IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2026, No. 3, pp. 465–488. DOI:10.46586/tches.v2026.i3.465-488

MIFA: An MILP-based Framework for Improving Differential Fault Attacks

Hanbeom Shin[1], Insung Kim[1], Sunyeop Kim[1,2], Byoungjin Seok[3], Deukjo Hong[4], Jaechul Sung[5], Seokhie Hong[6], Sangjin Lee[1] and Dongjae Lee[†7]

[1] Korea University, South Korea, {newonetiger,cmcom35,sangjin}@korea.ac.kr
[2] Nanyang Technological University, Singapore, kin3548@gmail.com
[3] Hansung University, South Korea, bjseok@hansung.ac.kr
[4] Jeonbuk National University, South Korea, deukjo.hong@jbnu.ac.kr
[5] University of Seoul, South Korea, jcsung@uos.ac.kr
[6] SmartM2M, South Korea, shhong@smartm2m.co.kr
[7] Kangwon National University, South Korea, dongjae.lee@kangwon.ac.kr

Abstract. At ASIACRYPT 2021, Baksi et al. introduced `DEFAULT`, a block cipher designed to algorithmically resist Differential Fault Attack (DFA), claiming 64-bit DFA security regardless of the number of injected faults. At EUROCRYPT 2022, Nageler et al. demonstrated that `DEFAULT`'s claimed DFA resistance can be broken by applying an information-combining technique. More recently, at ASIACRYPT 2024, Jana et al. improved DFA by searching for differential trails with a single solution. They showed that, for `DEFAULT` with a simple key schedule, injecting five faults at the fifth-to-last round reduces the key space to one, and for `BAKSHEESH`, injecting twelve faults at the third-to-last round achieves the same result. In this paper, we propose a new DFA framework that utilizes a Mixed-Integer Linear Programming (MILP) solver. This framework makes it possible to attack deeper rounds than previously achieved, reducing the number of fault injections required for key recovery. Furthermore, we present a method to determine the most efficient fault injection bit positions by systematically analyzing the input differences from all possible single bit-flip faults, thereby further reducing the required number of faults. This systematic analysis has the significant advantage of allowing us to theoretically calculate the required number of faults. Applying our framework, for `DEFAULT`, injecting three faults at the sixth-to-last round and two faults at the seventh- and eighth-to-last rounds reduces the key space to one.

Keywords: MIFA · Differential Fault Attack · MILP · DEFAULT · BAKSHEESH

1 Introduction

A fault attack is a class of physical attacks in which intentional faults are injected during the execution of a cryptographic or computing device, and the resulting faulty outputs or abnormal behaviors are exploited to extract secret information, recover internal states, or break security mechanisms. Faults can be induced through methods such as voltage or clock glitches [O'F16], electromagnetic pulses [MDH+13], laser injections [SBHS15], or memory disturbances [KDK+14], making fault attacks a powerful and practical threat to both cryptographic implementations and general-purpose systems.

Induced faults are commonly categorized into several models depending on how they alter data and computations. A bit-flip fault inverts one or more bits of a register or

[†]Corresponding Author

Licensed under Creative Commons License CC-BY 4.0.

memory value. A zeroing fault (or stuck-at-zero) forces the affected value to zero, while a stuck-at-one fault fixes bits to one regardless of their intended value. A bit-set fault sets a specific bit to a desired value, either 0 or 1. A randomizing fault replaces the value with an unpredictable state. In addition to these, an instruction skip fault occurs when an operation in the execution flow is bypassed. These models describe typical manifestations of physical fault injection and are widely used in the evaluation of cryptographic implementations.

In this work, we primarily focus on the bit-flip fault model. Achieving such precision is feasible in practice [BBB+10]. According to previous studies, techniques such as laser fault injection offer high accuracy in both space and time, enabling the induction of precise single bit-flip faults in the cryptographic state [ADM+10, DMN+12, SBHS15]. Furthermore, electromagnetic (EM) fault injection serves as an alternative method that does not require de-packaging the chip, and practical implementations of precise bit-level fault injections via EM have been demonstrated [OGSM15]. These advancements suggest that the bit-flip fault model is not merely a theoretical assumption but a practical threat.

Once a specific fault model is justified, the subsequent propagation of the fault within the cipher is determined solely by the cipher's design specifications and internal operations. In other words, the faulty ciphertexts obtained from a physical device are mathematically identical to those produced by a simulation that accurately implements the fault model. Consequently, software simulations serve as a sufficient and robust means to verify the attack's efficacy, including the required number of faults and computational complexity. This validation approach is consistent with methodologies employed in recent cryptanalytic studies [NDE22, JKP24].

Research on fault attacks in cryptography began with the introduction of Differential Fault Attack (DFA) by Biham and Shamir at CRYPTO 1997, which demonstrated that secret keys could be recovered by comparing correct and faulty outputs [BS97]. Biham et al. presented Impossible Differential Fault Analysis (IDFA) at FSE 2005, which eliminates incorrect key candidates that produce an impossible differential [BGN05]. Clavier proposed Ineffective Fault Analysis (IFA) at CHES 2007, showing that even faults that cause no visible output difference can still leak key information [Cla07]. Later, Fuhr et al. introduced Statistical Fault Analysis (SFA) at FDTC 2013, which recovers keys by collecting only faulty ciphertexts and exploiting statistical biases without requiring any correct outputs [FJLT13]. Zhang et al. presented Persistent Fault Analysis (PFA) at CHES 2018, where persistent faults are injected into constants such as S-boxes and later exploited through statistical analysis of collected faulty ciphertexts [ZLZ+18]. In the same year, Dobraunig et al. proposed Statistical Ineffective Fault Attack (SIFA) at CHES 2018, which leverages the probability distribution of ineffective outputs for key recovery [DEK+18]. More recently, at ACNS 2023, Kundu et al. presented Division property based Fault Analysis (DiFA) [KGSR23]. Subsequently, Jana et al. introduced Statistical Differential Fault Analysis (SDFA) at ASIACRYPT 2024, combining differential analysis with statistical techniques [JKP24], and Kundu et al. proposed ToFA (Truncated impossible differential based Fault Analysis) at CHES 2025, which applies truncated impossible differentials to fault analysis [KGAS25]. Various fault attacks on cryptographic systems have been proposed over the years and continue to be an active area of research.

DFA injects faults at specific points during the execution of a cryptographic algorithm and recovers secret keys by analyzing the differences between correct and faulty outputs. The attacker typically injects single or a few faults into intermediate states after several rounds and analyzes how the injected faults propagate through nonlinear components such as S-boxes to infer internal states and key bits. DFA is highly powerful because it can extract a large amount of information with only a small number of faults, but it requires pairs of correct and faulty outputs and precise control over the timing and location of fault injection to succeed. In block cipher settings, performing DFA on rounds that are farther from the plaintext and ciphertext, namely closer to the middle rounds, is considered more

challenging from a cryptanalytic perspective, and attacks that succeed with fewer injected faults are regarded as more valuable.

From a defense perspective, redundancy-based countermeasures are effective but often incur significant implementation overheads in terms of area and power consumption [KGAS25]. Specifically, this approach involves duplicating intermediate states to perform redundant computations for the remaining rounds and suppressing the output if a mismatch between the resulting ciphertexts is detected, thereby preventing the leakage of faulty outputs induced within the protected region. To balance security and efficiency, designers typically limit these protections to the final rounds, which are traditionally perceived as the most vulnerable to DFA [KGAS25]. In this context, successful attacks on deeper rounds carry significant weight as they compel designers to extend the scope of these expensive countermeasures. For example, consider an 80-round cipher protected against a known 5-round DFA by applying countermeasures to the last 5 rounds. If a new 8-round DFA is introduced, it effectively bypasses the existing protection, rendering the cipher vulnerable. To defend against this new threat, the protected region must be expanded, which increases the implementation overhead ratio from roughly 5/80 to 8/80. Consequently, fault attacks that succeed on deeper rounds, specifically those closer to the middle of the cipher, pose a significant practical threat by forcing an increase in defense costs.

`DEFAULT`, proposed by Baksi et al. at ASIACRYPT 2021, aims to provide algorithmic DFA resistance using S-boxes with Linear Structures (LS), claiming that the number of key candidates remains at least 2^{64} regardless of the number of injected faults [Bak22]. However, this claim was refuted by subsequent studies. Nageler et al. first demonstrated at EUROCRYPT 2022 that the security could be significantly undermined using an information-combining technique [NDE22]. Subsequently, Jana et al. presented a more advanced attack at ASIACRYPT 2024 using differential trails (i.e., specific propagation trails of differences through the cipher's internal rounds) with a single solution under the bit-flip fault model [JKP24]. They successfully executed a 5-round DFA, recovering the unique key with only 5 faults for the simple key schedule and $36 + \alpha$ faults for the rotating key schedule. These findings revealed that the originally claimed 2^{64}-level DFA security of `DEFAULT` does not actually hold.

Our Contributions

Building upon the work of Jana et al. [JKP24], we propose a new DFA framework, MIFA (MILP-based Differential Fault Attack), which significantly improves their approach under the bit-flip fault model. MIFA systematically searches for differential trails with a single solution targeting deeper rounds and maximizes attack efficiency. The framework is illustrated in Figure 1, and our main contributions are as follows:

First, we automate and advance the search process for differential trails with a single solution using Mixed-Integer Linear Programming (MILP), significantly expanding the search scope. While Jana et al. [JKP24] employed a meet-in-the-middle strategy by propagating input and output differences separately, this approach suffers from rapidly increasing complexity as the number of rounds increases. In contrast, we model the difference propagation of the entire cipher using MILP constraints and utilize a solver to explicitly verify the uniqueness of the solution for a given input-output difference pair by checking the number of feasible solutions. This modeling-based approach efficiently handles complex propagation behaviors, enabling the robust identification of unique differential trails even in deeper rounds, which were difficult to reach with previous matching techniques. Provided that the MILP model is correctly implemented, the fact that a single solution satisfies the constraints serves as a structural invariant that guarantees the trail is unique, regardless of the model's optimality or the solver's execution speed.

Second, we present a method to determine the optimal fault injection strategy by

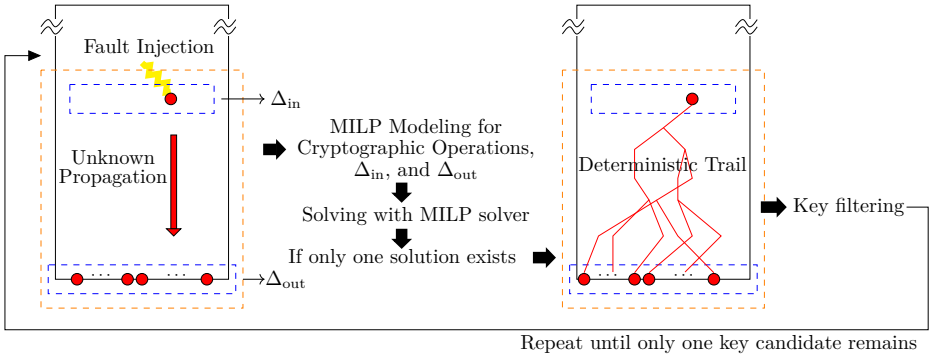

Figure 1: An illustration of our MILP-based DFA framework.

systematically analyzing all 128 possible single bit-flip faults. The efficiency of key recovery using a differential trail with a single solution is closely tied to the Linear Structure (LS) properties of the S-box. Specifically, an LS difference provides no key filtering capability because its corresponding entry in the Difference Distribution Table (DDT) is maximal (e.g., 16 for a 4-bit S-box), meaning it imposes no restrictions on the key candidates. Consequently, a trail containing a higher number of non-LS differences is essential for effective key filtering. However, in attacks on deeper rounds, an increase in non-LS differences paradoxically lowers the probability that a differential trail with a single solution exists, creating a trade-off. We address this by evaluating both the expected filtering capability and the probability of a unique trail's existence for every bit-flip position. This systematic analysis allows us to select the optimal fault that minimizes the total attack complexity.

Third, we provide a precise theoretical derivation of the required number of fault injections, supported by an analysis of the key space reduction behavior. We calculate the size of the remaining key candidate space to determine the exact number of faults required to recover the unique key. Crucially, this calculation explicitly accounts for the probability of a unique trail being found, thereby accurately reflecting the total expected number of injections, including any re-injections required when a fault fails to yield a differential trail with a single solution.

We applied this framework to `DEFAULT`, achieving results that surpass those of previous studies. For `DEFAULT` with the simple key schedule, we achieved a 6-round DFA with 3 faults, and 7- and 8-round DFAs with 2 faults each. For the rotating key schedule, the attacks require only $28 + \alpha$, $13 + \alpha$, and $11 + \alpha$ faults for 6-, 7-, and 8-round DFAs, respectively. Furthermore, we applied our framework to `BAKSHEESH` [BBC+23], successfully extending the attack range from the 3 rounds achieved by Jana et al. [JKP24] to 5 rounds. However, since these results are less efficient compared to the recently proposed ToFA (Truncated impossible differential-based Fault Analysis), we omit the detailed description of the key recovery for `BAKSHEESH` in this paper. A detailed comparison is provided in Table 1.

These attacks were validated through simulations. Our experiments confirmed that the probability of obtaining a differential trail with a single solution becomes negligible for attacks targeting 9 rounds or more on `DEFAULT`. This strongly suggests that our proposed attack has reached the maximal feasible depth for this approach, effectively establishing the practical upper bound for DFAs relying on unique trails against this cipher.

Table 1: Comparison of DFA attacks on DEFAULT.

Cipher	Attack	Rnd	#F	KS	Ref.
DEFAULT (Simple)	Information-Combining	1	16	2^{39}	[NDE22]
	SDFA	1	64∼128	1	[JKP24]
	Unique Diff. Trail	2	64	2^{32}	[JKP24]
	Information-Combining	3	16	2^{20}	[NDE22]
	Unique Diff. Trail	3	34	1	[JKP24]
	Unique Diff. Trail	4	16	1	[JKP24]
	Unique Diff. Trail	5	5	1	[JKP24]
	MIFA	6	3	1	Sec. 4.3
	MIFA	7	2	1	Sec. 4.3
	MIFA	8	2	1	Sec. 4.3
DEFAULT (Rotating)	Information-Combining	1	$1728 + \alpha$	1	[NDE22]
	Information-Combining	1	$288 + \alpha$	2^{32}	[NDE22]
	SDFA	1	64∼128	1	[JKP24]
	Information-Combining	3	$(84 \pm 15) + \alpha$	1	[NDE22]
	Unique Diff. Trail	3	$96 + \alpha$	1	[JKP24]
	Unique Diff. Trail	4	$48 + \alpha$	1	[JKP24]
	Unique Diff. Trail	5	$36 + \alpha$	1	[JKP24]
	MIFA	6	$28 + \alpha$	1	Sec. 4.4
	MIFA	7	$13 + \alpha$	1	Sec. 4.4
	MIFA	8	$11 + \alpha$	1	Sec. 4.4

Rnd: Target rounds from the end of the cipher; #F: Number of injected faults; KS: Remaining key space. α signifies the number of faults injected into the DEFAULT-CORE.

Organization

The remainder of this paper is organized as follows. Section 2 introduces the necessary preliminaries, including an overview of DFA, the specifications of DEFAULT, the background on DFA based on differential trails with a single solution, and MILP-based methods for searching for differential trails. In Section 3, we present our MILP-based framework for improving DFA. We apply the proposed framework to DEFAULT in Section 4 to perform a key recovery attack and derive the precise fault complexity. Finally, we conclude the paper in Section 5. The implementation code for our framework is publicly available online.[1]

2 Preliminaries

2.1 Differential Fault Attack

Differential Fault Attack (DFA) is a cryptanalytic technique that recovers secret keys by injecting intentional faults into a cryptographic implementation and analyzing the differences between correct and faulty outputs. Specifically, DFA treats the fault injected into an intermediate state as a specific input difference Δ_{in}. For instance, if an attacker induces a fault at the input of an S-box in the final round, they can obtain the correct ciphertext c and the faulty ciphertext c'. The attacker can then verify a key candidate k by checking whether the following equation holds:

$$S^{-1}(c \oplus k) \oplus S^{-1}(c' \oplus k) = \Delta_{\text{in}}.$$

[1] https://github.com/shb115/MIFA

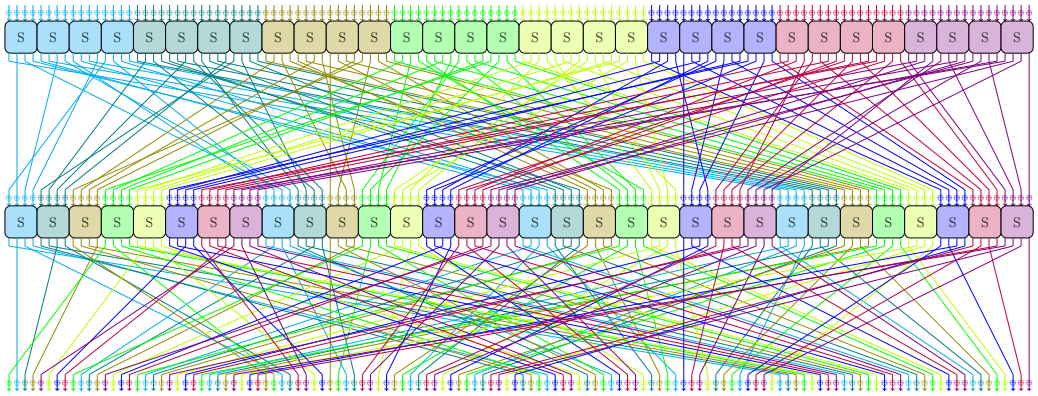

Figure 2: QR group structure of DEFAULT

Key candidates that fail to satisfy this equality are eliminated. Specifically, when guessing a sub-key corresponding to the S-box size, the number of remaining candidates satisfying the equation is exactly equal to the value of the entry in the DDT corresponding to the input and output differences. Consequently, utilizing difference pairs associated with smaller values in the DDT yields stronger filtering power, thereby allowing for a more efficient reduction of the key search space. Since DFA typically targets the final operations of a cipher to extract the key, we establish the following convention: throughout this paper, any reference to targeting n rounds (e.g., a 6-round or 7-round DFA) inherently denotes an attack on the last n rounds of the cipher, counted from the end, unless explicitly stated otherwise.

2.2 Descriptions of DEFAULT and BAKSHEESH

In this section, we provide the detailed specifications and structural properties of the two block ciphers analyzed in our study.

2.2.1 Description of DEFAULT

DEFAULT is a 128-bit block cipher proposed by Baksi et al. at ASIACRYPT 2021, designed to provide algorithm-level resistance against DFA. Its structure is given by

$$E = E_{\texttt{DEFAULT-LAYER}} \circ E_{\texttt{DEFAULT-CORE}} \circ E_{\texttt{DEFAULT-LAYER}},$$

where the two $E_{\texttt{DEFAULT-LAYER}}$ parts consist of 28 rounds each and the $E_{\texttt{DEFAULT-CORE}}$ part consists of 24 rounds, resulting in 80 rounds in total. DEFAULT-LAYER mainly provides DFA resistance, while DEFAULT-CORE is designed to resist classical cryptanalysis.

SubCells. Each 4-bit nibble of the state is substituted by an S-box. DEFAULT-LAYER uses S-box S with four LS elements, while DEFAULT-CORE uses S-box S_{core} with no non-trivial LS element, as detailed in Table 2.

PermBits. A bit permutation adopted from GIFT-128 [BPP+17] is applied. It forms a Quotient-Remainder (QR) group structure, where the 16 bits coming from 4 S-boxes in one round affect only 4 S-boxes in the next round. The figure illustrating the QR group structure can be found in Figure 2.

AddRoundConstants. A 6-bit round constant is XORed to specific bit positions (23, 19, 15, 11, 7, 3), and the most significant bit (bit 127) is toggled in every round to break symmetry between rounds.

`AddRoundKey`. A 128-bit round key is XORed to the state. `DEFAULT` specifies two types of key schedules.

Simple Key Schedule. In this variant, the same 128-bit master key K is applied as the round key for every round. That is, for any round r, $K_r = K$.

Rotating Key Schedule. This variant derives four distinct 128-bit round keys, denoted by K_0, K_1, K_2, and K_3, from the master key. Specifically, K_0 is set to the master key, and each subsequent key K_{i+1} is generated by applying 4 rounds of `DEFAULT-LAYER` operations to K_i. In each round r, the round key is determined cyclically as $K_{r \bmod 4}$.

Table 2: S-boxes and their DDTs used in `DEFAULT`

(a) `DEFAULT-LAYER` S-box S

u	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(u)$	0	3	7	e	d	4	a	9	c	f	1	8	b	2	6	5

(b) `DEFAULT-CORE` S-box S_{core}

u	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S_{\text{core}}(u)$	1	9	6	f	7	c	8	2	a	e	d	0	4	3	b	5

(c) DDT of S

$I\backslash O$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	·	·	8	·	·	·	·	8	·	·	·	·	·	·	·
2	·	·	·	·	·	·	·	8	·	·	·	·	·	8	·	·
3	·	·	·	·	8	·	·	·	·	·	·	·	·	8	·	·
4	·	·	·	·	·	·	·	8	·	·	·	·	·	8	·	·
5	·	·	·	·	8	·	·	·	·	·	·	·	·	8	·	·
6	·	·	·	·	·	·	·	·	·	·	16	·	·	·	·	·
7	·	·	·	8	·	·	·	·	8	·	·	·	·	·	·	·
8	·	·	·	·	·	·	8	·	·	·	·	·	8	·	·	·
9	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	16
a	·	8	·	·	·	·	·	·	·	·	·	8	·	·	·	·
b	·	·	8	·	·	·	·	·	8	·	·	·	·	·	·	·
c	·	8	·	·	·	·	·	·	·	·	·	8	·	·	·	·
d	·	·	8	·	·	·	·	·	8	·	·	·	·	·	·	·
e	·	·	·	·	·	·	8	·	·	·	·	·	8	·	·	·
f	·	·	·	·	·	16	·	·	·	·	·	·	·	·	·	·

(d) DDT of S_{core}

$I\backslash O$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	·	·	·	2	·	·	2	2	2	2	2	·	2	2	·
2	·	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4
3	·	2	·	2	2	2	·	·	2	·	2	·	·	·	2	2
4	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4	·
5	·	·	·	·	2	·	·	2	2	2	2	2	·	2	2	·
6	·	4	·	4	·	·	·	·	4	·	4	·	·	·	·	·
7	·	2	·	2	2	2	·	·	2	·	2	·	·	·	2	2
8	·	·	·	4	·	·	·	4	·	·	·	4	·	·	·	4
9	·	·	2	2	2	·	2	·	2	2	·	·	·	2	·	2
a	·	4	·	·	·	·	·	·	·	4	·	·	8	·	·	·
b	·	2	2	·	2	2	2	2	2	·	·	2	·	·	·	·
c	·	·	4	·	·	4	·	·	·	·	4	·	·	4	·	·
d	·	·	2	2	2	·	2	·	2	2	·	·	·	2	·	2
e	·	·	4	·	·	·	·	·	·	·	·	4	·	8	·	·
f	·	2	2	·	2	2	2	2	2	·	·	2	·	·	·	·

Linear Structure and DFA Resistance. To understand the DFA resistance of `DEFAULT`, it is essential to define the Linear Structure.

Definition 1 (Linear Structure)**.** For an S-box S, if a pair (Δ, ∇) satisfies

$$S(x) \oplus S(x \oplus \Delta) = \nabla$$

for all inputs x, then (Δ, ∇) is called a linear structure (LS) of S.

For every S-box, (0,0) is always an LS. We therefore refer to any other LS as a non-trivial LS, and an S-box that has at least one non-trivial LS is called an LS S-box. The S-box used in `DEFAULT-LAYER` is a 4-bit S-box with four LS elements; when these appear

as input and output differences, no key candidates can be eliminated. Moreover, [Bak22] claimed that due to the properties of the LS, it is impossible to reduce the key space to one, regardless of the number of faults injected. For a cipher based on an LS S-box, assume a fault is injected into the input of an S-box one round before the last. The attacker knows the correct ciphertext c, the faulty ciphertext c', and the injected input difference Δ_{in}, but does not know the actual S-box input u or the faulty input $u' = u \oplus \Delta_{\text{in}}$. If a pair (u, k) exists such that $S(u) \oplus k = c$ and $S(u \oplus \Delta_{\text{in}}) \oplus k = c'$, then due to the LS property, the pair $(u \oplus \Delta, k \oplus \nabla)$ also produces the same ciphertext pair (c, c'). In other words, if a key candidate k is possible, then $k \oplus \nabla$ is also possible for any LS output difference ∇, meaning multiple key candidates will always remain. Therefore, the size of the key candidate space cannot be reduced below $(\#\text{LS})^{\#\text{S-boxes}}$. For example, in `DEFAULT`, this leaves $4^{32} = 2^{64}$ key candidates.

2.2.2 Description of `BAKSHEESH`

`BAKSHEESH` [BBC+23] is a lightweight block cipher with a 128-bit block size and a 128-bit key, designed based on the GIFT-128 structure. It consists of 35 rounds, and each round is composed of the following four operations.

`SubCells`. A 4-bit S-box is applied to each 4-bit nibble of the state. The S-box contains a single non-trivial LS element, as detailed in Table 3.

Table 3: S-box and DDT of `BAKSHEESH`

(a) The `BAKSHEESH` S-box S

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S(x)	3	0	6	d	b	5	8	e	c	f	9	2	4	a	7	1

(b) DDT of S

$\Delta I \backslash \Delta O$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
1	·	·	·	4	·	·	4	·	·	·	·	4	·	·	4	·
2	·	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·
3	·	·	·	·	·	4	4	·	·	·	·	·	·	4	4	·
4	·	·	·	4	·	4	·	·	4	·	·	·	·	·	4	·
5	·	·	·	·	·	4	4	·	4	·	·	4	·	·	·	·
6	·	·	·	·	·	·	·	·	4	·	·	4	·	4	4	·
7	·	·	·	4	·	·	4	·	4	·	·	·	·	4	·	·
8	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	16
9	·	4	·	·	4	·	·	·	·	4	·	·	4	·	·	·
a	·	·	4	·	4	·	·	·	·	·	4	·	4	·	·	·
b	·	4	4	·	·	·	·	·	·	4	4	·	·	·	·	·
c	·	4	·	·	·	·	·	4	·	·	4	·	4	·	·	·
d	·	·	·	·	4	·	·	4	·	4	4	·	·	·	·	·
e	·	4	4	·	4	·	·	4	·	·	·	·	·	·	·	·
f	·	·	4	·	·	·	·	4	·	4	·	·	4	·	·	·

`PermBits`. The bits of the state are permuted using a bit permutation similar to that of GIFT-128 (QR group structure) but with different bit positions.

`AddRoundConstants`. At each round, a 6-bit round constant and one additional toggle bit are XORed into specific bit positions of the state.

`AddRoundKey`. The round key is XORed with the state. The first round key is the master key itself, and each subsequent round key is generated by rotating the previous key by one bit to the right.

2.3 DFA Based on Differential Trails with a Single Solution

Jana et al. [JKP24] proposed an efficient DFA technique that exploits differential trails with a single solution. The attack begins by injecting a single bit-flip fault several rounds prior to the last to obtain a pair of correct and faulty ciphertexts. Using the known fault difference and the observed ciphertext difference, they employ a meet-in-the-middle strategy by propagating the differences forward and backward to deduce the exact intermediate differences. This process identifies a specific differential trail that guarantees a single, unique solution for the internal states.

This unique trail is then utilized for key recovery; however, since the LS of the S-box inherently leaves 2^{64} key candidates at the last round regardless of the number of injected faults, it prevents the immediate recovery of a unique key. To overcome this structural limitation, the attacker combines key guesses and performs partial decryption to the upper rounds. By verifying whether the difference of the decrypted intermediate state exactly matches the deterministic intermediate differences specified by the pre-computed unique trail, incorrect key candidates are effectively filtered out. This filtering process is repeated, iteratively reducing the key space until the unique key is successfully recovered.

2.4 MILP-based Search for Differential Trails

Mixed-Integer Linear Programming (MILP) is a mathematical optimization paradigm used to optimize an objective function subject to a set of linear constraints, where some variables are restricted to integer values. Since its introduction to cryptography by Mouha et al. [MWGP11] and Sun et al. [SHW+14], it has been widely adopted for various cryptanalytic tasks, including finding differential trails [SHW+14] and identifying impossible differentials [ST17, CCF+21]. Furthermore, alongside MILP, SAT, and SMT solvers, recent works such as Hadipour et al. [HSE23, HGSE24] have demonstrated that Constraint Programming (CP) is highly effective in automating the search for impossible differential, zero correlation, and integral attacks.

Following the standard modeling methodology established in [SHW+14], where a differential trail represents the sequence of intermediate differences propagating through the cipher, the difference of each bit in the cipher state is represented by a binary variable. For non-linear operations like S-boxes, feasible input-output difference patterns are extracted from the DDT and converted into a set of linear inequalities. This conversion is typically achieved using techniques such as H-representation or logic minimization. Specifically, H-representation is a geometric approach that models the valid differential patterns as a convex hull defined by the intersection of half-spaces (linear inequalities). Alternatively, logic minimization tools like Espresso treat the valid patterns as the ON-set of a Boolean function, simplifying it to generate a minimal and compact set of linear constraints. Linear layers, including bit permutations and XOR operations, are modeled using equality constraints and XOR constraints, respectively. Operations involving constants or round keys are omitted as they do not affect differential propagation.

Based on this model, to find the optimal differential trail, the objective function is set to minimize the weight of the differential probability. This enables the solver to efficiently search for the trail with the maximum probability. Conversely, when analyzing impossible differentials, no objective function is defined. Instead, the target input difference Δ_{in} and output difference Δ_{out} are added as constraints to check the feasibility of a solution. If the solver returns an infeasible status, it indicates that the differential trail is structurally impossible.

3 MIFA: MILP-based Differential Fault Attack Framework

In this section, we propose MIFA, a framework designed to enhance DFA through the use of an MILP solver. Since our framework and the subsequent analyses focus on the differential propagation within the core round functions, the methodologies presented here are independent of the specific key schedule and apply equally to both the Simple and Rotating Key Schedules of `DEFAULT`. First, we introduce an MILP-based method capable of identifying differential trails with a single solution that extend deeper into the cipher than those found by previous approaches. Attacks targeting deeper rounds are widely regarded as more potent, as they compel designers to implement countermeasures across more rounds. Moreover, longer unique differential trails allow for more frequent key filtering steps per fault, thereby enhancing filtering efficiency and reducing the total number of required fault injections. Furthermore, we present a method for determining the most effective bit positions for fault injection by systematically analyzing all possible bit-flip faults. This optimization further decreases the number of faults necessary for key recovery.

3.1 MILP-based Search for Differential Trails with a Single Solution

In the preceding work by Jana et al. [JKP24], differential trails with a single solution were identified by propagating differences bidirectionally and verifying their collision at an intermediate round (meet-in-the-middle). However, as the number of rounds increases, this approach becomes computationally infeasible due to the exponential growth of the search space. To mitigate this limitation, we propose a new approach utilizing an MILP solver that efficiently searches for differential trails with a single solution even in deeper rounds.

The key contribution of our proposed method lies in a paradigm shift regarding the usage of the MILP solver. While traditional searches for impossible differentials focus on the absence of solutions (zero solutions), and standard differential characteristic searches focus on finding a trail with the optimal probability, our framework focuses on the uniqueness of the solution. Since the input and output differences obtained from actual physical fault injections are real phenomena, the MILP system modeling them must possess at least one feasible solution. If the solver returns exactly one solution for these constraints, it implies that the internal propagation trail is mathematically unique for the given input–output difference pair. This unique trail constitutes the differential trail with a single solution we seek.

The search process is summarized in the following steps:

Step 1. Inject a single bit-flip fault into the target round during encryption to obtain the correct ciphertext C and the faulty ciphertext C'.

Step 2. Identify the input difference Δ_{in} corresponding to the fault injection position, compute the output difference $\Delta_{\text{out}} = C \oplus C'$, and set these values as constraints for the input and output variables in the MILP model.

Step 3. Model the differential propagation of the cipher following the standard framework proposed by Sun et al. [SHW+14].

Step 4. Execute the solver to count the number of feasible solutions, denoted as N_{sol}.

Step 5. Proceed based on the value of N_{sol}:

- If $N_{\text{sol}} = 1$, the internal propagation trail is uniquely identified. Return the differential trail with a single solution and proceed to key recovery.
- If $N_{\text{sol}} > 1$, the unique trail cannot be determined. Return to **Step 1** to inject a new fault.

We applied this framework to DEFAULT and BAKSHEESH. Our experiments confirmed the feasibility of 6-, 7-, and 8-round DFAs on DEFAULT and 4- and 5-round DFAs on BAKSHEESH. This was verified through 1,000 independent experiments for each cipher using random data. We injected a single bit-flip fault at position 0x1 for DEFAULT and 0x4 for BAKSHEESH. The choice of the fault injection position significantly influences the attack complexity. We specifically selected these positions for this experiment because they exhibit the highest probability of yielding a differential trail with a single solution. A comprehensive analysis of all fault positions, which additionally takes key filtering efficiency into account, will be detailed in Section 3.2. The results are summarized in Table 4.

Table 4: Experimental results of finding unique differential trails for DEFAULT (with a 0x1 fault) and BAKSHEESH (with a 0x4 fault). Here, "# Unique" and "# Multiple" denote the number of cases that resulted in a unique solution and multiple solutions, respectively.

Cipher	Fault on Round	# Unique	# Multiple
DEFAULT	sixth-to-last	1000	0
	seventh-to-last	1000	0
	eighth-to-last	999	1
	ninth-to-last	0	12
BAKSHEESH	fourth-to-last	972	28
	fifth-to-last	457	543
	sixth-to-last	0	1000

The extremely low probability of finding unique solutions in deeper rounds (i.e., the ninth-to-last round for DEFAULT and the sixth-to-last round for BAKSHEESH) is an inherent structural consequence. As the number of rounds increases, the cipher's diffusion causes the number of possible difference propagation branches to multiply exponentially. Consequently, it becomes highly improbable, though not strictly impossible, for a given pair of input and output differences to be satisfied by only a single unique trail.

All experiments were conducted on a workstation running Ubuntu 22.04 LTS with Gurobi 11.0.0, equipped with dual Intel Xeon Gold 6230R CPUs (providing a total of 104 threads) and 2 TB of RAM. Leveraging this multi-core environment, we utilized 100 parallel processes to execute the experiments. The runtime of the MILP solver generally increased with the number of analyzed rounds, although it varied depending on the specific input fault and output difference. For DEFAULT, trails for 6- and 7-round DFAs were obtained in under a second, and 8-round DFAs took about 10 seconds on average. However, as the round depth increases, the number of variables and the overall search space within the MILP model expand drastically due to the inherent exponential growth in search complexity. Consequently, the execution time for 9 rounds surged to approximately 24 hours per trail. This severe computational bottleneck restricted our feasible sample size, which explains why the 9-round evaluation was limited to 12 experiments rather than the 1,000 independent trials conducted for the 6-, 7-, and 8-round cases. For BAKSHEESH, trails for 4- and 5-round DFAs were found in under a second, while the 6-round search averaged about 10 seconds.

3.2 Searching for Effective Fault Injections for Key Recovery

The efficiency of a DFA is heavily influenced by the bit position of the injected fault. Specifically, the S-boxes used in DEFAULT-LAYER and BAKSHEESH possess a Linear Structure (LS), which creates a critical issue: for certain input differences (specifically 0x0, 0x6, 0x9, or 0xf for DEFAULT-LAYER, and 0x0 or 0x8 for BAKSHEESH), the S-box produces a constant output difference regardless of the key value. This makes key filtering impossible. Consequently, selecting a fault that generates a high number of non-LS differences is

essential for effective key filtering. Accurate estimation of the expected number of non-LS differences at each round is therefore a prerequisite for calculating the precise number of faults required for a successful attack.

We conducted a comprehensive search to identify the most effective single bit-flip faults. While there are 128 possible single bit-flip faults for a 128-bit state, theoretically analyzing all differential trails over the full number of rounds for every fault is computationally infeasible due to the exponential growth of the search space. To address this, we adopted a hybrid approach. We theoretically calculated the expected values for the initial rounds where exhaustive analysis is computationally manageable, and then switched to experimental simulations to estimate the probabilities for the subsequent deeper rounds.

In the theoretical analysis phase, we simulated the propagation of differences over the target rounds for all possible single bit-flip faults. Instead of exhaustively enumerating all individual differential trails, we tracked the probability distribution of differences based on the transition probabilities derived from the DDT of the S-box. Specifically, for a given input difference, we calculated the possible output differences and their associated probabilities, propagating this distribution through subsequent rounds. Based on this probabilistic propagation model, we calculated the total expected number of non-LS differences occurring across the analyzed rounds.

The analysis revealed that the expected value of non-LS differences varies significantly depending on the fault position, classifying them into distinct groups as shown in Table 5. Note that in Table 5, the fault classes for `0x2` and `0x4` in `DEFAULT` exhibit identical properties. This is because they form an equivalence class due to the structural symmetry and the `PermBits` layer of `DEFAULT`. For brevity, we discuss the `0x2` fault as a representative example, but the analysis applies equally to the `0x4` fault.

Table 5: Expected number of non-LS differences over three rounds for different faults in `DEFAULT` and `BAKSHEESH`

Cipher	Fault Pattern	Expected Non-LS differences
`DEFAULT`	`0x2, 0x20, ...`, and `0x4, 0x40,...`	26.75 (Highest)
	`0x8, 0x800, 0x80000, ...`	19.875
	`0x80, 0x8000, 0x800000, ...`	19.625
	`0x10, 0x1000, 0x100000, ...`	17.125
	`0x1, 0x100, 0x10000, ...`	16.875 (Lowest)
`BAKSHEESH`	`0x8, 0x800, 0x80000, ...`	31.375 (Highest)
	`0x80, 0x8000, 0x800000, ...`	31.125
	`0x1, 0x10, ...`, and `0x2, 0x200, ...`	20.3125
	`0x20, 0x2000, 0x200000, ...`	20.25
	`0x40, 0x4000, 0x400000, ...`	17.09375
	`0x4, 0x400, 0x40000, ...`	16.53125 (Lowest)

Based on these expected values, we selected the optimal faults by balancing the high probability of generating non-LS differences (filtering efficiency) and the high probability of finding a differential trail with a single solution. A trade-off exists between these two factors: faults that generate many non-LS differences propagate rapidly, which benefits filtering but reduces the likelihood of a unique deterministic trail in deeper rounds. Instead of defining complex metrics, we adopted a single practical criterion: minimizing the total number of fault injections required for key recovery. We selected the fault that yields the lowest total fault count when considering both the success rate of finding a trail and the filtering power of that trail. Specifically, the total fault count defined in our metric is comprehensive; it accounts not only for the faults effectively used for key filtering but also for the expected number of wasted faults that fail to generate a differential trail with a

single solution.

For DEFAULT, the 0x2 fault produces the most non-LS differences and is most efficient for filtering. However, while it guarantees a deterministic trail for 6- and 7-round attacks, it fails to do so consistently for 8-round attacks. In contrast, the 0x1 fault, despite being the least effective for filtering, maintains a 99.9% probability of finding a deterministic trail even for an 8-round attack (as shown in Table 4). Consequently, for deep-round attacks (e.g., 8 rounds), the 0x1 fault is the optimal choice despite its lower filtering efficiency per fault.

A similar trade-off is observed in BAKSHEESH. The 0x8 fault provides the highest filtering efficiency and works well for 4-round attacks (88.8% trail probability) but becomes impractical for 5-round attacks (0.9%). Conversely, the 0x4 fault, which propagates more slowly, retains a viable 45.7% probability for 5-round attacks. Thus, we selected the 0x4 fault as the optimal position for analyzing BAKSHEESH.

For rounds beyond the reach of theoretical analysis, we estimated the average number of non-LS differences experimentally using 100,000 random trials. The empirical results obtained from these simulations are summarized in Table 6. This estimation is crucial for determining the precise number of faults required. As the number of rounds increases, the input difference propagates through the cipher, causing the difference distribution to approach a uniformly random distribution. Accordingly, we utilize theoretical calculations for the initial rounds, switch to experimental estimation until the distribution converges to uniformity, and assume a uniform random distribution for all subsequent rounds. We apply this methodology in Section 4 to calculate the probability of non-LS differences for our attack on DEFAULT.

Table 6: Average difference occurrences per nibble in DEFAULT with 0x2 fault

Round	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$r-n+4$	7.27	3.85	2.55	1.60	4.48	2.32	1.12	0.68	2.43	1.65	0.93	0.73	1.15	0.73	0.31	0.20
$r-n+5$	2.76	2.64	2.06	1.90	2.65	2.57	1.95	1.84	2.09	1.95	1.63	1.45	1.89	1.82	1.47	1.35
$r-n+6$	2.03	2.03	1.99	2.00	2.04	2.03	2.00	2.00	1.99	2.00	1.97	1.97	2.00	2.00	1.97	1.98

r denotes the round of the cipher, and $r-n$ denotes the fault injection round.

4 Key Recovery Attack on DEFAULT and Analysis of Required Faults

In this section, we apply the proposed MIFA framework to DEFAULT to perform a practical key recovery attack. Before detailing the attack procedure, we theoretically address the potential concern regarding irregular reductions in the number of candidate keys during filtering. We show that the number of candidate keys in nibble-level filtering always maintains a power-of-two form (2^k). Furthermore, we demonstrate that the lower bound of the candidate key space for group-level filtering is 2^4. We then verify this property through experimental results and present the specific attack procedure along with the number of faults required for key recovery.

4.1 Structural Properties of Key Space Reduction

The core of DFA involves identifying candidate keys that satisfy the differential equation $S(u) \oplus S(u \oplus \Delta_{\text{in}}) = \Delta_{\text{out}}$. Through an in-depth analysis specifically targeting DEFAULT, we have identified novel structural properties regarding the key recovery process: the discrete reduction pattern (2^k) at the nibble level and the structural lower bounds at the group level determined by the properties of equivalent keys.

Nibble Level: Equivalent Keys and 2^k Reduction. Due to the LS properties of the S-box, valid key candidates form sets of indistinguishable keys, referred to as equivalent keys [NDE22]. Since the S-box of `DEFAULT` possesses 4 LS, each nibble contains 4 equivalent keys. We define this set of 4 keys as a single key group. Consequently, the initial 16 candidate keys consist of exactly 4 key groups. The DDT of the `DEFAULT` S-box comprises solely the values 0, 8, and 16. In the context of key groups, a DDT value of 8 indicates that exactly 2 key groups are possible. Since the elements of a key group are always preserved or eliminated together, the size of the candidate pool reduces from 4 key groups (16 keys) to 2 key groups (8 keys), and finally to 1 key group (4 keys), thereby always maintaining a power-of-two form (2^k).

Group Level: Lower Bound of 2^4 Due to Equivalent Keys. At the group level, the lower bound of the key space is determined by the expansion of equivalent keys. In the structure of `DEFAULT`, a single group key affects 4 nibbles in the upper round, which are used for filtering. Each of these 4 nibbles independently possesses 4 equivalent keys. When combined at the group level, these result in a total of 16 indistinguishable keys. Consequently, regardless of the number of faults injected, the candidate keys within a group remain indistinguishable from one another. This makes it structurally impossible for the number of candidate keys at the group level to fall below 2^4.

Half Level: Convergence to a Unique Key via Inter-Round Dependency. At the half level, a total of 16 nibbles are involved in the filtering process. Unlike the group level which spans two rounds, the half level extends the key guessing process over three rounds, applying the final filtering at this third round to the candidates that have already passed the two-round constraints. The equivalent key property, maintained at the nibble and two-round group levels, is no longer preserved because the diffusion layer thoroughly intertwines these 16 nibbles across this extended three-round span, inherently breaking local equivalences. Furthermore, as detailed in Appendix A.2, our experimental results confirm this behavior, demonstrating that while the candidate space is bounded at 2^4 at the group level, the number of candidate keys successfully converges to a unique key (2^0) upon reaching the half level.

These theoretical findings regarding the number of remaining candidate keys at each level were verified experimentally, as detailed in Appendix A.

4.2 Effective Fault Selection for `DEFAULT`

Based on the analysis in Section 3.2, we select the input faults to minimize the total number of faults required for key recovery. Accordingly, we utilize `0x2` for the 6- and 7-round attacks, and `0x1` for the 8-round attack. The distribution of non-LS differences for these faults directly influences the calculation of the required number of fault injections presented in Section 4.3.

Specifically, for the `0x2` fault, theoretical derivation is possible up to round $r-n+3$. The average number of non-LS differences propagates theoretically as 1, 3, 7.5, and 16.25 for rounds $r-n$ to $r-n+3$, respectively. For round $r-n+4$, experimental verification yields 21.76, and from round $r-n+5$ onward, it converges to approximately 24, which represents a uniform distribution.

Similarly, for the `0x1` fault, the sparsity of active trails permits theoretical derivation up to round $r-n+4$. The theoretical propagation sequence from round $r-n$ to $r-n+4$ is 1, 2, 4.5, 10.375, and 19.273, respectively. Subsequent experimental verification shows 23.4 for round $r-n+5$, and from round $r-n+6$ onward, it also converges to approximately 24.

4.3 Improved DFA on DEFAULT with Simple Key Schedule

In DEFAULT, the simple key schedule applies the same master key to all rounds. Jana et al. [JKP24] proposed a key filtering method that guesses the partial key to partially decrypt the correct and faulty ciphertexts one round at a time, and then compares the resulting difference against that of a differential trail with a single solution. The detailed attack procedure for DEFAULT with a simple key schedule consists of the following four main steps:

Step 1: Filtering at the first-to-last round. The last round key is guessed nibble by nibble to perform a one-round partial decryption. The resulting differences are compared with the expected differences of the deterministic trail at the first-to-last round. If they do not match, the guessed key is discarded.

Step 2: Filtering at the second-to-last round. The filtered last round key is guessed group-wise, where a group consists of 4 nibbles at positions $\{0, 8, 16, 24\}$, $\{1, 9, 17, 25\}$, and so on. Then, the key at the first-to-last round is guessed nibble-wise. Using both guessed keys, partial decryption is performed to compare the resulting differences with the expected differences at the second-to-last round. At this stage, due to the simple key schedule, 4 nibbles at the first-to-last round are determined by the group key guessed in the last round (group-determined nibbles), while the others are not (non-group-determined nibbles). In the case of a non-group-determined nibble, the currently guessed group key is discarded only if all key candidates for that specific nibble, which survived Step 1, fail to satisfy the difference condition.

Step 3: Filtering at the third-to-last round. The filtered last round key is guessed half-wise, where a half consists of 4 groups (totaling 16 nibbles) corresponding to groups $\{0, 2, 4, 6\}$ and $\{1, 3, 5, 7\}$. Then, the key at the first-to-last round is guessed group-wise, and the key at the second-to-last round is guessed nibble-wise. Using these guessed keys, partial decryption is performed to compare the resulting differences with the expected differences at the third-to-last round.

At this stage, due to the simple key schedule, the 32 nibbles are classified into four distinct categories, each containing 8 nibbles: (1) *Fully-Determined*, where both the group key at the first-to-last round and the nibble key at the second-to-last round are determined by the guessed half key; (2) *Group-Determined*, where only the group key is determined; (3) *Nibble-Determined*, where only the nibble key is determined; and (4) *Non-Determined*, where neither key is determined.

For the cases where the group key, the nibble key, or both are not determined, the currently guessed half key is discarded if all corresponding candidate keys surviving from Steps 1 and 2 fail to satisfy the difference condition.

Step 4: Filtering before the third-to-last rounds. The filtered last round key is guessed as a full 128-bit key, and partial decryption is performed up to the target round. The key is discarded if the resulting differences do not match those of the corresponding round in the deterministic trail.

We calculate the number of faults required to reduce the key candidates to a unique value in a DFA on DEFAULT with a simple key schedule. We achieve this by tracking the size of the remaining key candidate space round by round, utilizing the identified differential trails with a single solution. Our calculation is based on the probabilities of non-LS differences occurring, as analyzed in Section 4.2.

4.3.1 6-Round DFA on DEFAULT with Simple Key Schedule

We analyze the DFA on DEFAULT with a simple key schedule, where faults are injected at the sixth-to-last round. Let n_t be the number of differential trails with a single solution.

Filtering at the first-to-last round. The reduction of the key candidate space begins at an initial size of 2^4. Since the DDT entry value for a non-LS difference is 8, the occurrence of a single non-LS difference unconditionally reduces the candidate space to 2^3. The probability that a specific nibble exhibits a non-LS difference is $24/32 = 3/4$. Specifically, the 2^4 key space consists of 4 key groups, and a non-LS difference restricts candidates to 2 groups. Thus, when multiple non-LS differences occur, the probability that the surviving key groups completely overlap, resulting in a candidate size of 2^3, is $(1/2)^n$. Therefore, the expected size of the key candidate space, given $n_t = 3$, is calculated as:

$$\left(\frac{1}{4}\right)^3 \cdot 2^4 + \left[3\left(\frac{1}{4}\right)^2\left(\frac{3}{4}\right) + 3\left(\frac{1}{4}\right)\left(\frac{3}{4}\right)^2\left(\frac{1}{2}\right)^2 + \left(\frac{3}{4}\right)^3\left(\frac{1}{2}\right)^3\right] \cdot 2^3 + \frac{351}{512} \cdot 2^2 \approx 2^{2.43}$$

Filtering at the second-to-last round. Assuming $n_t = 3$, the initial candidate space for a key group is $(2^{2.43})^4 = 2^{9.72}$. Guessing a single group key in the last round enables key filtering for four nibbles in the second-to-last round. Since the DDT entry for a non-LS difference is 8, the probability that a specific key candidate satisfies the differential condition is 1/2. This corresponds to the baseline of 1-bit filtering.

The number of group-determined nibbles is 4, and the total number of groups is 8. Considering the probability of a nibble exhibiting a non-LS difference, which is 21.76/32, the average filtering bits per group is calculated as:

$$\frac{4 \times (21.76/32)}{8} \approx 0.34$$

bits.

For the remaining 28 non-group-determined nibbles, the currently guessed last round group key is discarded only if all nibble key candidates at the first-to-last round fail to satisfy the difference condition. When the nibble key candidate space is 2^2 (probability 351/512), exactly one key group remains. Thus, the elimination probability is 1/2, yielding full 1-bit filtering effectiveness:

$$\frac{28 \times (21.76/32) \times (351/512)}{8} \approx 1.63$$

bits. When the candidate space is 2^3 (probability 153/512), two key groups remain. For the group key to be filtered, both remaining groups must be invalid. The probability of this simultaneous elimination is $(1/2) \times (1/2) = 1/4$. Since this is half the baseline elimination probability (1/2), we apply a weighting factor of 1/2 to the filtering:

$$\frac{28 \times (21.76/32) \times (153/512) \times (1/2)}{8} \approx 0.36$$

bits. Finally, when the candidate space is 2^4, four key groups exist. Since it is impossible for all four groups to fail simultaneously, no filtering occurs in this case.

Consequently, a total average of 2.33 bits are filtered per group key guess. Considering the structural lower bound at the group level, the number of remaining candidate keys after using n_t trails is given by $\max(2^4, 2^{9.72-2.33n_t})$.

Filtering at the third-to-last round. Since a half consists of four groups, considering the structural lower bound of each group (2^4), the initial candidate space for a half-key at the last round is $\max(2^{16}, 2^{38.88-9.32n_t})$. A single half-key guess allows for filtering across 16 nibbles at the third-to-last round. We apply the same logic as in the previous round, using the non-LS difference probability of 16.25/32 as calculated in the previous section.

The 32 nibbles involved are categorized based on their dependency on the half-key. For the eight *fully-determined* nibbles, both the group key and the nibble key are fixed. Since the key is unique, the elimination probability is 1/2. Thus, the average filtering per half is:

$$\frac{8 \times (16.25/32)}{2} \approx 2.03$$

bits.

For the eight *group-determined* nibbles, the group key is determined, but the nibble key depends on the candidate keys surviving from previous steps. The currently guessed half-key is discarded only if all nibble key candidates fail to satisfy the difference condition. Specifically, when the nibble key candidate space is 2^2 (probability 351/512), exactly one key group remains, resulting in an elimination probability of 1/2 and contributing approximately 1.39 bits ($= \frac{8\times(16.25/32)\times(351/512)}{2}$). When the candidate space is 2^3 (probability 153/512), two key groups remain. Since both key groups must fail simultaneously for the half-key to be filtered (probability 1/4), we apply a weighting factor of 1/2, resulting in a filtering effect of approximately 0.30 bits ($= \frac{8\times(16.25/32)\times(153/512)\times(1/2)}{2}$). In the case where the candidate space is 2^4, four key groups remain; since it is practically impossible for all four to fail simultaneously, this case contributes no filtering.

Finally, for the remaining 16 nibbles (classified as nibble-determined or non-determined), the group key is not determined by the half-key guess, effectively making the candidate space 2^{16}. Since the probability that all candidates in such a large space fail to satisfy the difference condition is negligible, we exclude this potential filtering from our complexity calculation.

Thus, a total average of 3.72 ($= 2.03 + 1.39 + 0.30$) bits are filtered per half-key guess. The final remaining key space is $2^{38.88-9.32n_t-3.72n_t} = 2^{38.88-13.04n_t}$. As discussed in Section 4.1, upon reaching the half level, the key candidates can structurally converge to a unique solution (2^0).

Filtering at the preceding rounds. For filtering at the fourth-to-last round, the full key is guessed by combining the two halves. At this stage, we observe an average of 7.5 active S-boxes with non-LS differences. Since each non-LS difference provides 1 bit of filtering as established previously, the candidate space reduces to $(2^{38.88-13.04n_t})^2 \cdot 2^{-7.5n_t} = 2^{77.76-33.58n_t}$. Considering the additional 3 and 1 non-LS differences occurring at the fifth- and sixth-to-last rounds respectively, the final number of remaining key candidates is:

$$2^{77.76-33.58n_t-3n_t-1n_t} = 2^{77.76-37.58n_t}$$

For $n_t = 2$, the remaining candidate space is approximately $2^{2.6}$, which implies that while the candidates are reduced to a very small number (approx. 6), it is insufficient for unique key recovery. However, using $n_t = 3$ reduces the space to $2^{77.76-37.58\times3} = 2^{-34.98}$. As this value is significantly less than 1, only the correct key remains. Therefore, injecting three faults is sufficient to recover the entire key, a result we have also verified experimentally.

4.3.2 7-Round DFA on `DEFAULT` with Simple Key Schedule

The complexity of the 7-round DFA is calculated in the same manner as the 6-round DFA.

Filtering at the first-to-last round. For $n_t = 2$, the average number of remaining key candidates per nibble is calculated as:

$$\frac{1}{16} \cdot 2^4 + \left[2\left(\frac{1}{4}\right)\left(\frac{3}{4}\right) + \left(\frac{3}{4}\right)^2\left(\frac{1}{2}\right)^2\right] \cdot 2^3 + \left(1 - \frac{1}{16} - \frac{33}{64}\right) \cdot 2^2 \approx 2^{2.77}$$

Filtering at the second-to-last round. For the four group-determined nibbles, an average of $4 \times (24/32)/8 = 0.375$ bits are filtered per group. For the 28 non-group-determined nibbles, the average filtering is:

$$\frac{28 \times (24/32) \times (33/64)}{8} + \frac{28 \times (24/32) \times (27/64) \times (1/2)}{8} \approx 1.91$$

bits per group. Thus, a total of 2.285 ($\approx 0.375 + 1.91$) bits are filtered, reducing the candidate space to $\max(2^4, 2^{11.08-2.285n_t})$.

Filtering at the third-to-last round. With a non-LS difference probability of 21.76/32, the initial candidate space per half is $2^{44.32-9.14n_t}$. For the eight fully-determined nibbles, the average filtering is:

$$\frac{8 \times (21.76/32)}{2} \approx 2.72$$

bits per half. For the eight group-determined nibbles, the average filtering is:

$$\frac{8 \times (21.76/32) \times (33/64)}{2} + \frac{8 \times (21.76/32) \times (27/64) \times (1/2)}{2} \approx 1.98$$

bits per half. Thus, a total of 4.7 ($= 2.72 + 1.98$) bits are filtered, resulting in $2^{44.32-13.84n_t}$ candidates.

Filtering at the preceding rounds. The final key candidate space reduces to

$$2^{88.64-13.84 \cdot 2n_t - 27.75n_t} = 2^{88.64-55.43n_t}.$$

Substituting $n_t = 2$ yields $2^{88.64-110.86} = 2^{-22.22}$. Since this value is less than 1, injecting two faults is sufficient to recover the entire key.

4.3.3 8-Round DFA on `DEFAULT` with Simple Key Schedule

The complexity of the 8-round DFA is calculated similarly, but using the `0x1` input difference.

Filtering at the first-to-last round. For $n_t = 2$, the average number of remaining key candidates per nibble is approximately $2^{2.77}$.

Filtering at the second-to-last round. The remaining key candidate space per group is $2^{11.08-2.285n_t}$.

Filtering at the third-to-last round. For the eight fully-determined nibbles, the average filtering is:

$$\frac{8 \times (23.4/32)}{2} \approx 2.925$$

bits per half. For the eight group-determined nibbles, the average filtering is:

$$\frac{8 \times (23.4/32) \times (33/64)}{2} + \frac{8 \times (23.4/32) \times (27/64) \times (1/2)}{2} \approx 2.125$$

bits. Thus, a total of 5.05 ($\approx 2.925 + 2.125$) bits are filtered, reducing the candidate space to $2^{44.32-14.19n_t}$.

Filtering at the preceding rounds. The final key candidate space is reduced to

$$2^{88.64-14.19\cdot 2n_t-37.148n_t} = 2^{88.64-65.528n_t}.$$

Substituting $n_t = 2$ yields $2^{88.64-131.056} = 2^{-42.416}$. Since this value is less than 1, injecting two faults is sufficient to recover the entire key.

To validate these theoretical complexity estimates, we conducted a large-scale simulation. The detailed experimental results regarding the average number of remaining candidate keys at each filtering step are provided in Appendix A.

4.4 Improved DFA on DEFAULT with Rotating Key Schedule

In DEFAULT, the rotating key schedule uses four independent round keys repeatedly. Nageler et al. [NDE22] demonstrated that due to this key schedule and the properties of the LS S-box, if (k_0, k_1) is the correct key, an equivalent key pair $(\hat{k_0}, \hat{k_1})$ capable of performing the same decryption also exists. Consequently, the S-box of the DEFAULT-LAYER has 4 equivalent keys, the round function implies 2^{64} equivalent keys, and the entire DEFAULT-LAYER with the rotating key schedule possesses $(2^{64})^3 = 2^{192}$ equivalent keys.

The attack on DEFAULT with a rotating key schedule, as presented by Jana et al. [JKP24], consists of the following three steps:

Step 1: Reduction to Equivalent Key Space. This step reduces the key candidate space over four rounds, corresponding to the four independent keys used in the rotating key schedule. For the first three keys encountered in the reverse direction (e.g., $\hat{k}_3, \hat{k}_2, \hat{k}_1$), we utilize differential trails with a single solution to filter the key space, followed by reducing it to a normalized representative $\bar{k}_i$. For instance, if the candidate set is {0x0, 0x5, 0xa, 0xf}, the normalized representative is $\bar{k}_i =$ 0x0. Partial decryption is performed using this $\bar{k}_i$ to proceed to the next round. For the $\hat{k}_0$, the candidate space is filtered to 2^{64} but not normalized. Consequently, the combination of the normalized keys $\bar{k}_1, \bar{k}_2, \bar{k}_3$ and the 2^{64} candidates for $\hat{k}_0$ forms the equivalent key space.

Step 2: Identification of the Unique Key. In this step, faults are injected at a round corresponding to twice the number of rounds in Step 1. Partial decryption is performed using the normalized keys $\bar{k}_i$ obtained in Step 1 to identify a differential trail with a single solution. Using this trail, the candidate space for $\hat{k}_0$ is filtered from 2^{64} down to a unique value. Consequently, the multiple equivalent key spaces collapse into a single unique key space.

Step 3: Recovery of the Original Key. Finally, faults are injected into the DEFAULT-CORE to recover the master key from the identified equivalent key.

We apply the same optimization approach used for the simple key schedule to this rotating key schedule scenario to minimize the number of required faults. Specifically, by utilizing an MILP solver to identify differential trails with a single solution for 6-, 7-, and 8-round DFAs, we significantly reduce the fault requirements for Steps 1 and 2.

For Step 1, due to the diffusion properties of the cipher, a deterministic trail tends to exhibit more non-LS differences as it propagates through more rounds. This implies that if the key space can be reduced to 2^{64} with high probability at the fourth-to-last round, the same reduction can be achieved with even higher probability at subsequent rounds (first- to third-to-last). Therefore, our analysis focuses on calculating the required number of fault injections to achieve reduction at the fourth-to-last round.

For Step 2, identifying the key within the equivalent key space requires injecting additional faults at significantly earlier rounds (i.e., twelfth-, fourteenth-, and sixteenth-to-last rounds for the 6-, 7-, and 8-round attack scenarios, respectively). Attacking such deep

rounds presents a problem analogous to the simple key schedule attack. Consequently, it is sufficient to inject only 3, 2, and 2 additional faults for each respective scenario to uniquely determine the equivalent key.

4.4.1 6-Round DFA on DEFAULT with Rotating Key Schedule

For a fault injected at position 0x2 at the sixth-to-last round, the probability of obtaining a non-LS difference at the fourth-to-last round is 7.5/32. The probability that the key space fails to reduce to 2^2 (which occurs when non-LS differences appear at most once, or appear multiple times but yield redundant information) can be modeled using the binomial distribution as:

$$\left(\frac{24.5}{32}\right)^{n_t} + n_t \cdot \left(\frac{7.5}{32}\right) \cdot \left(\frac{24.5}{32}\right)^{n_t-1} + \sum_{i=2}^{n_t} \binom{n_t}{i} \left(\frac{7.5}{32}\right)^{i} \cdot \left(\frac{24.5}{32}\right)^{n_t-i} \cdot \left(\frac{1}{2}\right)^{i}$$

We determined the minimum number of trails, n_t, such that this failure probability is less than 5%, thereby guaranteeing a success rate of over 95%. It is important to note that the final fault count derived from this n_t represents the total number of random fault injections required to obtain these effective trails with high probability. Our analysis confirms that starting from $n_t = 25$, the key space is reduced to 2^2 with a probability exceeding 95%. Including the three faults required for Step 2, a total of 28 faults are sufficient to recover the key in the 6-round attack scenario.

4.4.2 7-Round DFA on DEFAULT with Rotating Key Schedule

For a fault injected at position 0x2 at the seventh-to-last round, the probability of a non-LS difference appearing at the fourth-to-last round is 16.25/32. Similar to the 6-round attack, we confirmed that starting from $n_t = 11$, the key space of all nibbles reduces to 2^2 with a probability greater than 95%. Including the two faults required for Step 2, a total of 13 faults are sufficient to recover the equivalent key in the 7-round attack scenario.

4.4.3 8-Round DFA on DEFAULT with Rotating Key Schedule

For a fault injected at position 0x1 at the eighth-to-last round, the probability of a non-LS difference appearing at the fourth-to-last round is 19.27/32. Similar to the 6- and 7-round attacks, we confirmed that starting from $n_t = 9$, the key space of all nibbles reduces to 2^2 with a probability exceeding 95%. Including the two faults required for Step 2, a total of 11 faults are sufficient to recover the equivalent key in the 8-round attack scenario.

5 Discussion and Conclusion

Before concluding, we discuss the scalability and general applicability of our proposed framework. Regarding the fault model, our primary analysis relies on the single bit-flip assumption. We evaluated the framework under broader physical assumptions, including nibble, byte, and random multi-bit faults. However, empirical results indicated that such faults cause differences to propagate with significantly higher diffusion. This rapid propagation yields an unmanageable number of feasible solutions in the MILP model, making it practically infeasible to obtain a differential trail with a single solution for deeper rounds. Therefore, we conclude that the single bit-flip fault is the optimal injection model for DFA methodologies that exploit differential trails with a single solution.

Furthermore, regarding the applicability to other structural designs, our framework is generically applicable to any S-box. Nevertheless, a fundamental trade-off exists when targeting non-LS S-boxes. Ciphers utilizing non-LS S-boxes inherently exhibit superior

differential properties, which trigger stronger key filtering power but drastically reduce the probability that a differential trail with a single solution exists over multiple rounds. Consequently, our framework based on differential trails with a single solution is less suitable for ciphers employing non-LS S-boxes, as finding such trails becomes inherently difficult.

In this paper, we proposed MIFA, a new framework that utilizes an MILP solver to significantly enhance the efficiency of DFA attacks. By systematically identifying differential trails with a single solution in deeper rounds, our approach enables successful attacks on rounds that were previously considered difficult to reach. As the number of attackable rounds increases, the efficiency of key filtering improves substantially, leading to a natural reduction in the required number of faults. We also emphasized that targeting these deeper rounds is of significant practical value, as countermeasures are often concentrated on the last few rounds.

Furthermore, we investigated the key filtering efficiency for all possible bit-flip faults. We selected the optimal bit position by comprehensively balancing the trade-off between key filtering efficiency and the probability of finding a differential trail with a single solution. A significant contribution of our work is the precise theoretical calculation of the required number of faults, achieved by proving that the key candidate space for LS S-box-based ciphers always reduces in the form of 2^k and by identifying the structural lower bound of key candidates during filtering at the second-to-last round.

Consequently, we achieved state-of-the-art results on DEFAULT. For DEFAULT with the simple key schedule, we demonstrated that 6-, 7-, and 8-round DFAs can be performed with only 3, 2, and 2 faults, respectively. For the rotating key schedule, our attacks require totals of 28, 13, and 11 faults for the same respective rounds. These results represent the optimal efficiency achievable among methods relying on differential trails with a single solution. Furthermore, we successfully applied our framework to BAKSHEESH, extending the attack range to 5 rounds; however, since these results are less efficient compared to the recently proposed ToFA [KGAS25], we omitted the detailed key recovery procedures from this paper.

Acknowledgements

We would like to express our sincere gratitude to the anonymous reviewers and the shepherd for their insightful comments and constructive feedback, which significantly improved the quality of this manuscript. This research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 25411243) and in part by the 2025 Research Grant from Kangwon National University.

References

[ADM+10] Michel Agoyan, Jean-Max Dutertre, Amir-Pasha Mirbaha, David Naccache, Anne-Lise Ribotta, and Assia Tria. How to flip a bit? In *2010 IEEE 16th International On-Line Testing Symposium*, pages 235–239. IEEE, 2010.

[Bak22] Anubhab Baksi. DEFAULT: Cipher-level resistance against differential fault attack. In *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*, pages 177–216. Springer, 2022.

[BBB+10] Alessandro Barenghi, Guido M Bertoni, Luca Breveglieri, Mauro Pellicioli, and Gerardo Pelosi. Fault attack on AES with single-bit induced faults. In

2010 Sixth International Conference on Information Assurance and Security, pages 167–172. IEEE, 2010.

[BBC+23] Anubhab Baksi, Jakub Breier, Anupam Chattopadhyay, Tomáš Gerlich, Sylvain Guilley, Naina Gupta, Takanori Isobe, Arpan Jati, Petr Jedlicka, Hyunjun Kim, et al. BAKSHEESH: Similar yet different from GIFT. *Cryptology ePrint Archive*, 2023.

[BGN05] Eli Biham, Louis Granboulan, and Phong Q Nguyen. Impossible fault analysis of RC4 and differential fault analysis of RC4. In *International Workshop on Fast Software Encryption*, pages 359–367. Springer, 2005.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present: Towards reaching the limit of lightweight encryption. In *International conference on cryptographic hardware and embedded systems*, pages 321–345. Springer, 2017.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Annual international cryptology conference*, pages 513–525. Springer, 1997.

[CCF+21] Tingting Cui, Shiyao Chen, Kai Fu, Meiqin Wang, and Keting Jia. New automatic tool for finding impossible differentials and zero-correlation linear approximations. *Science China Information Sciences*, 64(2):129103, 2021.

[Cla07] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 181–194. Springer, 2007.

[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*, pages 547–572, 2018.

[DMN+12] Jean-Max Dutertre, Amir-Pasha Mirbaha, David Naccache, Anne-Lise Ribotta, Assia Tria, and Thierry Vaschalde. Fault round modification analysis of the advanced encryption standard. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 140–145. IEEE, 2012.

[FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on AES with faulty ciphertexts only. In *2013 workshop on fault diagnosis and tolerance in cryptography*, pages 108–118. IEEE, 2013.

[HGSE24] Hosein Hadipour, Simon Gerhalter, Sadegh Sadeghi, and Maria Eichlseder. Improved search for integral, impossible differential and zero-correlation attacks: application to Ascon, ForkSKINNY, SKINNY, MANTIS, PRESENT and QARMAv2. *IACR Transactions on Symmetric Cryptology*, 2024(1):234–325, 2024.

[HSE23] Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 128–157. Springer, 2023.

[JKP24] Amit Jana, Anup Kumar Kundu, and Goutam Paul. More vulnerabilities of linear structure sbox-based ciphers reveal their inability to resist DFA. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 168–203. Springer, 2024.

[KDK+14] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. *ACM SIGARCH Computer Architecture News*, 42(3):361–372, 2014.

[KGAS25] Anup Kumar Kundu, Shibam Ghosh, Aikata Aikata, and Dhiman Saha. ToFA: Towards fault analysis of GIFT and GIFT-like ciphers leveraging truncated impossible differentials. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(3):614–643, 2025.

[KGSR23] Anup Kumar Kundu, Shibam Ghosh, Dhiman Saha, and Mostafizar Rahman. Divide and rule: DiFA-division property based fault attacks on PRESENT and GIFT. In *International Conference on Applied Cryptography and Network Security*, pages 89–116. Springer, 2023.

[MDH+13] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 77–88. IEEE, 2013.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *International Conference on Information Security and Cryptology*, pages 57–76. Springer, 2011.

[NDE22] Marcel Nageler, Christoph Dobraunig, and Maria Eichlseder. Information-combining differential fault attacks on DEFAULT. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 168–191. Springer, 2022.

[O'F16] Colin O'Flynn. Fault injection using crowbars on embedded systems. *Cryptology ePrint Archive*, 2016.

[OGSM15] Sébastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine. EM injection: Fault model and locality. In *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 3–13. IEEE, 2015.

[SBHS15] Bodo Selmke, Stefan Brummer, Johann Heyszl, and Georg Sigl. Precise laser fault injections into 90 nm and 45 nm SRAM-cells. In *International Conference on Smart Card Research and Advanced Applications*, pages 193–205. Springer, 2015.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 158–178. Springer, 2014.

[ST17] Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects: Revealing structural properties of several ciphers. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 185–215. Springer, 2017.

[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 Transactions on Cryptographic Hardware and Embedded Systems*, pages 150–172, 2018.

A Experimental Verification of Candidate Key Reduction

To verify the theoretical key recovery complexity presented in Section 4.3, we conducted a large-scale simulation based on the 8-round attack scenario for `DEFAULT` with a simple key schedule. The goal of this experiment is to measure the average number of remaining candidate keys at the nibble, group, half, and state levels, which are updated at each filtering step.

A.1 Experimental Setup and Data Selection

We generated 1,010 random master keys and performed attacks by injecting 4 faults for each key. Although our theoretical analysis indicates that key recovery is feasible with fewer faults, we employed 4 faults in this experiment solely to expedite the experiment.

As discussed in Section 3.1, randomly injected faults do not always yield a differential trail with a single solution. Our results showed that in 8 out of 1,010 tests, none of the four injected faults yielded a differential trail with a single solution. While this issue could be resolved in a real attack scenario by injecting additional faults, we excluded these 8 cases because the primary objective of this experiment is to statistically analyze the reduction of key candidates specifically when a deterministic trail is successfully formed. Consequently, we performed our statistical analysis on the 1,002 valid test cases where a unique solution was identified.

A.2 Experimental Results

For the 1,002 valid experiments, we measured the average number of remaining key candidates at each recovery step. The results, expressed in the form of 2^k, are as follows:

- **Step 1 (Nibble Level):** This step involves guessing the last round key at the nibble level, with filtering performed at the first-to-last round. The experimental results indicated that the average number of remaining candidates per nibble was approximately $2^{2.18}$.
- **Step 2 (Group Level):** This step involves guessing the last round key at the group level, with filtering performed at the second-to-last round. Here, the average number of remaining candidates per group was approximately $2^{4.19}$.
- **Step 3 (Half Level):** This step involves guessing the last round key at the half level, with filtering performed at the third-to-last round. At this stage, the key candidate space reduced drastically, leaving an average of only $2^{0.42}$ candidates per half. This implies that in most cases, the key is uniquely determined or very few candidates remain.
- **Step 4 and Beyond (State Level):** This step involves guessing the last round key at the state level, with filtering performed at the fourth-to-last round and preceding rounds. The results showed that at Step 4 (Round 4), the average number of candidates converged to $2^{0.00} = 1$. This demonstrates that the unique master key was successfully recovered in all 1,002 test cases upon reaching this stage.

This experimental data aligns closely with our theoretical calculations, demonstrating that our proposed attack framework is highly effective for key recovery in practice.