to value X to turn it into the defective value X'. Most of the time, it is assumed that the obtained faulty value and its corresponding correct value have some extractable relationships. One can mention fault models like stuck-at, bit-flips, random fault, random-AND fault, biased fault, etc., where the referred interconnection is distinct in each case. In this work, we acquire an unprecedented perspective to shine light on an out-of-sight sort of relation and propose a novel fault-based attack dubbed "linked fault analysis"(LFA) based on it. LFA can be interpreted as a more powerful variation

of several well-known fault attacks against software implementations of cryptographic primitives in various scenarios. While in a traditional fault attack, the fault model is defined based on the relation between the correct value and the defective one produced by fault injection, the LFA leverages a model in which the fault involves more than one intermediate value, the target variable X, and a second variable Y. The relationship brought on by the fault effect would be the change of the X value according to the Y value. We entitle this event as a linked change and take its advantage toward our intended novel analysis.

1.1 Overview of Prior Transient Fault Attacks

What would be a decent place to dive if you wanted to explore the vast universe of fault attacks? Classifications appear to be rational starting points to avoid getting lost between the various and varied fault attacks that have been proposed thus far and their attributes and application. Therefore, adhering to conventions, we, too, opt for this entryway to open our new outlook on this world and determine where our newly launched attack will land. However, before discussing different existing categorizations, we must first answer a critical question: Which criterion yields a more precise or thorough classification? Fortunately or not, this is not a single-answer question. Numerous variables can play a major or minor role in certain applications or settings. Different fault attack tactics include different demands and assumptions, each advantageous and appropriate in a particular context. The quantity of faulty and correct ciphertexts required to carry out a successful attack is, for example, a typical criterion used in the literate for comparing fault attacks. Though this is quite an essential attribute, especially when the number of queries is limited (for instance due to the fresh-key policy), in practical situations, the fewer ciphertexts may be of minor priority, and other factors gain heavier consideration. Consider, for instance, the attacker's ability to encrypt the same input several times. Or the behavior of the attack while encountering the underlying countermeasures. Our surveys indicate that considering these last two factors simultaneously, will cover a good number of most famous introduced attacks in the field. It can also pinpoint the precise location of *LFA*. Thus, in what follows, we attempt to provide more information about these two benchmarks and classify well-known existing fault attacks based on them.

Control vs. no-control over the inputs From this point of view, attacks can be classified based on whether the attacker can repeatedly encrypt a fixed plaintext or not. The fixed input is assumed to be chosen by the attacker in some of these attacks and in some others, this is not a necessary condition. In each case, the underlying assumption is the attacker's access to the input and continually asking for the same plaintext to be encrypted.

Bypassing vs. losing to redundant-based countermeasures In the presence of standard redundant-based countermeasures (infection-based [GST12, TBM14, FCL+20] or detection-based [BCN+06]), the adversary is unable to detect defective ciphertexts. Hence there it would not be possible for him/her to execute attacks that rely on the faulty ciphertext. Nevertheless, the fault cannot be discovered if it does not alter any intermediate value. This phenomenon is known as an ineffective fault. Some attacks are able to circumvent standard countermeasures because they exclusively employ such kind of events.

The classification of the aforementioned categories into four subcategories is shown in Table 1. Differential Fault Analysis (DFA) [BS97] requires that both correct and faulty ciphertexts correspond to a given plaintext. DFA has been widely applied to numerous unprotected implementations of block ciphers [SMC09, AM12, AM13, AMT13], authenticated encryption schemes [RCC+16] and even stream ciphers [DCAM15]. Extensive research has been conducted on DFA, and multiple generalized and automated frameworks have been developed to identify vulnerabilities against this form of attack [SMD18, BHL18, HBZL19].

Table 1: Classifications of Prior Transient Fault Attacks on Block Ciphers

	Input's Control	No need to Control
Cannot Bypass countermeasures	DFA, IDFA, DFIA, CFA	SFA, LDFA*
Bypass countermeasures	FTA, FSA, SEFA	IFA, SIFA, LIFA**

*,** Linked-DFA and Linked-IFA, proposed in this research (Section 3).

Similar to the classical type of differential cryptanalysis, the Impossible Diferential Fault Attack (IDFA) is a differential-based fault attack based on an impossible characteristic [DFL11]. Algebraic Fault Attack (AFA) is similar to DFA in that both faulty and correct outputs are analyzed [ZGZ+13, ZZG+13, ZGZ+16]. However, AFA constructs algebraic equations and focuses on the solution of a system of equations instead of using differential characteristics. Differential Fault Intensity Analysis (DFIA) [GYTS14] is another method in which the key-recovery procedure relies on the repetition of encryption for a fixed plaintext and the knowledge of the value of faulty ciphertexts. Collision Fault Analysis (CFA) [BK06] applies when an attacker can encrypt two related inputs, injects a fault into one of the computations, and then exploits a collision that may occur (usually in the outputs). Therefore, redundant-based countermeasures can prevent these attacks, and the attacker must also control the input to perform these attacks.

Fault Sensitivity Analysis (FSA) [LSG+10] needs encrypting the same plaintext during the profiling phase in order to determine the critical fault injection intensity for various plaintexts. The adversary should be able to determine whether the injected flaw was effective. Therefore, FSA does not necessarily require faulty ciphertexts. Fault Template Attack (FTA) [SBR+20] is a promising technique that can circumvent redundant-base countermeasures since it does not rely on faulty ciphertexts. However, an attacker should be able to encrypt a fixed (but unknown) plaintext several times. Therefore, both SFA and FTA can circumvent redundant-based countermeasures, but in practice the attacker must control the input to repeat the encryption process of the same input.

Statistical Fault Analysis (SFA) [FJLT13] demands faulty ciphertexts but does not require repeated encryption over a particular plaintext; therefore, it can be thwarted by redundant-based countermeasures.

Several attacks without input control have been presented to circumvent redundant-based countermeasures. These attacks exclusively target cases in which the error has no effect on the computation. This class includes the Safe Error Attack (SEA) [YJ00] , which is typically relevant to public-key cryptographic systems, as well as the Ineffective Fault Analysis (IFA) [Cla07] and its statistical counterpart (SIFA) [DEK+18]. IFA typically uses models such as stuck-at faults, which are difficult to achieve in practice and require the use of costly tools such laser [SBHS15, SHS16].

1.2 Our Contributions

In certain applications, it is not possible to encrypt (or sign) the same message more than once. Protocols in which the message is padded with a random value are a prominent illustration of such situations. Consequently, fault attacks that require input control (whether they require chosen plaintexts or repeated computation with an unknown input) are inapplicable in such applications. Consider as another example the devices that use block ciphers in operation modes such as Cipher feedback (CFB), Output feedback (OFB), and Counter mode (CTR). Despite the fact that it is frequently believed that the underlying block cipher is utilized in ECB mode, alternative modes like CTR, OFB, and CFB may also be applied in real-world applications, where the input is fundamentally not repeated. The modes construct the following block by encrypting successive values which depend on a value "counter" or "initial value (IV)" that will not repeat over an extended period.

Attacks that require input control are challenging to carry out in these applications because it is not always possible for an attacker to reset the value of the counter [BBB+18].

Methods that do not require the attacker to have input control frequently rely on particular statistical characteristics. As a result, the key recovery approach often needs more samples (in comparison to attacks with input control) to statistically distinguish the right key from the wrong one. The first proposed fault attack against symmetric primitives, DFA, involves input control but uses a lot fewer data than SFA. The tiny amount of data can be important in some real-world situations. However, as this kind of attack often requires input control, acquiring less data is not possible for free. In this paper, we propose a new fault technique that allows us to perform fault attacks with significantly less data (relative to previously presented fault attacks in the same class) and without the input control need.

The existence of relationships between two intermediate values, one in the correct computation process and the other in the faulty computation, is a prerequisite for DFA and CFA. This trait enables the attacker to make use of these relations and carry out the attack despite the limited data availability. We use a similar strategy in our first suggested method, however, our approach is based on correlations between the intermediate values of only faulty computation. This can be regarded within the framework of the differential fault approach as an internal differential property that does not require access to the correct computation. Thus, our offered technique can be conducted with a very small number of inputs (similar to DFA and CFA) and does not require repeating the encryption process (similar to SFA).

We demonstrate our method's considerable adaptability by showing its applicability to ineffective ciphertexts too. The power of these ciphertexts is in circumventing redundant-based countermeasures. Similar to attacks that do not require input control and are applicable in the face of countermeasures (such as SIFA), our method can greatly minimize the amount of required data. Additionally, our method gives the attacker the ability to identify missed fault occurrences under specific circumstances, which is essential in real-world applications because missed faults can significantly affect SIFA's performance.

Skipping the execution of a single or multiple microprocessor instruction is a highly effective way for modifying processed values. Variation the input voltage and inducing a clock glitch are inexpensive approaches that can result a variety of faults. One should note that certain faults, such as those that target the equality check in detection-based countermeasures or reduce the number of iterations in a loop, are apparent targets and must therefore be protected. So, the community is already aware of the necessity to protect these obvious faulting targets with additional safeguards.

2 Preliminaries

2.1 Notations

While most of the fault attacks are not specific to any particular cipher, in the interest of clarity, we will describe the attacks with an illustration using an R-round word-oriented Substitution-Permutation Network (SPN) cipher. The majority of block ciphers, such as AES, have SPN structure. Let us assume a standard SPN design $E_K(P)$ which accepts a b-bit plaintext P and k-bit key to produce the cipehrtext C. The process of encrypting the plaintext using the cipher can be viewed as an iteration of an invertible function that is referred to as round funtion $F_{sk_r}()$, where sk_r stands for the r-th round key for $1 \leq r \leq R$. As implied by the name of the structure, each round of the cipher employed a substitution $\mathcal{NL}$ (non-linear layer) and a permutation $\mathcal{L}$ (linear layer), followed by the addition of a round key. Typically, the substitution layer is constructed by combining a number of relatively basic nonlinear bijective functions known as Sboxes. The b-bit state formed of L

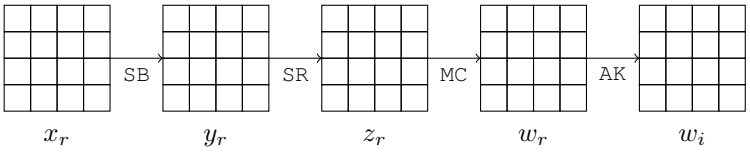

Figure 1: AES round function

words of the same length ($m = b/L$) and are represented. Let $x_r[i]$ and $y_r[i]$ represent, respectively, the $i-$th word input into the substitution layer in the $r-$th round and the $i-$th word output from it where $0 \leq j \leq L-1$ and $1 \leq r \leq R$. As $\mathcal{L}$ is linear, it could be eliminated in the final round or replaced with a simpler version. A group operation, like addition and XOR, can be employed to combine subkeys into states.

While our approaches are applicable to any block cipher, we focus on AES because it is the most widely used cipher in practical applications.AES is a 128-bit block cipher with a key size of 128, 192, or 256 bits with a substitution-permutation network (SPN) structure. AES operates on an array of eight bytes ($m = 8$). In our experiments, we considered AES with 128-bit keys, denoted by AES-128, which contains $R = 10$ rounds, each of which involves four transformations: SubByte (SB), ShiftRows (SR), MixColumns (MC), and AddRoundKey (AK). Exceptionally, the last round lacks MixColumns. x_r represents the state input of round r, while y_r, z_r, and w_r represent the states after applying the SB, SR, and MC operations of round r, respectively. Since the output of the $r-$th round is the input of the $(r+1)-$th round, we represent it with the notation x_{r+1}.

2.2 DFA

Differential fault analysis was originally introduced in [BS97] and is the most prevalent technique for fault analysis. The assumption is that the attacker can inject a difference at a certain intermediate value in a given round. In other words, the attacker is aware of the distinction between the faulty value and the correct value in the intermediate value. Typically, the injected fault is injected into one of the state's words preceding the nonlinear layer. In the basic technique for the key-recovery procedure, the attacker guesses the subkey bits involved in the last rounds of the cipher and partially decrypts both faulty and correct ciphertexts in order to compute the desired intermediate value. It is expected that the intended difference will be observed more frequently for the correct key than for the wrong key. This technique exploits the fact that the input difference and output difference of Sboxes have a precise relationship.

2.3 SIFA

The Statistical Fault Attack employs a biased distribution across an intermediate value. The attacker decrypts given faulty ciphertexts with the key candidates across the final round(s) of the cipher and computes a statistical scoring function $\mathcal{S}(\hat{p})$ for each key candidate to determine how closely the computed distribution matches the predicted distribution with the correct key. Given N samples, depending on whether the attacker is aware or unaware of the faulty value's distribution p, different statistical test can be utilized.

If distribution of faulty value p is known, log-likelihood ratio (LLR) statistic can be used as it is described in Equation (1), where θ denotes the uniform distribution.

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

If the faulty distribution is biased but it is unknown to the attacker, Squared Euclidean Imbalance (SEI) can be used as sown in Equation (2).

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

As a result, the attacker either use $\mathcal{S}(\hat{p}) = \mathrm{LLR}(k)$ or $\mathcal{S}(\hat{p}) = \mathrm{SEI}(k)$ to rank the candidates of key. $\mathcal{S}(\hat{p})$ follows a normal distribution large amount of samples N is obtained by the attacker:

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

Selçuk [Sel08] proved the difference Δ_a between the score of correct key k and the score of a wrong key with rank $2^{\kappa-a}$ of 2^κ possible keys also has a normal distribution:

$$\Delta_a \sim \mathcal{N}(\mu_\Delta, \sigma_\Delta^2) \,, \tag{3}$$

where $\mu_\Delta = \mu_p - \mu_\theta - \sigma_\theta \cdot \Phi_{0,1}^{-1}(1 - 2^{-a})$ and $\sigma_\Delta \approx \sigma_p$. The quantity a is referred as *advantage* of the attack and can be up to κ. The success probability of an attack with a-bit advantage,can be estimated based on Equation (4) [BGN12, Sel08]:

$$\mathbb{P}(\Delta_a > 0) = \Phi_{0,1}\left(\frac{\mu_p - \mu_\theta - \sigma_\theta \cdot \Phi_{0,1}^{-1}(1 - 2^a)}{\sigma_p}\right) . \tag{4}$$

The number of required ciphertexts in LLR and SEI tests can be estimated based on Equation (5) and Equation (6), repsectively.

$$N_{\mathrm{LLR}} \approx \frac{2 \cdot [\Phi_{0,1}^{-1}(P_{\mathrm{S}}) + \Phi_{0,1}^{-1}(\alpha)]^2}{C(p,\theta)} \tag{5}$$

$$N_{\mathrm{SEI}} \approx \frac{\beta \cdot \Phi_{0,1}^{-1}(\alpha)}{C(p,\theta)} \qquad \text{for } P_{\mathrm{S}} = 0.5. \tag{6}$$

where $C(p,\theta)$ deontes the capacity and is given in Equation (7) in case p is close to θ.

$$C(p,\theta) = \sum_{x \in \mathcal{X}} \frac{(p(x) - \theta(x))^2}{\theta(x)} \,. \tag{7}$$

SIFA is simultaneously enabled by SFA and IFA methods. SIFA utilizes a biased distribution of intermediate value over ineffective faults. SIFA is able to evade detection- and infection-based countermeasures since it only requires ineffective outputs.

2.4 Limitations of Instruction Skip Attacks in Block ciphers

A variety of laser pulses [BCN+06, TK10], electromagnetic pulses [DDR+12, MDH+13a, DDRT12], clock glitches [BGV11, KH14, EHH+14, YGS15, YGS+16], and power glitches [SH08] have been suggested as methods for skipping one or more instructions on microcontrollers. The majority of these attacks on symmetric-key primitives need input control, and more significantly, they cannot circumvent common countermeasures by nature. Following is a brief discussion of instruction-skipping attacks against block ciphers.

The loops and functions can be easily disrupted by a fault injection; hence, it is feasible to bypass a significant portion of the encryption and perform standard cryptanalysis, such as differential cryptanalysis, to the output [MDH+13b]. It is doubtful that a reduced-round cipher will yield ineffective output, preventing this type of attack from evading redundant-based countermeasures.

An other possible attack is to bypass the addition instruction of the final round key, which is added to the state via an exclusive-or operation [BGV11, BJC15]. The last round subkey can be easily obtained by computing the exclusive-or of the faulty ciphertext with the correct ciphertext. Skipping addition is a well-known attack technique that has been also employed against stream ciphers [FXKH17]. Pessl and Prokop recently demonstrated at CHES 2021 that addition instruction skip can be used to attack lattice-based KEMs [PP21]. Despite the public key schemes [SH08, BBKN12], this approach is not typically relevant to symmetric primitives in the presence of standard countermeasures. Given that symmetric primitives are not based on algebraic structures, skipping an operation typically results in an active fault that can be identified by redundancy.

Another method is to skip the equality check for skipping the conditional branching following the redundant computation [EHH+14].

3 Linked Fault Analysis (LFA)

3.1 Fautl Model

We target software-based implementations on microcontrollers by introducing a new fault analysis which exploits "instruction skips," a common fault injection technique on these devices. As we will demonstrate in the following sections, our method is effective and practical, can be conducted with inexpensive tools, and can circumvent countermeasures based on redundancy. In our attack's model we assume that the attacker can control the intensity and duration of the fault. Besides, we presume, similar to the majority of proposed fault attacks, that the attacker is aware of the timing. It should be noted that in most circumstances, it is assumed that the fault's location (time) is known to the attacker, however this can be established by trial and error. We assume that the nonlinear layer of the target block cipher is implemented using one or more precomputed look-up tables that are known to the attacker, and that each table is called several times throughout round function execution.

In this paper, we employ the instructions skip technique to induce a fault on an intermediate value v such that the faulty value v' is linked to another intermediate value u, where the attacker is unaware of the values of u, v and v'. We show that under some conditions, if u and v are processed after one another, the value of faulty v (i.e. v') becomes identical to the value of u. This fault can be induced using cheap instruments through the use of well-timed power spikes or clock glitches. This linked fault across intermediate values may result in linked words in the ciphertexts, which an attacker can use to obtain information about the secret key. This fault can be happened by instruction skip in at least two situations:

1. Sometimes, in order to accelerate the encryption process of software implementation, auxiliary variables are used to store the words of the input state that will be processed in parallel during the subsequent steps. The sequential loading of two distinct words of the input state into the variables like u and v is an appropriate target for our attack.

2. The nonlinear layer of block ciphers typically consists of Sboxes. To execute the nonlinear layer, the Sbox (or Sboxes) is progressively invoked with distinct values. Alternately, one may utilize larger, precomputed look-up tables for both nonlinear and linear layers. However,loading the output of such look-up tables is an integral feature of the vast majority of block ciphers and can be exploited to generate a linked fault in the input or output of the called function.

For the time being, we will assume that the instruction skip occurs perfectly. We will

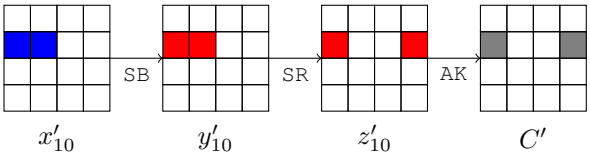

Figure 2: Last Round of AES: Similar-colored bytes are linked together.

discuss the effect of missed or unwanted faults later in the last subsection and explain how to deal with them.

3.2 Linked Differential Fault Attack (LDFA)

In the absence of redundant-based Countermeasures, attacks like DFA and CFA are applicable. A common property of DFA and CFA is utilizing the existence of two intermediate values that are either identical or related in the process of correct and faulty computations. This property permits the attacker utilize deterministic relations and perform the attack with extremely small amount of data. In our first proposed method, we apply this same idea, but instead, our method is based on relations between the intermediate values of the faulty computation itself. For the sake of clarity, we will first discuss our attack against AES, followed by a generic key-recovery approach against any cipher.

Assume that the byte $x[j]$ becomes equal to $x[i]$) or $y[j]$ becomes equal to $S(x[i])$ when instruction skip is injected during the execution of Sboxes. This fault causes two bytes (with unknown values), $z'_{10}[SR[i]]$ and $z'_{10}[SR[j]]$, to become equal following a ShiftRow operation. After adding the subkey sk_{10}, the corresponding bytes of ciphertexts $C'[SR[i]]$ and $C'[SR[i]]$ are not necessarily exactly equal but they can be used to retrieve information about the subkey. Figure 2 demonstrates the propagation of a linked fault for $i = 1$ and $j = 5$. Given a faulty ciphertext C', the attacker can guess the value of two bytes of the last round $sk_{10}[SR[i]]$ $sk_{10}[SR[j]]$, and partially decrypt the ciphertext C' to compute intermediate values $z'_{10}[SR[i]]$ and $z'_{10}[SR[j]]$ for the guessed key. If $z'_{10}[SR[i]] \neq z'_{10}[SR[j]]$, the guessed key is wrong otherwise it can be candidate for the correct key. The probability a wrong key survives is 2^{-8}. Since there are 2^{16} possible values for $sk_{10}[SR[i]]$ $sk_{10}[SR[j]]$, only $2^{16} \cdot 2^{-8} = 2^8$ candidates remain. Hence, the number of remaining key candidates is reduced by a factor of 2^8 with evaluating the faulty ciphertext of each linked fault. In other words, the attacker can retrieve 8-bit information about the last round key by injecting only one fault and utilizing only one faulty ciphertext. Given h distinct linked faults and faulty ciphertexts, the residual key entropy reduces from k bits to $(k - 8 \times h)$ bits. As previously noted in Section 3.1, the actual number of linked faults that an attacker is able to inject relies on the implementation and, more specifically, the number of look up tables that are called throughout the round function's execution.

While we presented the attack for the AES cipher, it may be applied to any cipher by adopting a similar technique that has been utilized for key recovery in various differential fault attacks. Assume that u and v, two $m-$bit intermediate values in the r-th round, are linked after fault injection. In other words, u and v either satisfy the relation $u' = l(v)$, for example, they are equal. The intermediate values u and v, as well as some subkey bits over the last round(s), influence a portion of the ciphertext. In other words, it is possible to determine the intermediate values u' and v by guessing the value of these subkey bits. These bits are referred to as the target key bits. There are $\tau = 2^\kappa$ candidates for a κ-bit target key. Given N faulty ciphertexts $C'_1, C'_2, ..., C'_N$, it is possible to partially decrypt the ciphertexts for a guessed key k and determine if the relation $u' = l(v)$ holds. This technique is illustrated in Algorithm 1. For the correct key, the relation $u' = l(v)$ holds in all cases, whereas for the wrong key, it holds with probability $2^{-m \cdot N}$. Equation (8) can be used to compute the number of faulty ciphertexts needed to eliminate all wrong key

candidates.

$$\tau \cdot 2^{-m \cdot N} < 1 \Rightarrow 2^{\kappa - m \cdot N} \leq 1 \Rightarrow \kappa - m \cdot N \leq 0 \Rightarrow N \geq \frac{\kappa}{m} \tag{8}$$

Algorithm 1 Key-recovery process in LDFA

Require: Faulty ciphertexts $C'_1, C'_2, \ldots, C'_N$, Key candidates for last round(s) $k_0, ..., k_{\tau-1}$.
Ensure: Correct Key

```
for ℓ = 0 to (τ − 1) do
    for h = 1 to N do
        (u′, v) ← E^{-1}_{k_ℓ}(C′_h)                ▷ Partial decryption
        if u′ = l(v) then
            cnt[ℓ] = cnt[ℓ] + 1
        end if
    end for
end for
return argmax_ℓ(cnt[ℓ])
```

One should note that in contrast to DFA, this method does not require the correct ciphertext. Hence the attacker does not need to repeat the encryption process twice for a specific input value, which requires control over the input. We call this technique as Linked Differential Fault Attack (LDFA) because it leverages the difference between the bytes of ciphertexts that are linked in some way. However, the benefit of LDFA over other approaches such as DFA and SFA is evident. LDFA uses linked bytes to take advantage of the favorable features of known attacks (limited number of data necessary in DFA and no need to control input in SFA) and overcome their limitations (control of input in DFA and CFA, and a large number of required data in SFA).

Similar to attacks such as DFA and SFA, LDFA cannot circumvent redundant-based countermeasures. In the following part, we demonstrate that linked fault analysis is adaptable to be utilized in applications employing conventional countermeasures.

3.3 Linked Ineffective Fault Attack (LIFA)

Some proposed attacks concentrate on ineffectual faults. As they let the attacker to circumvent several fault countermeasures, such as detection-based and infection-based countermeasures, these attacks have an obvious benefit. Injecting a stuck-at fault that is proposed in IFA is not a simple process, and it is not always practicable due to the need for costly equipment and specific knowledge about the target. SIFA utilizes an unbiased distribution of a targeted intermediate value over ineffective events. SIFA does not necessitate a complicated mechanism to inject the fault, but it cannot be performed with a few ineffective ciphertexts. To distinguish the correct key from a wrong key, sufficient ineffective ciphertexts must be collected in order to detect the desired bias in the distribution of the targeted intermediate value.

In contrast to IFA, LFA does not require complex equipment. LFA, unlike SIFA, does not rely on a static that cannot be distinguished by a small number of data. Therefore, linked fault analysis appears to be a suitable technique when a redundant countermeasure is utilized in the target device. Following is a description of how LFA can be easily adapted to be performed over ineffective ciphertexts.

Similar to the previous subsection, we first outline the attack by considering a linked fault in AES that causes the byte $x'[j]$ to become equal to $x[i]$ or $y'[j]$ to become equal to $S(x[i])$. The faulty ciphertext C' is unavailable to the attacker when redundant-based countermeasures are present. Nevertheless, if $y_{10}[i]$ is equal to $y_{10}[j]$, the fault has no effect on the processed data. In other words, if the relation $y_{10}[i] = y_{10}[j]$ holds, the fault is ineffective, and the device outputs the faulty ciphertext, which is actually the correct one. The probability of observing an ineffective ciphertext is 2^{-8}. This indicates that,

on average, the attacker must repeat the process 2^8 times for each fault. After that, the attacker can retrieve 8-bit information about the key in a similar manner explained in the previous subsection by observing an ineffective event.

Generally speaking, linked faults occur between two m-bit intermediate values u and v, where m is quite small (typically 4 or 8 in most of the ciphers). Adopting LFA over ineffective events is therefore highly efficient, as one ineffective fault happens on average for every 2^m faulty computations. Since LFA requires a very small number, the total amount of data required for LIFA might be significantly less than for other ineffective fault variant attacks.

3.4 Utilizing a Link Between the Words of Faulty Ciphertext

As previously discussed, LFA has an intriguing feature that produces a link between the faulty value (v') and another value (u). The existence of such a link may be conveyed to the faulty ciphertext (C') in the sense that one portion of the faulty ciphertext (e.g. $C'[i]$) has a key-dependent link to another portion (e.g. $C'[j]$), although likely with a more complex relation (e.g. $C'[i] = f(C'[j], sk_R)$) holds for a known function f). This part demonstrates that this link can be utilized in two situations. First, we demonstrate that it is likely to be employed for a key-recovery attack that is faster. Then, we demonstrate that it assists the attacker in determining the location of the flaw if the attacker lacks detailed knowledge of the implementation. In order to demonstrate the advantage of a key-dependent link in a faulty ciphertext, we will analyze the LFA on AES example from the preceding sections.

3.4.1 More Efficient Key Recovery

In previous subsections, the secret key was retrieved in a conventional manner. Pilling of the last round for a guessed key and checking a characteristic for the intermediate value(s) is a key-recovery method that dates back to the 1990s, prior to the introduction of fault attacks. In this section, we demonstrate that it may be possible to retrieve the key more efficiently without requiring the partial decryption of (faulty) ciphertexts in a traditional manner.

In the following part, for the sake of clarity, we will study the attack on the final round of AES to explain how one portion of the faulty ciphertext can be linked to another via a key-dependent relationship. As shown in Equation (9), it is straightforward to verify that the bytes of faulty ciphertexts $C'[SR[i]]$ and $C'[SR[j]]$ are linked based on the value $\Delta sk_{10}[i,j] \triangleq sk_{10}[SR[i]] \oplus sk_{10}[SR[j]]$.

$$C'[SR[i]] \oplus C'[SR[j]] = (z'_{10}[SR[i]] \oplus sk_{10}[SR[i]]) \oplus (z'_{10}[SR[j]] \oplus sk_{10}[SR[j]]) = \Delta sk_{10}[i,j] \tag{9}$$

Equation (9) states that the relation between the linked bytes of ciphertext depends on the value of the key and it is deterministic.

$$\Pr(C'[SR[i]] \oplus C'[SR[j]] = \alpha) = \begin{cases} 1 & \alpha = \Delta sk_{10}[i,j], \\ 0 & \alpha \neq \Delta sk_{10}[i,j]. \end{cases} \tag{10}$$

As illustrated in Algorithm 2, the relation given in Equation (10) allows an attacker to extract one byte of information about the final subkey with only one faulty ciphertext and without the need to perform costly key guessing.

It worth mentioning that in case of ineffective fault, $z_{10}[SR[i]]$ and $z_{10}[SR[j]]$ have a similar relationship (after the ShiftRow operation) and Algorithm 2 can be used similarly but over ineffective ciphertexts.

$$\Pr(C[SR[i]] \oplus C[SR[j]] = \alpha|\mathsf{i}) = \begin{cases} 1 & \alpha = \Delta sk_{10}[i,j], \\ 0 & \alpha \neq \Delta sk_{10}[i,j]. \end{cases} \tag{11}$$

Algorithm 2 Efficient key-recvoery attack on last round AES

Require: Faulty ciphertexts $C'_1, C'_2, \ldots, C'_N$ ▷ In case of LIFA, only ineffective ciphertexts are considered
Ensure: Correct value of $\Delta sk_{10}[i,j]$
1: **for** $h = 1$ to N **do**
2: $\quad \delta \leftarrow C'_h[SR[i]] \oplus C'_h[SR[j]]$
3: $\quad cnt[\delta] = cnt[\delta] + 1$
4: **end for**
5: **return** $argmax_\delta(cnt[\delta])$

3.4.2 Unknown Fault's Location

In this section, we illustrate that the assumption of precise knowledge of the loading order of intermediate values can be relaxed if some ciphertext words have a specified relationship similar to the preceding section's illustration. If we suppose that an m-bit relation $C'[i] = f(C'[j], sk_R]$ holds in a general instance, the attacker may be able to determine the values of i and j. The basic principle is to examine the relationship $C'[i] = f(C'[j], sk_R]$ for all possible values of i and j over all key candidates. This process is illustrated in Algorithm 3. If we take two portions of ciphertexts that are not affected by the fault, this relation is expected to hold with a probability of 2^{-m} even for the correct key, whereas it always holds for a right guess of i and j for the correct key. Given N faulty ciphertexts and assuming that there are ℓ_1, ℓ_2, and τ candidates for i, j, and sk_R, respectively, the relation holds about $\ell_1 \cdot \ell_2 \cdot \tau 2^{-m}$ times for an incorrect guess. Equation (12) can be used to compute the number of faulty ciphertexts required to eliminate all wrong guesses. For example, the last round attack on AES has respectively $\ell_1 = 16$, $\ell_2 = 16$, and $\tau = 256$ possibilities for i, j, and sk_R (here it refers to $\Delta sk_{10}[i,j]$), while m = 8-bit filtering is provided. Given N=3 faulty ciphertexts, the attacker is able to determine not only the value of $\Delta sk_{10}[i,j]$, but also the location of the fault and how it is linked to other word. It should be noted that a similar technique may be applied to LIFA, however rather than exploiting faulty ciphertext, the attacker applies Algorithm 3 on ineffective ciphertexts.

$$\ell_1 \cdot \ell_2 \cdot \tau 2^{-m \cdot N} < 1 \Rightarrow 2^{log(\tau+\ell_1+\ell_2)-m \cdot N} \leq 1 \Rightarrow \log(\tau+\ell_1+\ell_2) - m \cdot N \leq 0 \Rightarrow N \geq \frac{\log(\tau+\ell_1+\ell_2)}{m} \tag{12}$$

Algorithm 3 Finding the location of fault

Require: Faulty ciphertexts $C'_1, C'_2, \ldots, C'_N$ ▷ In case of LIFA, only ineffective ciphertexts are considered
Ensure: Location of fault and its linked word (i,j), and correct value of $\Delta sk_{10}[i,j]$
1: **for** $h = 1$ to N **do**
2: $\quad$ **for** $i = 0$ to 15 **do**
3: $\quad\quad$ **for** $j = i + 1$ to 15 **do**
4: $\quad\quad\quad \delta \leftarrow C'_h[SR[i]] \oplus C'_h[SR[j]]$
5: $\quad\quad\quad cnt_{i,j}[\delta] = cnt_{i,j}[\delta] + 1$
6: $\quad\quad$ **end for**
7: $\quad$ **end for**
8: **end for**
9: **return** $argmax_{i,j,\delta}(cnt_{i,j}[\delta])$

3.5 Affect of Missed and Unwanted Faults

Until now, we have assumed that fault injection is perfect, which is rarely the case in practice. It is possible that fault will not be injected as desired due to a number of factors. In this part, we first explore the influence of these occurrences and then demonstrate that LFA is less affected by undesirable events than other fault analyses.

3.5.1 Unwanted Faults

In certain instances, the instruction skip may occur in an undesirable location. "Unwanted fault" describes a situation in which the fault is not injected at the desired time (or location). Injecting fault at the incorrect time can result in an undesirable fault. This can occur due to either the unintentional noise that occurs naturally in practical experiments or the intentional noise that results from hiding countermeasures. Hiding countermeasures, such as shuffling and dummy operations, are intended to make fault injection into a specific place more complex by restricting the attacker's knowledge of the precise moment of instruction execution and processed data. Dummy operations incur significant additional expenses, and their impact against attacks like SEFA is limited. However, shuffling appears to be a successful, low-overhead countermeasure against the vast majority of fault attacks.

3.5.2 Missed Faults

There may be instances in which no instruction skip occurs. These instances, which cannot be distinguished from ineffective events, are referred to as "missed faults". Even with robust equipment and perfect setup, it is possible to have missed faults. Due to the fact that missed faults might be interpreted as ineffective faults, they have a substantial effect on fault attacks that employ ineffective events. This fact stimulated the development of novel attacks [PP21, VZB+22] that utilize correct ciphertexts that correspond to effective faults, called non-faulty effective ciphertexts. However, an attacker must perform the encryption twice to obtain non-faulty effective ciphertexts: once to compute the faulty encryption to determine if it is effective, and again to obtain the corresponding correct ciphertext.

3.5.3 LFA in the Presence of Undesired Cases

Most fault attacks, if not all, effect through missed and unwanted faults since they are typically indistinguishable from desired faults, especially when the attacker lacks input control to repeat the experiment with a fixed input. In this part, we demonstrate how LFA may manage undesirable situations without requiring input-controlled computation repetition.

In this part, we solely discuss situations where extra countermeasures, such as shuffling, are not employed. It indicates that the noise is the result of a missed fault or that the fault was introduced at the incorrect moment due to a weakness in the attack's setup. We denote the rate of missed faults and unwanted (and unintentional) faults by Π_m and Π_u, respectively. Performing N experiments, there exist about $N \times (1 - (\Pi_m + \Pi_u))$ cases in which the relation $u' = l(v)$ always holds. Since no fault occurs or it occurs in wrong location for other $N \times (\Pi_m + \Pi_u)$ cases, the relation $u' = l(v)$ holds with the probability of 2^{-m}. In contrast to the situation which was discussed in Section 3.2 and Section 3.3, the relation $u' = l(v)$ is not deterministic. However, as it is shown in Equation (13), $u' = l(v)$ satisfies for the correct key and a wrong key with different probability.

$$\Pr(u' = l(v)) = \begin{cases} (1 - (\Pi_m + \Pi_u)) + 2^{-m} \cdot (\Pi_m + \Pi_u) & \text{Correct key} \\ 2^{-m} \cdot (\Pi_m + \Pi_u) & \text{Wrong key.} \end{cases} \tag{13}$$

Hence, the attack can apply Algorithm 1 similarly but he requires utilize more data. Similar discussion holds for the key recovery attack presented in Section 3.4.1 in which we assume a key-dependent relation like $C'[i] = f(C'[j], sk_R))$ holds for two parts of faulty ciphertext where f is a known function. For instance we can consider again the last round attack on AES. Since no fault occurs or it occurs in wrong location for other $N \times (\Pi_m + \Pi_u)$ outputs, the value of $C[SR[i]] + C[SR[j]]$ is a random 8-bit value. Hence, the distribution of is not uniform as it is indicated in Equation (14).

$$\Pr(C[SR[i]] \oplus C[SR[j]] = \alpha) = \begin{cases} (1 - (\Pi_m + \Pi_u)) + 2^{-8} \cdot (\Pi_m + \Pi_u) & \alpha = \Delta sk_{10}[i,j], \\ 2^{-8} \cdot (\Pi_m + \Pi_u) & \alpha \neq \Delta sk_{10}[i,j]. \end{cases} \tag{14}$$

Consequently, Algorithm 2 can be applied similarly here but the required data increases.

4 Countermeasures and Further Discussions

4.1 Shuffling

Shuffling is a simple and effective countermeasure against fault attacks. The main principle behind shuffling is to randomize the order of operations that can be executed in any order. Sboxes can be executed in any order when it comes to block ciphers, for example. In the presence of shuffling, an attacker cannot predict the order of the calling look-up tables because it is randomized each cipher execution. As a result, injecting fault at a certain time does not necessarily affect a fixed intermediate value. This is why the performance of the majority of fault attacks is severely impacted by shuffling. Even SEFA, which is substantially less susceptible to noise than SIFA, is shown to be equally affected by the shuffling.

As explained in Section 3.4.2, in the key-recovery procedure of LFA, the attacker should not necessarily know which byte has been altered and how it is linked to other bytes. Even in this instance, it is presumed that the fault location and its linked word have been fixed (but unknown). In the key-recovery procedure of LFA, the attacker must be aware of which byte has been altered and how it is linked to other bytes. Input state contains L words that can be processed in $L(L-1)$ distinct ways. In order to generate a linked fault such as $u' = l(v)$, the values of u and v must be processed sequentially and at precisely the same moment that the attacker injects the fault (see Section 3.1). However, this is not the case when shuffling is present. After injecting a fault, if shuffling is employed, there are two possible outcomes. The intended fault occurs when the intermediate values u and v (or v and u) are processed during the injection of fault. The likelihood of such an occurrence is $\frac{2}{L(L-1)}$. With probability $1 - \frac{2}{L(L-1)}$, the intermediate values processed during fault injection deviate from what the attacker desires. In these instances, the relation $u' = l(v)$ may randomly satisfy with a probability of 2^{-m}. Hence, as it is shown in Equation (15), $u' = l(v)$ satisfies for the correct key and a wrong key with different probability.

$$\Pr(u' = l(v)) = \begin{cases} \frac{2}{L(L-1)} + 2^{-m} \cdot (1 - \frac{2}{L(L-1)}) & \text{Correct key} \\ 2^{-m} & \text{Wrong key.} \end{cases} \tag{15}$$

4.2 Repetition of instructions

The linked fault presented in this research can be generated in practice by skipping instructions. Various instruction-level countermeasures for defending the implementation against fault attacks have been researched and presented in prior studies. Idempotent instructions (such as the move instruction) can be implemented multiple times without

affecting the output [PM19]. This countermeasure should be utilized with caution, as the number of repetitions plays an essential role. For example, in our experiment, duplicating the instructions does not prevent the attack, but we might still perform LFA with more samples. In addition, there are some security concerns regarding the protection offered by multiple copies of instructions. While it is sometimes assumed that these countermeasures can only be compromised by injecting several faults using an expensive fault injection setup, it is demonstrated that their security can be compromised by injecting a single clock glitch with a low-cost tool [YGS+16]. Despite these issues, it is undeniable that instruction-level countermeasures such as the repeating of idempotent instructions can make the application of LFA substantially more difficult, particularly when combined with other countermeasures such as Shuffling.

5 Experiments

We come up with the realization of the linked fault by means of a basic yet typical setup. We set our target system's (μController's) clock to be fed by an external clock. The clock is then built utilizing a Field-Programmable Gate Array (FPGA). By doing so, the first achievement is the ability to increase the μController's working frequency from its highest achievable value to tens of times, using the FPGA's internal Phase-Locked Loop (PLL). Take an ATMEGA328p case, we have an increase from 16MHz to 160 MHz. The second achievement is the ability to induce the fault at the exact desirable point in time and algorithm calculations in every repetition of the fault inducement. In fact, the FPGA's high synchronicity with the μController allows us, as attackers, to have an accurate control over the time and location of our fault. The mechanism includes an alerting signal from the desired point of the algorithm (suppose the start of the next to last round of the AES). From the FPGA side, this signal triggers the commencement of the frequency perturbation. Two more subsidiary parameters aid in finely determining the exact point at which the frequency rise will result in an linked fault, as well as the duration of the increase. We refer to these two parameters as the the fault's *start* and *offset* value, respectively. In what follows, we describe and report the result of our first experiment.

We consider an 8-bit AVR μController (ATMEGA328p) on which the AES algorithm is running. We target the start of the last AES round. The AES implementation is carried out through a conventional C code with standard libraries in which the realization of the substitution box is by means of T-tables. Since our attack involves only a single instruction skip, the performance of the attack will not significantly be affected by the usage of assembly-optimized code. Our primary results show that out of 23862 tests, we successfully achieve linked fault in almost 99.1% of them. We observe ineffective faults, missed faults, and unwanted faults in 108, 42, and 54 of the experiments, respectively.

6 Conclusions

In this study, we present a new technique for fault analysis that employs a link between a faulty intermediate value and a second intermediate value. Our method, unlike most fault analysis techniques, does not rely on a biased fault model and has the potential to circumvent many countermeasures. We proved the impact of LFA by suggesting two key-recovery approaches. Our first technique, LDFA, is applicable to unprotected implementations of block ciphers that require an extremely minimal amount of data. LIFA, our second method, can circumvent conventional detection- and infection-based countermeasures. LIFA is less affected by missed faults or unwanted faults compared to SIFA, although, unlike SEFA, it does not require input control.

References

[AM12] Subidh Ali and Debdeep Mukhopadhyay. Differential fault analysis of twofish. In *Inscrypt*, volume 7763 of *Lecture Notes in Computer Science*, pages 10–28. Springer, 2012.

[AM13] Subidh Ali and Debdeep Mukhopadhyay. Improved differential fault analysis of CLEFIA. In *FDTC*, pages 60–70. IEEE Computer Society, 2013.

[AMT13] Subidh Ali, Debdeep Mukhopadhyay, and Michael Tunstall. Differential fault analysis of AES: towards reaching its limits. *J. Cryptogr. Eng.*, 3(2):73–97, 2013.

[BBB+18] Anubhab Baksi, Shivam Bhasin, Jakub Breier, Mustafa Khairallah, and Thomas Peyrin. Protecting block ciphers against differential fault attacks without re-keying. In *HOST*, pages 191–194. IEEE Computer Society, 2018.

[BBKN12] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proc. IEEE*, 100(11):3056–3076, 2012.

[BCN+06] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. *Proc. IEEE*, 94(2):370–382, 2006.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

[BGN12] Céline Blondeau, Benoît Gérard, and Kaisa Nyberg. Multiple differential cryptanalysis using LLR and χ^2 statistics. In Ivan Visconti and Roberto De Prisco, editors, *Security and Cryptography for Networks – SCN 2012*, volume 7485 of *LNCS*, pages 343–360. Springer, 2012.

[BGV11] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An in-depth and black-box characterization of the effects of clock glitches on 8-bit mcus. In *FDTC*, pages 105–114. IEEE Computer Society, 2011.

[BHL18] Jakub Breier, Xiaolu Hou, and Yang Liu. Fault attacks made easy: Differential fault analysis automation on assembly code. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):96–122, 2018.

[BJC15] Jakub Breier, Dirmanto Jap, and Chien-Ning Chen. Laser profiling for the back-side fault attacks: With a practical laser skip instruction attack on AES. In *CPSS@ASIACSS*, pages 99–103. ACM, 2015.

[BK06] Johannes Blömer and Volker Krummel. Fault based collision attacks on AES. In *FDTC*, volume 4236 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2006.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[CB19] Andrea Caforio and Subhadeep Banik. A study of persistent fault analysis. In Shivam Bhasin, Avi Mendelson, and Mridul Nandi, editors, *Security, Privacy, and Applied Cryptography Engineering - 9th International Conference, SPACE 2019, Gandhinagar, India, December 3-7, 2019, Proceedings*, volume 11947 of *Lecture Notes in Computer Science*, pages 13–33. Springer, 2019.

[CGR20] Sébastien Carré, Sylvain Guilley, and Olivier Rioul. Persistent fault analysis with few encryptions. In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design - 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1-3, 2020, Revised Selected Papers*, volume 12244 of *Lecture Notes in Computer Science*, pages 3–24. Springer, 2020.

[Cla07] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 181–194. Springer, 2007.

[DCAM15] Prakash Dey, Abhishek Chakraborty, Avishek Adhikari, and Debdeep Mukhopadhyay. Improved practical differential fault analysis of grain-128. In *DATE*, pages 459–464. ACM, 2015.

[DDR+12] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, P. Orsatelli, Philippe Maurine, and Assia Tria. Injection of transient faults using electromagnetic pulses -practical results on a cryptographic system-. *IACR Cryptol. ePrint Arch.*, page 123, 2012.

[DDRT12] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In *FDTC*, pages 7–15. IEEE Computer Society, 2012.

[DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: Exploiting ineffective fault inductions on symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):547–572, 2018.

[DFL11] Patrick Derbez, Pierre-Alain Fouque, and Delphine Leresteux. Meet-in-the-middle and impossible differential fault analysis on AES. In *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 274–291. Springer, 2011.

[EHH+14] Sho Endo, Naofumi Homma, Yu-ichi Hayashi, Junko Takahashi, Hitoshi Fuji, and Takafumi Aoki. A multiple-fault injection attack by adaptive timing control under black-box conditions and a countermeasure. In *COSADE*, volume 8622 of *Lecture Notes in Computer Science*, pages 214–228. Springer, 2014.

[ESP20] Susanne Engels, Falk Schellenberg, and Christof Paar. SPFA: SFA on multiple persistent faults. In *17th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2020, Milan, Italy, September 13, 2020*, pages 49–56. IEEE, 2020.

[FCL+20] Jingyi Feng, Hua Chen, Yang Li, Zhipeng Jiao, and Wei Xi. A framework for evaluation and analysis on infection countermeasures against fault attacks. *IEEE Trans. Inf. Forensics Secur.*, 15:391–406, 2020.

[FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on AES with faulty ciphertexts only. In Wieland Fischer and Jörn-Marc Schmidt, editors, *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*, pages 108–118. IEEE Computer Society, 2013.

[FXKH17] Kazuhide Fukushima, Rui Xu, Shinsaku Kiyomoto, and Naofumi Homma. Fault injection attack on salsa20 and chacha and a lightweight countermeasure. In *TrustCom/BigDataSE/ICESS*, pages 1032–1037. IEEE Computer Society, 2017.

[GPT19] Michael Gruber, Matthias Probst, and Michael Tempelmeier. Persistent fault analysis of ocb, DEOXYS and COLM. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2019, Atlanta, GA, USA, August 24, 2019*, pages 17–24. IEEE, 2019.

[GST12] Benedikt Gierlichs, Jörn-Marc Schmidt, and Michael Tunstall. Infective computation and dummy rounds: Fault protection for block ciphers without check-before-output. In *LATINCRYPT*, volume 7533 of *Lecture Notes in Computer Science*, pages 305–321. Springer, 2012.

[GYTS14] Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont. Differential fault intensity analysis. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 49–58. IEEE, 2014.

[HBZL19] Xiaolu Hou, Jakub Breier, Fuyuan Zhang, and Yang Liu. Fully automated differential fault analysis on software implementations of block ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):1–29, 2019.

[KH14] Thomas Korak and Michael Hoefler. On the effects of clock and power supply tampering on two microcontroller platforms. In *FDTC*, pages 8–17. IEEE Computer Society, 2014.

[LSG+10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2010.

[MDH+13a] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller. In *FDTC*, pages 77–88. IEEE Computer Society, 2013.

[MDH+13b] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller. In *FDTC*, pages 77–88. IEEE Computer Society, 2013.

[PM19] Sikhar Patranabis and Debdeep Mukhopadhyay. *Idempotent Instructions to Counter Fault Analysis Attacks*, pages 195–208. Springer International Publishing, Cham, 2019.

[PP21] Peter Pessl and Lukas Prokop. Fault attacks on cca-secure lattice kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):37–60, 2021.

[PZRB19] Jingyu Pan, Fan Zhang, Kui Ren, and Shivam Bhasin. One fault is all it needs: Breaking higher-order masking with persistent fault analysis. In Jürgen Teich and Franco Fummi, editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019*, pages 1–6. IEEE, 2019.

[RCC+16] Debapriya Basu Roy, Avik Chakraborti, Donghoon Chang, S. V. Dilip Kumar, Debdeep Mukhopadhyay, and Mridul Nandi. Fault based almost universal forgeries on CLOC and SILC. In *SPACE*, volume 10076 of *Lecture Notes in Computer Science*, pages 66–86. Springer, 2016.

[SBH+21] Hadi Soleimany, Nasour Bagheri, Hosein Hadipour, Prasanna Ravi, Shivam Bhasin, and Sara Mansouri. Practical multiple persistent faults analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):1–24, 2021.

[SBHS15] Bodo Selmke, Stefan Brummer, Johann Heyszl, and Georg Sigl. Precise laser fault injections into 90 nm and 45 nm sram-cells. In *CARDIS*, volume 9514 of *Lecture Notes in Computer Science*, pages 193–205. Springer, 2015.

[SBR+20] Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay. Fault template attacks on block ciphers exploiting fault propagation. In *EUROCRYPT (1)*, volume 12105 of *Lecture Notes in Computer Science*, pages 612–643. Springer, 2020.

[Sel08] Ali Aydin Selçuk. On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, 2008.

[SH08] Jörn-Marc Schmidt and Christoph Herbst. A practical fault attack on square and multiply. In *FDTC*, pages 53–58. IEEE Computer Society, 2008.

[SHP09] Jörn-Marc Schmidt, Michael Hutter, and Thomas Plos. Optical fault attacks on AES: A threat in violet. In Luca Breveglieri, Israel Koren, David Naccache, Elisabeth Oswald, and Jean-Pierre Seifert, editors, *Sixth International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2009, Lausanne, Switzerland, 6 September 2009*, pages 13–22. IEEE Computer Society, 2009.

[SHS16] Bodo Selmke, Johann Heyszl, and Georg Sigl. Attack on a DFA protected AES by simultaneous laser fault injections. In *FDTC*, pages 36–46. IEEE Computer Society, 2016.

[SMC09] Dhiman Saha, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. A diagonal fault attack on the advanced encryption standard. *IACR Cryptol. ePrint Arch.*, page 581, 2009.

[SMD18] Sayandeep Saha, Debdeep Mukhopadhyay, and Pallab Dasgupta. Expfault: An automated framework for exploitable fault characterization in block ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):242–276, 2018.

[TBM14] Harshal Tupsamudre, Shikha Bisht, and Debdeep Mukhopadhyay. Destroying fault invariant with randomization - A countermeasure for AES against differential fault attacks. In *CHES*, volume 8731 of *Lecture Notes in Computer Science*, pages 93–111. Springer, 2014.

[TK10] Elena Trichina and Roman Korkikyan. Multi fault laser attacks on protected CRT-RSA. In *FDTC*, pages 75–86. IEEE Computer Society, 2010.

[VZB+22] Navid Vafaei, Sara Zarei, Nasour Bagheri, Maria Eichlseder, Robert Primas, and Hadi Soleimany. Statistical effective fault attacks: The other side of the coin. *IEEE Trans. Inf. Forensics Secur.*, 17:1855–1867, 2022.

[YGS15] Bilgiday Yuce, Nahid Farhady Ghalaty, and Patrick Schaumont. Improving fault attacks on embedded software using RISC pipeline characterization. In *FDTC*, pages 97–108. IEEE Computer Society, 2015.

[YGS+16] Bilgiday Yuce, Nahid Farhady Ghalaty, Harika Santapuri, Chinmay Deshpande, Conor Patrick, and Patrick Schaumont. Software fault resistance is futile: Effective single-glitch attacks. In *FDTC*, pages 47–58. IEEE Computer Society, 2016.

[YJ00] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Transactions on computers*, 49(9):967–970, 2000.

[ZGZ+13] Xinjie Zhao, Shize Guo, Fan Zhang, Zhijie Shi, Chujiao Ma, and Tao Wang. Improving and evaluating differential fault analysis on LED with algebraic techniques. In *FDTC*, pages 41–51. IEEE Computer Society, 2013.

[ZGZ+16] Fan Zhang, Shize Guo, Xinjie Zhao, Tao Wang, Jian Yang, François-Xavier Standaert, and Dawu Gu. A framework for the analysis and evaluation of algebraic fault attacks on lightweight block ciphers. *IEEE Trans. Inf. Forensics Secur.*, 11(5):1039–1054, 2016.

[ZLZ+18] Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren. Persistent fault analysis on block ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):150–172, 2018.

[ZZG+13] Fan Zhang, Xinjie Zhao, Shize Guo, Tao Wang, and Zhijie Shi. Improved algebraic fault analysis: A case study on piccolo and applications to other lightweight block ciphers. In *COSADE*, volume 7864 of *Lecture Notes in Computer Science*, pages 62–79. Springer, 2013.

[ZZJ+20] Fan Zhang, Yiran Zhang, Huilong Jiang, Xiang Zhu, Shivam Bhasin, Xinjie Zhao, Zhe Liu, Dawu Gu, and Kui Ren. Persistent fault attack in practice. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):172–195, 2020.