The ISC Int'l Journal of Information Security
ISeCure

October 2025, Volume 17, Number 3 (pp. 1–10)
http://www.isecure-journal.org

PRESENTED AT THE ISCISC'2025 IN TEHRAN, IRAN.

Linked Ineffective Fault Analysis on DES Cipher **

Vahid Soleimani Hesari [1], Hadi Soleimany [1,*], Ali Asghar Beigizadi Mazandarani [1], and Hamed Ramzanipour [2]

[1] *Cyber Research Center, Shahid Beheshti University (SBU), Tehran, Iran*
[2] *Department of Communication, Faculty of Electrical Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran*

ARTICLE INFO.

Keywords:
IoFault Analysis; Linked Fault Analysis; Feistel Block Cipher; DES; LIFA

Type:

doi:

Abstract

Linked Ineffective Fault Analysis (LIFA) is a novel fault analysis technique that operates without requiring input control and demonstrates resilience against noise compared to Statistical Ineffective Fault Analysis (SIFA), while maintaining similar attack assumptions. However, prior studies on LIFA have focused primarily on SPN block ciphers, leaving the security of the DES cipher one of the Feistel ciphers unexplored. Furthermore, the application of LIFA in the presence of multiple faults remains unaddressed. This paper bridges these gaps by applying LIFA to the widely utilized DES cipher, aiming to evaluate the effectiveness of this attack on Feistel-based structures. We effectively apply LIFA across various scenarios and demonstrate the feasibility of inducing multiple linked faults. Our results reveal that the nibble-based structure of DES allows for the establishment of two simultaneous links instead of one, significantly enhancing the efficacy of fault attacks on DES. To validate our approach, we conducted both simulations and real-world experiments using frequency glitch fault injection on an ATMEGA328p microcontroller. The results show that the proposed LIFA framework for the DES cipher achieves superior performance compared to existing methods such as SIFA, further advancing the state of cryptographic fault analysis.

1 Introduction

Fault attacks (FAs) constitute a class of physical attacks where adversaries deliberately induce faults in targeted devices to compromise their operation. These attacks pose a significant risk to the security of cryptographic systems. Fault induction can be achieved through various techniques, such as clock or voltage glitches, electromagnetic (EM) pulses, laser irradiation, and similar methods. The concept of fault attacks was first introduced by Boneh *et al.* in 1996 [1]. Since the inception of fault attacks, numerous techniques have been devised to exploit them against cryptographic algorithms. These attacks are broadly categorised into three distinct groups.

Persistent fault attacks (PFAs) represent the first category [2, 3], characterized by faults that remain in effect until the target device is reset. Since their introduction at CHES 2018, PFAs have received significant attention and

* Corresponding author.
**The ISCISC'2025 program committee effort is highly acknowledged for reviewing this paper.
Email addresses: v.soleimanihesari@mail.sbu.ac.ir, h_soleimany@sbu.ac.ir, beigizad@yahoo.com, ramzanipourh@gmail.com
ISSN: 2008-2045 © 2025 ISC. All rights reserved.

have been applied to software implementations of various cryptographic primitives [4–7].

The second category encompasses fault attacks that induce permanent modifications to the target device. The third category comprises transient fault attacks that cause temporary disruptions to the target device. This paper specifically focuses on transient faults. Numerous methods based on transient faults have been proposed, including DFA [8], IDFA [9], FSA [10], DFIA [11], FTA [12], and other similar techniques.

Transient fault attacks are unified by a common characteristic: the injected fault alters an intermediate value, transforming it into a new, faulty value.

A novel fault attack, termed Linked Fault Analysis (LFA), was recently introduced in [13]. Within this framework, a variant known as Linked Ineffective Fault Analysis (LIFA) focuses exclusively on ineffective fault occurrences. LIFA holds a distinct advantage over certain other fault attacks, as it neither requires input control nor is hindered by redundancy-based countermeasures. However, most prior studies on ineffective fault analysis have been restricted to Substitution-Permutation Network (SPN) ciphers, leaving Feistel ciphers unexplored. Furthermore, existing works typically assume a single-fault model and ignore realistic scenarios where **missed faults** (faults that do not alter the ciphertext) and **unwanted faults** (extra unintended faults) occur. These assumptions limit the applicability of previous research to real-world implementations, where multiple simultaneous faults are common in practice. As cryptographic algorithms are increasingly deployed in resource-constrained IoT and embedded environments, practical fault models must address these non-ideal conditions.

1.1 Our Contribution

In this work, we study Linked Ineffective Fault Analysis (LIFA) on Feistel ciphers by focusing on DES. While LIFA has already been introduced for SPN ciphers, its application to DES has not been addressed before. The Feistel structure makes the attack more challenging because faults injected in one half of the block can be masked by the XOR operation, which hides the statistical patterns that LIFA needs. Showing how LIFA can still be applied to DES is one of the main contributions of this paper.

We also extend the study of LIFA to more realistic conditions where multiple faults may occur at the same time. in implementation, fault injection often produces missed faults, where the output does not change, and unwanted faults, where extra faults appear unexpectedly. These cases were not considered in earlier works. We show that such scenarios can be handled in our framework and even used to improve the effectiveness of the attack.

Finally, we support our study with experiments on an ATmega328p microcontroller using frequency glitching, along with simulation results. The experiments confirm that LIFA can be applied to DES in practice and show that it can achieve better results compared to SIFA [14]. These findings demonstrate that LIFA is a practical and effective approach for fault analysis of Feistel ciphers.

1.2 Outline

The structure of the remainder of this paper is as follows: Section 2 Offers an overview of Feistel ciphers and one of them and Linked Fault Analysis (LFA). In Section 3, our proposed attack methodology is described in detail applied to the DES cipher in various scenarios. We present our experimental results in Section 4. Finally, we conclude in Section 5.

2 Preliminaries

This section establishes the foundational concepts, notations, and cryptographic structures used throughout the paper. The focus is primarily on the Feistel cipher structure and the DES (Data Encryption Standard) algorithm, as well as the fault analysis techniques applied to them.

2.1 Notation

To establish clarity, this section outlines the notations and parameters adopted throughout the paper. The length of the block in DES cipher is represented by b, whereas k indicates the length of the master key. Notations such as K, C, and M correspond to the master key, ciphertext, and plaintext, respectively. Denoted by n, the total number of rounds is a critical parameter, with r specifying the round number for $1 \leq r \leq 16$. The variable l is used to signify the number of S-boxes involved in the round function. Intermediate values in Feistel rounds are represented by X_L and X_R, which refer to the left and right sides, respectively. The round function, identified as F_r, operates alongside the substitution box (S-box), denoted by S, which constitutes the nonlinear layer. Each S-box takes an input of length m.

To describe the Feistel round function F_r, the input and output are denoted as $X_r[i]$ and $Y_r[i]$, respectively. The output of the nonlinear S-box layer is symbolised by z_r. For $1 \leq i \leq l$, the i-the S-box in the r-th round receives input $y_r[i]$ and produces output $z_r[i]$. The terms I and O denoted general inputs and outputs. Each round-specific subkey is represented by K_r.

These notations provide a consistent framework for describing Feistel ciphers, facilitating the methodologies discussed in this paper.

2.2 Feistel Ciphers

Figure 1 depicts the structure of a b-bit Feistel cipher. In this design, the round function processes the input X_r^L us-

ing the subkey sk_r, producing an output that is XORed with X_r^R to generate $X_{(r+1)}^L$. Simultaneously, X_r^L is directly passed unchanged to become $X_{(r+1)}^R$. The round function F_r consists of three operational layers. The first layer, $(F_{r,I})$, serves as a linear input layer. This is followed by a nonlinear layer composed of S-box, labeled as $S_1, \ldots, S_l$, which map inputs $y_r[i]$ to outputs $z_r[i]$. Finally, the output layer, denoted as $(F_{r,O})$, completes the process without imposing specific linearity constraints. This architectural design enhances the cipher's resilience against cryptographic attacks while ensuring robust and efficient data encryption.

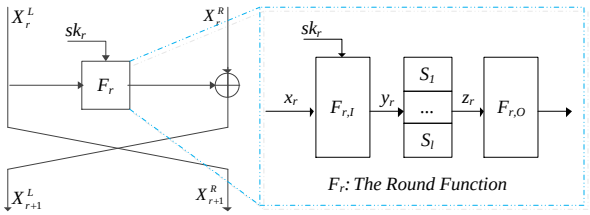

Figure 1. Generalized round of a Feistel cipher

2.3 Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a 64-bit Feistel cipher comprising 16 rounds. The encryption process begins with an initial permutation applied to the 64-bit input and concludes with an inverse permutation after the final round. The round function within DES is composed of four distinct steps:

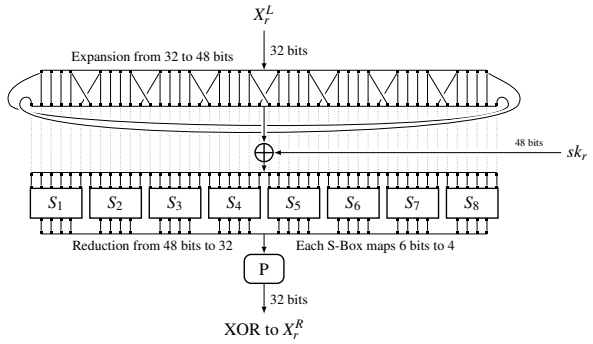

Figure 2. Inner functiion of DES

(1) Expansion: The 32-bit left half of the input block is expanded to 48 bits during this step. This expansion is performed using a predefined expansion table, commonly referred to as the E box.
(2) Round Key: The expanded 48-bit input is XORed with the round key using bitwise operations. These round keys are generated through a key schedule that applies a series of shifts and permutations to the original key, ensuring unique keys for each encryption round.
(3) Substitution: Following the Round key step, the 48 bits are divided into eight groups, each containing six bits. Each group is processed individually by a unique S-box, which replaces the six input bits with four output bits based on a nonlinear transformation table.
(4) Permutation: The 32-bit output from the substitution step is rearranged according to a predefined permutation table, known as the P-box, completing the round function.

2.4 Statistical Ineffective Fault analysis

Statistical Fault Attack (SFA) leverages a biased distribution of faulty intermediate values to compromise the security of cryptographic systems. In this attack, the adversary decrypts a set of faulty ciphertexts from the final round(s) of the cipher using key candidates. To evaluate how closely the resulting distributions align with the expected biased distribution, a statistical scoring function is computed for each key candidate.

When redundancy-based countermeasures [15] are employed, the attacker cannot access faulty ciphertexts in cases where the fault is effective. However, in scenarios involving ineffective faults, the ciphertexts remain observable. Exploiting this, Statistical Ineffective Fault Analysis (SIFA) relies on the biased distribution of intermediate values caused by ineffective faults to bypass these countermeasures. A more recent approach, Statistical Effective Fault Attack (SEFA [16]), adopts similar statistical techniques but focuses on effective faults.

$$SEI(k) = \sum_{x \in X} (\hat{p}_k(x) - \theta(x))^2 \tag{1}$$

where X denotes the set of possible intermediate values, $\theta(x)$ is the uniform distribution over X, and $\hat{p}_k(x)$ represents the estimated probability distribution of the intermediate value x under the key hypothesis k. This scoring function measures the deviation of the observed distribution from the ideal uniform distribution. The number of required ciphertexts to acquire the correct key can be estimated based on Equation 2.

$$N_{SEI} \approx \frac{\beta \cdot \phi_{0,1}^{-1}(\alpha)}{C(p,\theta)} \tag{2}$$

where α is the significance level of the test, β denotes the statistical confidence parameter, and $\phi_{0,1}^{-1}(\cdot)$ is the inverse of the cumulative distribution function of the standard normal distribution. The denominator $C(p,\theta)$ is a measure of distance between the empirical and uniform distributions, defined as:

$$C(p,\theta) = \sum_{x \in X} \frac{(\hat{p}_k(x) - \theta(x))^2}{\theta(x)} \tag{3}$$

The probability of an ineffective event is denoted by the ineffectivity rate Π_i. Hence, the total number of required ciphertexts for SIFA is finally given as:

$$N = \frac{N_{SEI}}{\Pi_i}.$$

Algorithm 1 Key-recovery over last round of DES without noise for SIFA

Require:
Ineffective (faulty) ciphertexts $C_1, C_2, \ldots, C_N$
Key candidates $\mathscr{K}$ for $sk_{16}[i]$ (2^6 candidates, i.e. $\mathscr{K} = \{0, 1, ..., 2^6 - 1\}$).
Ensure: Correct key
for $\ell = 0$ to $2^6 - 1$ **do**
  $(sk_{16}[i]) \leftarrow \ell$
  for $h = 1$ to N **do**
    $X_{16}^L \leftarrow C_h^L$
    $z_{16}[i] \leftarrow S_i(X_{16}[i] \oplus sk_{16}[i])$
    calculate SEI
  end for
end for
return $argmax_{SEI}(Correctkey[MaxSEI])$
return $\mathscr{K}$

2.5 Linked Ineffective Fault Analysis (LIFA)

Linked Fault Analysis (LFA) is a novel type of fault attack that exploits the relationship between two intermediate values resulting from a single fault. In this technique, a second variable, v, is linked to the faulty target variable, u', through a specific relationship, such as equality, expressed as $u' = l(v)$. This linkage allows the attacker to uncover critical key bits necessary for decryption.

Two scenarios arise depending on the presence or absence of redundancy-based countermeasures:

In the absence of redundancy-based countermeasures, the attacker guesses the involved subkey bits and decrypts N faulty ciphertexts C'_i, where $1 \leq i \leq N$, to compute two m-bit intermediate values, u' and v. The relationship $u' = l(v)$ always holds for the correct key but only holds for an incorrect key with a probability of 2^{-m}. If there are 2^κ candidates for the κ-bit guessed key, approximately $N = \frac{\kappa}{m}$ faulty ciphertexts are sufficient for key recovery. This method is referred to as Linked Differential Fault Analysis (LDFA) in [13].

In the presence of redundancy-based countermeasures, the attacker instead focuses on ineffective fault events. By applying the same techniques as described earlier, the key can still be retrieved. This variation of the method is called Linked Ineffective Fault Analysis (LIFA) in [13].

The attacker exploits the cipher's operational routines—specifically the loading of S-box outputs into RAM or flash memory during the encryption round. By skipping crucial load instructions [17], either those selecting data addresses or those writing data into memory, the attacker can induce the desired fault. Similar to most fault attacks, a key assumption while conducting an LFA is control over fault intensity and duration. It means that attacker must be able to precisely manage the intensity and duration of the fault injected, which directly influences the success rate of changing the intermediate values.

3 LIFA on DES

Although the application of LFA to SPN-based ciphers has been explored in prior studies, the original proposal did not address the security of Feistel ciphers under LFA. In this section, we focus on presenting LIFA in the context of DES.

3.1 Overview of the Proposed Attack

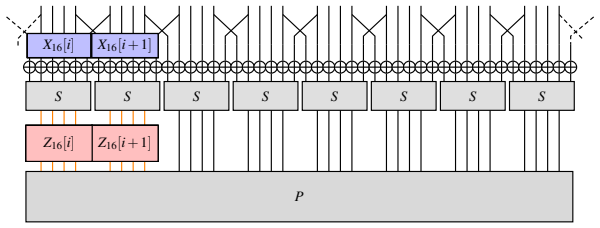

Figure 3. Linked fault in the final round of DES

We assume that the implementation incorporates a redundancy-based countermeasure. Specifically, the encryption process is executed twice with the same secret key in the DES framework. If the two results differ, the system either blocks the ciphertexts or produces them randomly as a safeguard.

We focus on the final round of DES, where $y_{16}[i]$ and $y_{16}[i+1]$ are two consecutive state variables processed sequentially. These variables are defined as $y_{16}[i] = x_{16}[i] \oplus sk_{16}[i]$ and $y_{16}[i+1] = x_{16}[i+1] \oplus sk_{16}[i+1]$ (refer to section 2.1). A fault is introduced during the S-box evaluation of $y_{16}[i+1]$. Under certain conditions, the resulting faulty value $z'_{16}[i+1] = S(y_{16}[i+1])$ coincides with $S(y_{16}[i])$. Such a fault can occur due to an instruction skip. Consequently, after the fault, $z'_{16}[i+1]$ becomes correlated with $z_{16}[i]$. If the attacker determines that the fault is ineffective, they can infer $z_{16}[i] = z_{16}[i+1]$. Hence, it's easy to observe that the following relationship exists between the part of the subkey of the last round:

$$S_{i+1}(x_{16}[i+1] \oplus sk_{16}[i+1]) = S_i(x_{16}[i] \oplus sk_{16}[i]) \quad (4)$$

Figure 3 illustrates this observation for the final round of DES.

3.2 LIFA on DES Without Noise

According to Algorithm 1 the intermediate values $z_{16}[i]$ and $z_{16}[i+1]$, as well the subkey bits $sk_{16}[i, i+1]$, influence a portion of the ciphertext. In other words, the intermediate values $z_{16}[i]$ and $z_{16}[i+1]$ can be determined by guessing the value of the subkey bits $sk_{16}[i, i+1]$ and decrypting the ciphertext C. There are $\tau = 2^{12}$ candidates

Algorithm 2 Key-recovery over last round of DES without noise

Require:
Ineffective (faulty) ciphertexts $C_1, C_2, \ldots, C_N$
Key candidates $\mathscr{K}$ for $sk_{16}[i, i+1]$ (2^{12} candidates, i.e. $\mathscr{K} = \{0, 1, ..., 2^{12}-1\}$).
Ensure: Correct key
for $\ell = 0$ to $2^{12}-1$ **do**
  $(sk_{16}[i], sk_{16}[i+1]) \leftarrow \ell$
  for $h = 1$ to N **do**
    $X_{16}^L \leftarrow C_h^L$
    $z_{16}[i] \leftarrow S_i(X_{16}[i] \oplus sk_{16}[i])$
    $z_{16}[i+1] \leftarrow S_{i+1}(X_{16}[i] \oplus sk_{16}[i+1])$
    if $z_{16}[i] \neq z_{16}[i+1]$ **then**
      Eliminate ℓ, i.e. $\mathscr{K} = \mathscr{K} \setminus \{\ell\}$
    end if
  end for
end for
return $\mathscr{K}$

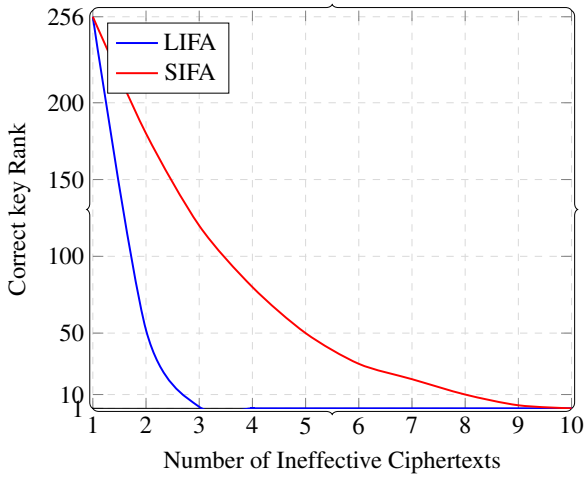

Figure 4. Effectiveness of LIFA and SIFA Attacks without noise

for the target key bits (i.e. $sk_{16}[i, i+1]$). Given N ineffective ciphertexts $C_1, C_2, ..., C_N$, we can partially decrypt the ciphertexts for a guessed key and check whether the relation $z_{16}[i] = z_{16}[i+1]$ holds or not. The relation holds for the correct key in all cases, but with a probability of 2^{-4} for a wrong key. In other words, if the relation does not hold, the guessed key is wrong and can be eliminated from the key candidates. This technique is illustrated in Algorithm 2. Given N ineffective ciphertets, $2^{12} \times 2^{-4 \cdot N}$ candidates remain. To determine the correct key uniquely, around $N = 3$ ineffective ciphertexts are needed. Since the ineffective rate is $\Pi_{\text{ineff}} = 2^{-4}$, in total almost $3 \times 2^4 = 48$ faulty computations are required to retrieve the key. To validate the proposed approximation, we conducted simulations of the LIFA attack on DES. The results, presented in Figure 4, demonstrate that, in implementation, only three ineffective ciphertexts are sufficient to recover the target keys.

3.3 In the Presence of Missed and Unwanted Faults

Thus far, we have assumed an ideal scenario where no fault events are missed, and the attack execution setup functions flawlessly. However, in implementation, achieving perfect fault injection is rarely feasible. Faults may fail to inject as intended for two primary reasons. In this section, we analyse the LIFA attack on DES under various practical scenarios.

(1) **Missed faults:** A missed fault occurs when an instruction skip fails to execute following fault injection. Even with a highly reliable and precise setup, missed faults are unavoidable. As they are difficult to distinguish from ineffective faults, missed faults can significantly impact fault attacks that depend on ineffective events.
(2) **Unwanted faults:** Unwanted faults occur when an injected fault takes place at an unintended time or location. These faults can result from incidental noise during experiments.

Most fault attacks are affected by both missed and unwanted faults, as these faults are often indistinguishable from the intended ones, particularly when attackers lack input control to repeat the experiment with a fixed input.

Let Π_m and Π_u represent the rates of missed faults and unwanted faults, respectively. For N experiments, approximately $N \times (1 - (\Pi_m + \Pi_u))$ cases will uphold the relation $z_{16}[i] = z_{16}[i+1]$. In the remaining $N \times (\Pi_m + \Pi_u)$ cases, where no fault occurs or the fault happens in an incorrect location, the relation $z_{16}[i] = z_{16}[i+1]$ holds with an average probability of 2^{-4}. Unlike the scenario in Section 3.2, here $z_{16}[i] = z_{16}[i+1]$ is not deterministic.

$$\Pr(z_{16}[i] = z_{16}[i+1]) = \begin{cases} (1 - (\Pi_m + \Pi_u)) + 2^{-4} \cdot (\Pi_m + \Pi_u) & \text{Correct key} \\ 2^{-4} & \text{Wrong key.} \end{cases} \quad (5)$$

However, as discussed before and it's shown in Equation 5, the probability that $z_{16}[i] = z_{16}[i+1]$ holds differs between the correct key and an incorrect key.

To successfully implement the attack in the presence of missed and unwanted faults, Algorithm 2 can be adapted with minor modifications. As previously noted, the equation $z_{16}[i] = z_{16}[i+1]$ does not always hold for the correct key. However, the probability of this relationship being true is higher for the correct key than for an incorrect key guess. By employing a probabilistic approach, as described in Algorithm 3, and increasing the number of required ciphertexts, the key can still be recovered. In the following, we analyse various scenarios to determine the amount of data needed to uniquely recover the key.

We executed Algorithm 3 under three distinct conditions, corresponding to missed fault rates of 30%, 50%,

Algorithm 3 Key-recovery over last round of DES with missed fault

Require:
Ineffective (faulty) ciphertexts $C_1, C_2, \ldots, C_N$
Key candidates $sk_{16}[i, i+1]$ (2^{12} candidates).
Ensure: Correct key
for $\ell = 0$ to $2^{12} - 1$ **do**
  $(sk_{16}[i], sk_{16}[i+1]) \leftarrow \ell$
  for $h = 1$ to N **do**
    $X_{16}^L \leftarrow C_h^L$
    $z_{16}[i] \leftarrow S_i(X_{16}[i] \oplus sk_{16}[i])$
    $z_{16}[i+1] \leftarrow S_{i+1}(X_{16}[i] \oplus sk_{16}[i+1])$
    if $z_{16}[i] = z_{16}[i+1]$ **then**
      $cnt[\ell] = cnt[\ell] + 1$
    end if
  end for
end for
return $argmax_\ell(cnt[\ell])$

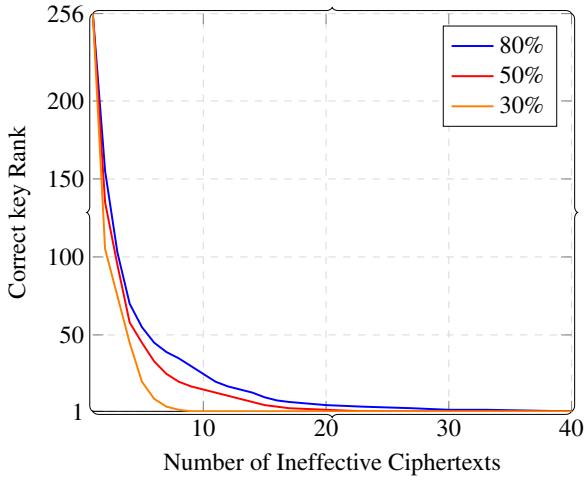

Figure 5. LIFA on DES in the presence of missed faults in three scenarios with Π_m=30%, 50% and 80%

and 80%, respectively. The results are presented in Figure 5. As expected, higher missed fault rates require a greater number of ineffective ciphertexts to successfully recover the key. Nevertheless, as shown in Figure 5, the number of required ciphertexts remains relatively low, ensuring the practical feasibility of the attack.

Figure 6 summarises the three missed fault scenarios, including numerical data and percentages, to facilitate comparison and understanding of the outcomes. This information provides the average probabilities of accurately identifying the key for each of the five given values, based on 100 testing iterations. Each session was evaluated under different percentages of missed faults.

3.4 LIFA vs SIFA

To the best of our knowledge, Feistel-based ciphers have not been evaluated against SIFA thus far. Moreover, it's a

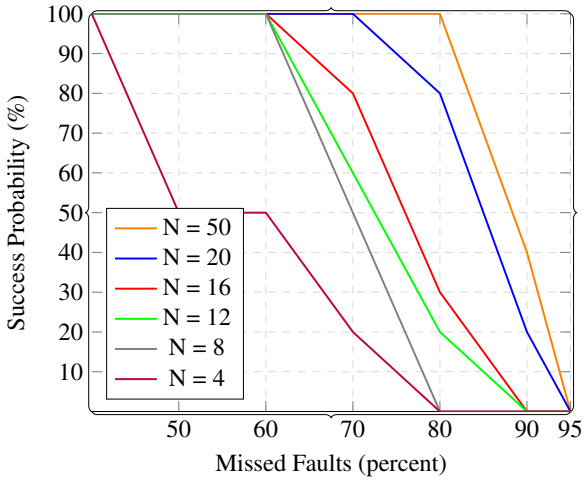

Figure 6. Success Probability of Key Identification Across Different Missed Fault Rates

valuable perspective to determine which approach, SIFA or LIFA, offers greater advantages to an attacker. However, we have not encountered a comprehensive study addressing this comparison. We used the same CPU and identical settings to model SIFA and LIFA concurrently, ensuring a fair evaluation. As illustrated in Table 1, when a single

Algorithm 4 Key-recovery over last round of DES in case of multiple faults

Require: Ineffective (faulty) ciphertexts $C_1, C_2, \ldots, C_N$, Key candidates $sk_{16}[i, i+1]$ (2^{12} candidates), and Key candidates $sk_{16}[i+2, i+3]$ (2^{12} candidates).
Ensure: Correct key
for $\ell = 0$ to $2^{12} - 1$ **do**
  $(sk_{16}[i], sk_{16}[i+1]) \leftarrow \ell$
  for $h = 1$ to N **do**
    $X_{16}^L \leftarrow C_h^L$
    $z_{16}[i] \leftarrow S_i(X_{16}[i] \oplus sk_{16}[i])$
    $z_{16}[i+1] \leftarrow S_{i+1}(X_{16}[i] \oplus sk_{16}[i+1])$
    if $z_{16}[i] = z_{16}[i+1]$ **then**
      $cnt[\ell] = cnt[\ell] + 1$
    end if
  end for
end for
return $argmax_\ell(cnt[\ell])$
for $\ell = 0$ to $2^{12} - 1$ **do**
  $(sk_{16}[i+2], sk_{16}[i+3]) \leftarrow \ell$
  for $h = 1$ to N **do**
    $X_{16}^L \leftarrow C_h^L$
    $z_{16}[i+2] \leftarrow S_i(X_{16}[i+2] \oplus sk_{16}[i+2])$
    $z_{16}[i+3] \leftarrow S_{i+3}(X_{16}[i+2] \oplus sk_{16}[i+3])$
    if $z_{16}[i+2] = z_{16}[i+3]$ **then**
      $cnt[\ell] = cnt[\ell] + 1$
    end if
  end for
end for
return $argmax_\ell(cnt[\ell])$

Table 1. Comparison of LIFA and SIFA

	# Required Faulty Computations				
	Ineffective Rate	Without any-noise	Missed Faults (30%)	Missed Faults (50%)	Missed Faults (80%)
LIFA	$6\% \simeq 2^{-4}$	$3 \times 16 = 48$	$10 \times 16 = 160$	$22 \times 16 = 352$	$45 \times 16 = 720$
LIFA†	$0.4\% \simeq 2^{-8}$	$4 \times 256 = 1,024$	$12 \times 256 = 3,072$	$26 \times 256 = 6,656$	$50 \times 256 = 12,800$
SIFA	8%	$10 \times 16 = 160$	$21 \times 16 = 336$	$48 \times 16 = 768$	$94 \times 16 = 1,504$
SIFA‡	0.3%	$22 \times 256 = 5,632$	$31 \times 256 = 7,936$	$75 \times 256 = 19,200$	$91 \times 256 = 23,296$

† LIFA with double link.

‡ SIFA with double fault.

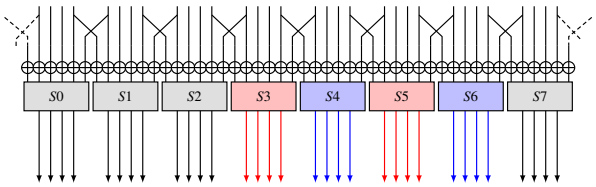

Figure 7. Location of links in experimental result

link and a single fault are applied simultaneously in both approaches, the data ratio for LIFA is approximately 1:4 relative to SIFA in a noise-free environment. This distinction becomes particularly significant when the ineffective data rates are nearly identical. Additionally, in low-data-rate scenarios, the double-link configuration plays a crucial role in improving the efficiency of key recovery calculations.

we assessed the impact of missed faults on the total data rate. Even with varying percentages of missed data, LIFA consistently exhibited superior performance and resilience.

3.5 Multiple Linked Faults

In the Data Encryption Standard (DES), the S-boxes are 6×4. If the fault injection process is capable of generating two linked faults simultaneously, the number of faulty computations required to recover the whole key is reduced by half, without altering the time complexity of the attack. Specifically, considering the architecture of the microcontroller and its implementation, if the fault affects the four words (z[i]), (z[i+1]), (z[i+2]), and (z[i+3]), additional correlations among these words can be established without incurring extra computational costs for the attack. As demonstrated in the subsequent section, our practical implementation indicates that a single fault injection can establish a link between (z[i]) and (z[i+2]), as well as between (z[i+1]) and (z[i+3]). This approach can be described as achieving *"two birds with one stone."* In essence, injecting a single fault results in the creation of multiple linked faults, as illustrated in Figure 7.

Consequently, as outlined in Algorithm 4, this method enables the attacker to independently recover 24 bits of the subkey sk_{16} in two distinct parts. Hence, this method was applied to both SIFA and LIFA simulation approaches. Table 1 demonstrates the use of multiple links and faults for key recovery. Although LIFA, in both single- and multiple-link configurations, retrieves more key bits, it requires fewer ineffective data compared to SIFA in both fault models, even in the presence of countermeasures.

4 Experimental Hardware Evaluation

This section presents a practical evaluation of the proposed fault analysis techniques on a physical hardware platform. We describe the experimental setup used to perform real-time fault injections and report the empirical findings obtained from applying LIFA and SIFA to the TWINE-80 cipher implemented on an embedded microcontroller. The results are analysed in terms of effectiveness, efficiency, and resilience under realistic operating conditions.

It's important to note that the improvements observed

in our results are not tied to the specific hardware platform or tools used. The Arduino Nano and FPGA setup only served as a convenient environment for implementing the experiments. The statistical bias exploited by LIFA arises from the linked ineffective fault model itself and the way the attack is structured, rather than from device-specific behaviour. We also observed that the attack remained effective across different operating frequencies and glitch parameters, which further supports that the results are a consequence of the proposed methodology and not of the underlying technology.

4.1 Experimental Setup

Our experimental framework was built upon a standardised configuration to ensure attacks were carried out effectively and precisely. The core components of the setup consisted of two embedded computing units: an Arduino Nano with an ATmega328P core and a Field-Programmable Gate Array (FPGA), each serving distinct but complementary roles in executing the attack. The ATmega328P was responsible for processing the cryptographic algorithm and generating a timing flag or trigger pulse, which was sent to the FPGA to indicate the precise moment when the fault injection should occur.

To control the target's clock signal, the onboard oscillator was replaced with an external clock source generated by the FPGA's internal Phase-Locked Loop (PLL), which acted as the clock glitching mechanism. Under normal conditions, the Arduino operates using a low-speed clock provided by the FPGA. When a glitch event is triggered, an interrupt is activated, causing the Arduino to temporarily switch to a high-speed clock for the glitch duration. After the glitch period ends, the same interrupt restores the Arduino to the low-speed reference clock. This setup ensures precise control of the glitch timing and frequency transitions, with the interrupt managing both the initiation and termination of the high-speed clock.

In this experiment, the glitch signal was uniquely elevated to 32 MHz, compared to the 8 MHz reference clock. This adjustment was achieved using two key parameters: the **start value**, which defines the fault's initiation point, and the **offset value**, which specifies the duration of the frequency increase. The FPGA configures the PLL using these tunable parameters. By adjusting the **start** and **offset** values, the attacker can precisely target and disrupt specific steps of the process, as these values are calibrated to match the effect of the glitch on the encryption runtime.

4.2 Practical Findings

Initially, we applied our setup to gather ineffective ciphertexts containing a single link in the intermediate values. Notably, our experimental results closely aligned with the simulation outcomes. As illustrated in Table 1, the number of ineffective exploitable data in a noise-free setup is four. This count remains consistent for identifying both a single link and repeated evaluations for discovering a double link. This approach enables us to recover 24 bits of the key while focusing solely on the *start* and **offset** parameters within the same setup.

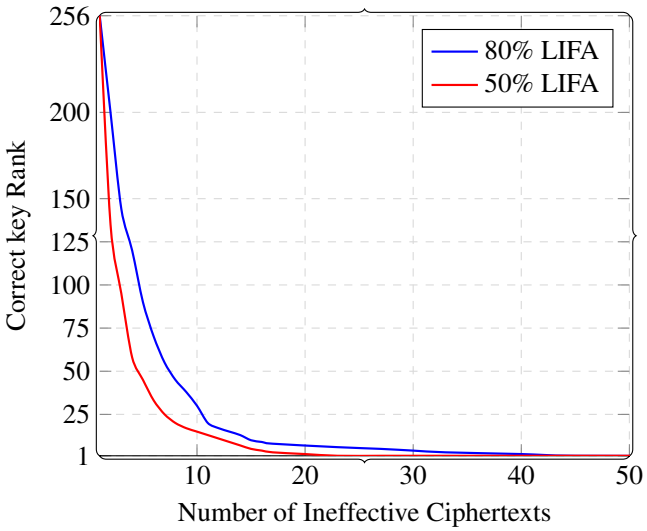

Figure 8. Missed Fault in Experimental Single Linked LIFA

Last but not least, let us examine this approach in relation to another unintended consequence of implementing fault attacks. In typical scenarios, missed faults can cause significant deviations in the collection of exploitable ineffective data, complicating key recovery. To address this, we accounted for this additive effect in our implementation and re-executed the investigation to gather valuable data.

Surprisingly, the results once again aligned closely with the simulation-based approaches. This consistency reinforces the validity of our theoretical concepts and supports their experimental confirmation. Figure 8 illustrates the LIFA attack based on single link under fault conditions where 50% and 80% of the faults are missed.

5 Conclusion

This study highlights the superior performance of Linked Ineffective Fault Analysis (LIFA) over Statistical Ineffective Fault Analysis (SIFA) for DES key recovery. LIFA requires only a very small number of ciphertexts to retrieve the correct key and remains effective even when fault injections are incomplete or imperfect. Our experiments on a simple and low-cost hardware setup confirm its practicality, while the software side benefits from lower computational effort compared to SIFA. Together, these results show that LIFA is both effective and realistic as a fault-based cryptographic analysis technique.

References

[1] 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.

[2] Andrea Caforio and Subhadeep Banik. A study of persistent fault analysis. In *Security, Privacy, and Applied Cryptography Engineering: 9th International Conference, SPACE 2019, Gandhinagar, India, December 3–7, 2019, Proceedings 9*, pages 13–33. Springer, 2019.

[3] Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren. Persistent fault analysis on block ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 150–172, 2018.

[4] 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.

[5] Susanne Engels, Falk Schellenberg, and Christof Paar. SPFA: SFA on Multiple Persistent Faults. In *17th Workshop on Fault Detection and Tolerance in Cryptography, FDTC*.

[6] Nasour Bagheri, Sadegh Sadeghi, Prasanna Ravi, Shivam Bhasin, and Hadi Soleimany. SIPFA: statistical ineffective persistent faults analysis on Feistel ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 367–390, 2022.

[7] Hadi Soleimany, Nasour Bagheri, Hosein Hadipour, Prasanna Ravi, Shivam Bhasin, and Sara Mansouri. Practical multiple persistent faults analysis. *Cryptology ePrint Archive*, 2021.

[8] 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 17*, pages 513–525. Springer, 1997.

[9] Wei Li, Wenwen Zhang, Dawu Gu, Yanqin Cao, Zhi Tao, Zhihong Zhou, Ya Liu, and Zhiqiang Liu. Impossible differential fault analysis on the LED lightweight cryptosystem in the vehicular ad-hoc networks. *IEEE Transactions on Dependable and Secure Computing*, 13(1):84–92, 2015.

[10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In *Cryptographic Hardware and Embedded Systems, CHES 2010: 12th International Workshop, Santa Barbara, USA, August 17-20, 2010. Proceedings 12*, pages 320–334. Springer, 2010.

[11] 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.

[12] Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay. Fault template attacks on block ciphers exploiting fault propagation. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 612–643. Springer, 2020.

[13] Ali Asghar Beigizad, Hadi Soleimany, Sara Zarei, and Hamed Ramzanipour. Linked fault analysis. *IEEE Transactions on Information Forensics and Security*, 19:632–645, 2024.

[14] 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*, pages 547–572, 2018.

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

[16] Navid Vafaei, Sara Zarei, Nasour Bagheri, Maria Eichlseder, Robert Primas, and Hadi Soleimany. Statistical effective fault attacks: the other side of the coin. *IEEE Transactions on Information Forensics and Security*, 17:1855–1867, 2022.

[17] Karine Heydemann, Nicolas Moro, Emmanuelle Encrenaz, and Bruno Robisson. Formal verification of a software countermeasure against instruction skip attacks. *IACR Cryptol. ePrint Arch.*, 2013:679, 2013.

Vahid Soleimani Hesari received his B.S. in Electrical Engineering from Arak University of Technology in 2022 and his M.S. in Electrical Engineering with a focus on Secure Telecommunications and Cryptography from Shahid Beheshti University in 2024. He is currently a researcher specializing in hardware security and the cryptanalysis of symmetric encryption algorithms, with particular interest in fault injection attacks, and side-channel analysis.

Hadi Soleimany has been serving as an Associate Professor at the Cyberspace Research Institute, Shahid Beheshti University, Iran, since 2015. He received his PhD in Theoretical Computer Science from Aalto University, Finland, in 2015. He also held postdoctoral research positions at the Technical University of Denmark (DTU) during the summers of 2016 and 2017: his primary research interests lie in the practical aspects of cryptography.

Ali Asghar Beigizad received his M.S. in Electrical Engineering with a focus on Secure Telecommunications and Cryptography from Shahid Beheshti University in 2022. He is currently a researcher specializing in hardware security with particular interest in fault injection attacks.

Hamed Ramzanipour received his B.S. degree in electrical engineering from University of Science and Technology of Mazandaran, Iran, in 2019. He also received the M.S. degree in system telecommunication engineering from Shahid Rajaee University, Tehran, Iran in 2021: his research interests cryptography, hardware security, and side-channel analysis.