DEFAULT: Cipher Level Resistance Against Differential Fault Attack

Anubhab Baksi[1], Shivam Bhasin[2], Jakub Breier[3], Mustafa Khairallah[1], Thomas Peyrin[1], Sumanta Sarkar[4], and Siang Meng Sim[5]

[1] Nanyang Technological University, Singapore
[2] Temasek Labs NTU, Singapore
[3] Silicon Austria Labs, Graz, Austria
[4] TCS Innovation Labs, India
[5] DSO National Laboratories, Singapore

`ANUBHAB001@e.ntu.edu.sg`, `sbhasin@ntu.edu.sg`, `jbreier@jbreier.com`, `mustafa.khairallah@ntu.edu.sg`, `thomas.peyrin@ntu.edu.sg`, `sumanta.sarkar@gmail.com`, `crypto.s.m.sim@gmail.com`

Abstract. Differential Fault Analysis (DFA) is a well known cryptanalytic technique that exploits faulty outputs of an encryption device. Despite its popularity and similarity with the classical Differential Analysis (DA), a thorough analysis explaining DFA from a designer's point of view is missing in the literature. To the best of our knowledge, no DFA immune cipher at an algorithmic level has been proposed so far. Furthermore, all known DFA countermeasures somehow depend on the device/protocol or on the implementation such as duplication/comparison. As all of these are outside the scope of the cipher designer, we focus on designing a primitive which can protect from DFA on its own. We present the first concept of cipher level DFA resistance which does not rely on any device/protocol related assumption, nor does it depend on any form of duplication. Our construction is simple, software/hardware friendly and DFA security scales up with the state size. It can be plugged before and/or after (almost) any symmetric key cipher and will ensure a non-trivial search complexity against DFA. One key component in our DFA protection layer is an SBox with linear structures. Such SBoxes have never been used in cipher design as they generally perform poorly against differential attacks. We argue that they in fact represent an interesting trade-off between good cryptographic properties and DFA resistance. As a proof of concept, we construct a DFA protecting layer, named `DEFAULT-LAYER`, as well as a full-fledged block cipher `DEFAULT`. Our solutions compare favourably to the state-of-the-art, offering advantages over the sophisticated duplication based solutions like impeccable circuits/`CRAFT` or infective countermeasures.

Keywords. differential fault attack, protection, SBox, differential attack, `DEFAULT`

1 Introduction

Fault Attacks (FA) are considered strong implementation threats rendering many ciphers vulnerable. Unlike classical cryptanalysis, which assumes no interference with the internal operations of a cipher, in the case of FA the attacker has more control over the device where the cipher is currently being executed. As a result, among other options, he is able to suddenly alter an external input to the device (such as voltage level, EM radiation, heat, etc.), forcing it to run under sub-optimal condition. This type of condition can result in incorrect (*faulty*) output from the device. This faulty output may then help the attacker to gain information about the secret key. FA gained much popularity among the security/cryptography researchers and has been deployed to analyze a variety of ciphers.

When it comes to analyzing symmetric key cryptographic primitives, the most popular choice for FA is generally the *Differential Fault Analysis* or *Differential Fault Attack* (DFA) [11]. DFA is very powerful: almost all (if not all) block ciphers which are considered secure with respect to classical attacks have been shown to be vulnerable to DFA. Note that, to the best of our knowledge, no cipher has yet been designed to have a natural DFA immunity, although there were no shortage of new cipher proposals or new DFA countermeasures in recent years.

The crux of this situation is, as we observe, a lack of theoretical results towards designing DFA-resistant primitives, akin to its classical counterpart, the *Differential Analysis* (DA). Cipher designers have been very careful to design DA resistant ciphers, but not much attention has been given to design a DFA-resistant cipher. Indeed, designing a DFA-resistant cipher looks like a very difficult task as the attacker has enormous power in this setting.

The usual DFA protections lie outside the domain of cipher design. At one end, some device/protocol level technique is used, while at the other end, duplication based protection is used (see Section 2.2 for more details). Duplication based countermeasures assume that the fault can alter the execution within a predesignated boundary. Thereafter, a comparison (which can be direct or with an error-detection code) between the two executions is used to detect a fault. Since device/protocol level solutions are beyond the control of the cipher designer, the best option to ensure DFA protection is duplication[6]. Given this scenario, our work analyzes this problem and proposes a new type of solution, which is able to ensure a non-trivial search complexity for the attack when using DFA, solely based on the cipher construction itself. We use the basic design strategy and components of the lightweight block cipher `GIFT-128` [8] and thus manage to keep our design within low-cost performance figures. Note that the DFA protection mechanism could be costly and that our design does not need duplication or any protocol level countermeasure, we believe our work opens up a new genre of low-cost DFA countermeasure.

Our Contributions. In this work, we attempt to explain DFA from a cipher designer's perspective. We raise the question: how far a cipher designer can go to protect his primitive against DFA? In order to tackle this problem, we explore the potential offered by the SBox, one of the basic building blocks of symmetric-key cryptography algorithms. In the Substitution-Permutation Network (SPN) block

[6]It may still be argued that duplication based protections cannot be guaranteed at the cipher design level, and hence off-limit to a cipher designer.

ciphers (and even in some f-function of Feistel network block ciphers), this is normally the only non-linear component, and it is naturally vulnerable to DFA (as DFA does not work on a linear component, whereas it works very well on a non-linear one). In a nutshell, strong linearity makes it hard to attack a cipher with DFA, but too much linearity will of course render a cipher either insecure or not efficient. The goal is therefore to try to find a good trade-off.

Since a secure cipher cannot be constructed by using only linear components, we naturally focus on finding a building block that are somewhat in the middle ground between an SBox and a linear function. The middle ground, unsurprisingly, lies in a weak class of SBoxes, whose members behave like a linear function in some aspects. Such SBoxes have properties which are generally considered undesirable for a cipher construction. For example, (non-trivial[7]) *Linear Structures*[8] (LS) that we define in Definition 4 are something a cipher designer typically will try to avoid. More relevant discussion on this topic can be found in Section 3.

This leads to a paradoxical situation: an SBox which is more resistant against differential attacks is weaker against DFA, while those which are more resistant against DFA are considered weaker against differential attacks.

To circumvent this situation, we propose to keep the main cipher to be protected (which is presumably secure against classical attacks) as it is, but adding two keyed permutations L_1 and L_2 as additional layers before and after it, respectively. These keyed permutations will ensure a special structure to make DFA non-trivial, and this will render the entire construction DFA resistant. Indeed, assuming a certain fault model for DFA (see Section 2.3), the attacker has to attack the first or last rounds of the overall cipher to make the attack work. Note that when DFA security for either encryption-only or decryption-only circuit is desired, then only a single keyed permutation layer is needed. At the same time, the classical security of the construction, which is guaranteed by the main cipher, will not be hampered (refer to Section 4 for more discussion).

To validate our claims, we propose an SPN-based construction of a 128-bit keyed permutation L ($= L_2 = L_1$), `DEFAULT-LAYER`, using a 4×4 SBox. However, the novelty is that this SBox will contain 3 (non-trivial) LS. We show that this keyed permutation can provide safeguard against DFA up to a non-trivial search complexity ($2^{n/2}$ for an n-bit block cipher). `DEFAULT-LAYER` is hardware/software friendly and any variant of L with a multiple of 16-bit can be constructed (we recommend it to be at least 128-bit). As the DFA security scales up with the size of L (which does not happen for ciphers like `AES`), if a 256-bit variant of L is used it will effectively provide a DFA security of (at least) 2^{128} computations. In fact, by properly choosing the SBox, it is possible to go beyond $2^{n/2}$ security (see Table 1).

The idea of our keyed permutation `DEFAULT-LAYER` further leads to a complete SPN-based cipher `DEFAULT`, hoping for improved performances. It uses

[7]Non-triviality here refers to the non-zero linear structures, since zero is always a linear structure.

[8]It is to be noted that the concept of "linearity" in the context of the classical linear cryptanalysis is different from linear structures.

DEFAULT-LAYER as a component, while the other component (which does not have security against DFA) is named DEFAULT-CORE, which is sandwiched between two DEFAULT-LAYER instances. More discussion on these constructions can be found in Section 4 with the design rationale explained in Section 5. Furthermore, detailed security analyses against classical attacks, DFA, DFA-assisted classical attacks, and side-channels attacks can be found in Section 6.

The benchmarking results are given in Section 8. We choose duplication (either in the spatial or the temporal domain) as a reference countermeasure for benchmarking as duplication is a widely adopted fault protection method in commercial products. DEFAULT incurs similar overheads when compared to duplicated GIFT-128, both in hardware and software. As described later, DEFAULT resists a much wider class of fault models as compared to duplication, thus giving it an advantage. In retrospect, our solution can be considered lightweight compared to more sophisticated duplication countermeasures (such as infection or error-detecting codes). For instance, when compared to the recent block cipher CRAFT [10], CRAFT leads to a $2.45\times$ overhead when protecting against single bit faults at the output and scales higher for protecting against more faults. CRAFT is proposed as a block cipher with fault protection as a prime target, designed with carefully chosen components that incur lower overhead when protected with error detection codes. DEFAULT, on the other hand, explores an alternative methodology to design a cipher with natural DFA resistance and is not limited to a specific number of faults. Similarly, infective countermeasures can have $\approx 3\times$ cost increase when compared to the basic implementation [7].

As an independent contribution, we also study how to model a cipher which has an SBox with linear structures when searching for differential and linear bounds using automated tools. This is described in Section 7.

2 Background and Preliminaries

2.1 Differential Fault Attacks in a Nutshell

As already mentioned, DFA is closely related to DA. In a classical DA, a difference is introduced in plaintexts (resp., ciphertexts) at the beginning of the cipher encryption (resp. decryption). Detecting the expected output difference requires large amount of data, where the data complexity is inversely proportional to the differential probability. Cipher designers often prove security against DA by showing that the probability of any differential trail is too low for launching a DA.

In comparison, in DFA the input difference is inserted in the form of a transient fault and can be applied anytime during the course of the encryption/decryption. In practice, faults are injected near the end of the cipher execution, effectively bypassing most of the rounds designed to resist DA when compounded. This difference propagates through only a handful of non-linear components, and based on the output differential value, the adversary is able to reduce the key search space significantly.

The cryptanalysis procedure in DFA consists of two orthogonal terms, namely, fault complexity (the number of faulty encryptions) and search complexity (computational/memory complexity required). The general trend is to reduce the fault complexity while keeping the search complexity within an acceptable limit.

2.2 Differential Fault Attack Protections

The state-of-the-art DFA countermeasures can be broadly classified into the following categories [5]:

1. A separate, dedicated device that detects (and takes precaution) [21] or a shield that blocks any potential source of a fault.
2. The underlying communication protocol between Alice and Bob ensures that a fault does not occur with a significant probability. This can be ensured, e.g., by assuming a small portion of the circuit is protected by other means [6].
3. Duplicate the cipher execution followed by implicit/explicit check for the equality of the executions, so-called *duplicated computations*. One may refer to [7] for a study of such countermeasures.
4. Use mathematical solutions to render DFA ineffective/inefficient.

One may notice that the countermeasures in above-mentioned categories 1 and 2 are basically engineering solutions and generally outside the scope of cryptography design. In a slight contrast, category 3 is somewhat close to what a cipher designer can specify. Yet, identical faults in the duplicated computations will result in no differences between the outputs and treated as if no fault is injected. This tricks the countermeasure to release the faulty output, and works against state-of-the-art countermeasures like infection [7]. Although relatively hard to achieve in practice, this type of attack was shown to be feasible in [33] and we refer to it as *duplicate fault*. While the device could be protected by using different encodings for the two executions of the cipher, any such method would add additional performance cost. We also mention that sophisticated duplication countermeasures may require additional components as well as an external source of randomness [7].

Our work falls under category 4, together with countermeasures like impeccable circuits [2]/`CRAFT` [10] and `FRIET` [34]. The authors of [2] proposed an efficient DFA protection mechanism based on error detection codes and this idea was later extended to a block cipher, named `CRAFT`. `CRAFT` employs error detection codes, which have different performance figures and fault coverage depending on the underlying code. Any fault injection that successfully alters the output beyond the detectable bound will make the DFA protection of `CRAFT` ineffective. In comparison, our construction is free from such limitation (more details in Section 6.4).

2.3 Our Claim

Novel Idea against DFA. At a higher level, most of the countermeasures, including `CRAFT` and duplicated computation, aim at *fault detection* which could be fooled by stronger equipment that makes the faults go undetected. In comparison, we aim at *fault resilience*[9], meaning we allow the faults to propagate and even

[9]The term “fault injection resilience” is first introduced in [20].

output faulty ciphertexts, but the amount of information that an adversary can learn from them is limited: *we impose a lower bound on the search complexity of DFA*. Even with stronger equipment access, an adversary cannot overcome the lower bound of the search complexity. In addition, our design is completely at the algorithmic level, scalable, can be applied to existing ciphers and does not require an additional source of randomness.

Analysis Methods. Instead of enumerating the various fault models and fault attacks, we consider how an attack gains sensitive information, i.e. the analysis method. We can broadly categorise the analysis methods into two types:

1. Deduce information from the differential values of the executions.
2. Deduce information from the statistical bias of the executions.

The fundamental reason why our design increases the search complexity of DFA is due to the larger number of solutions for any given differential (details in Section 3). Hence, for attacks that gain information from the differential values (analysis method 1), it is not going to be as effective. We believe that our design could actually provide protection beyond DFA. In a broader sense:

Our design can protect against DFA and any form of FA that deduces information from the differential values of the executions.

Other attacks that exploit information leakages from statistical biases under analysis method 2 are beyond our focus. We provide more discussions in Section 6.5.

2.4 Difference Distribution Table and Related Properties

A *Difference Distribution Table* (DDT) is an analysis table used in DA. For an $n \times n$ SBox S, it is basically the $2^n \times 2^n$ matrix, where the row δ $(= 0, 1, \ldots, 2^n - 1)$ and column Δ $(= 0, 1, \ldots, 2^n - 1)$, denoted as $\mathrm{DDT}_S[\delta, \Delta]$, stores the number of solution(s) x for $S(x) \oplus S(x \oplus \delta) = \Delta$. Notice that $\mathrm{DDT}_S[0, 0] = 2^n$ as $S(x) \oplus S(x \oplus 0) = 0$ holds for all x. The maximum entry at the DDT of S, except the case $\delta = \Delta = 0$, is called the *Differential Uniformity* (DU).

In order for an SBox to be better resistant against DA, the (non-zero) maximal values in the DDT have to be small, otherwise DA will be more effective. Thus, symmetric-key cryptography designers almost exclusively look for SBoxes which have smaller values in the DDT. However, the situation for DFA is completely opposite. Here, if the (non-zero) DDT values are small, then the attacker has fewer solutions for the unknown input when collecting faulty outputs. Thus, he is able to narrow down the search space more efficiently: a DDT with smaller (non-zero) values will make the DFA easier. Hence, we see that the strategy to thwart DA is exactly opposite to that of DFA. This paradoxical situation is among the challenges to build a cipher level DFA protection.

We call SBoxes S_1 and S_2 *Affine Equivalent* (AE) if there exist two affine permutations A_1 and A_2 such that $S_2 = A_1 \circ S_1 \circ A_2$. AE SBoxes have the same DDT up to a permutation. Therefore, differential uniformity is invariant under affine equivalence, so are the other cryptographic properties like non-linearity, algebraic degree, etc. It is to be noted that the affine equivalence classification of all 4×4 SBoxes has been completed already — there are 302 such classes. We follow the class representative SBoxes given in [15, Chapter 5.4.2]. For a more compact representation, an element $\alpha \in \mathbb{F}_2^n$ will be denoted by its corresponding integer value from $[0, 2^n - 1]$.

Definition 1 **($S_\alpha\langle\delta\rangle$).** *For the SBox S, the fault δ and the value α, the set of solutions of the equation $S(x) \oplus S(x \oplus \delta) = S(\alpha) \oplus S(\alpha \oplus \delta)$ is the set $S_\alpha\langle\delta\rangle$.*

Notice that, both α and $\alpha \oplus \delta \in S_\alpha\langle\delta\rangle$. Basically, the cardinality of $S_\alpha\langle\delta\rangle$ gives the entry of the DDT at the δ^{th} row which contains α, which is at the column $\Delta = S(\alpha) \oplus S(\alpha \oplus \delta)$. By applying fault δ, the attacker cannot identify α from other elements which belong to $S_\alpha\langle\delta\rangle$.

Definition 2 **($\mathbf{MinF}_S(\alpha)$).** *For an $n \times n$ SBox S with input α,*

$$MinF_S(\alpha) = \begin{cases} -1 & \text{if } \bigcap_{\delta=1}^{2^n-1} S_\alpha\langle\delta\rangle \neq \{\alpha\}; \\ t & \text{where } t = \min k \text{ such that } \bigcap_{i=1}^{k} S_\alpha\langle\delta_i\rangle = \{\alpha\}. \end{cases}$$

Hence $\mathrm{MinF}_S(\alpha) = -1$ means, no matter what fault values that an attacker chooses, he will be left with more than one choice for α. Also notice that, if $\mathrm{MinF}_S(\alpha) \neq -1$, then it must be ≥ 2.

Definition 3 **($\mathbf{MinF}_S$).** *Given an $n \times n$ SBox S, $MinF_S$ is defined as:*

$$MinF_S = \begin{cases} \max_{0 \leq \alpha \leq 2^n-1} MinF_S(\alpha) & \text{if } MinF_S(\alpha) \neq -1,\ \forall \alpha \in \{0, 1 \ldots, 2^n - 1\}; \\ -1 & \text{otherwise.} \end{cases}$$

The subscript S is dropped if understood from context.

The interpretation of MinF_S can be given as, given an SBox S, it is the lower bound number of faults required to uniquely solve any input.

Definition 4 **(Linear Structure).** *For $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, an element $a \in \mathbb{F}_2^n$ is called a linear structure of F if for some constant $c \in \mathbb{F}_2^n$, $F(x) \oplus F(x \oplus a) = c$ holds $\forall x \in \mathbb{F}_2^n$.*

Note that the set of all linear structures of F denoted as $\mathcal{L}(F)$ forms a subspace of $\mathbb{F}_2^n$ and is termed as the *linear space* of F. If $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ has a (non-zero) linear structure then 2^n becomes an entry in the corresponding DDT. In that case $\mathrm{DU} = 2^n$, thus F performs worst against differential attacks compared to all F's that do not have a (non-zero) linear structure.

Definition 5 **(Coordinate Function and Component Function).** *Suppose $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is defined as $F(x) = (f_0(x), \ldots, f_{n-1}(x))$ for all $x \in \mathbb{F}_2^n$, where $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ for $i = 0, \ldots, n-1$. Then each f_i is called a coordinate function of F. Furthermore, the linear combinations of f_i's are called the component functions of F.*

Definition 6 **(Non-linearity).** *The non-linearity of the Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is the minimum distance of f to the set of all affine functions. Furthermore, the non-linearity of $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is the minimum of the non-linearities of all the component functions of F.*

3 Characterizing SBoxes in View of DFA

From now on, we implicitly assume that neither δ or Δ is 0 and that an SBox S is a permutation. We denote $\Delta(\alpha, \delta)$ the output difference for input value α and input difference δ.

Theorem 1. *Let $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ have a solution $x = \alpha$. Further, let a be a (non-zero) linear structure of S. Then, $(\alpha \oplus a)$ is also a solution of $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$, i.e., the coset $\alpha \oplus \mathcal{L}(S)$ is a subset of solutions of $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$. So, $MinF_S = -1$.*

Proof. As a is a linear structure of S, we have that $S(x) \oplus S(x \oplus a)$ is constant. Taking derivative with respect to δ, $\forall x$ we get $S(x) \oplus S(x \oplus a) \oplus S(x \oplus \delta) \oplus S(x \oplus a \oplus \delta) = 0$. Using $x = \alpha$, $S(\alpha) \oplus S(\alpha \oplus \delta) \oplus S(\alpha \oplus a) \oplus S(\alpha \oplus a \oplus \delta) = 0 \implies S(\alpha \oplus a) \oplus S(\alpha \oplus a \oplus \delta) = S(\alpha) \oplus S(\alpha \oplus \delta) = \Delta(\alpha, \delta)$. Hence, $(\alpha \oplus a)$ is also a solution of $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$. □

Theorem 1 gives an interesting insight regarding DFA resistance in SBoxes. If an SBox has a (non-trivial) linear structure, then it is not possible to find the input to the SBox just by analyzing the effect of faults, no matter how many faults are injected. In such cases, the attacker has to search exhaustively among the set of solutions to find the proper input. This increases the search complexity associated with DFA.

Lemma 1 (Converse of Theorem 1). *For the input α to S, if $\alpha \oplus a$ is a solution of $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ for all input differences δ, then a $(\neq 0)$ is a linear structure of S.*

Remark 1. Theorem 1 and Lemma 1 are valid for all (non-trivial) linear structure(s) of S. In other words, the larger the number of (non-trivial) linear structures, the larger the number of candidates that will be in the intersection of solution sets of all faults.

Lemma 2. *Suppose S_1 and S_2 are two $n \times n$ SBoxes having ℓ_1 and ℓ_2 linear structures (including the trivial linear structure 0) respectively, then the $2n \times 2n$ SBox (S_1, S_2) will have $\ell_1 \ell_2$ linear structures (including the trivial linear structure $(0, 0)$).*

Lemma 3. *Suppose $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ to be any function and $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ to be linear. Then $L \circ F$ and F have the same number of linear structures.*

Theorem 2. *If $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ has only two solutions, then there exists a δ' such that α is the unique common solution for $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ and $S(x) \oplus S(x \oplus \delta') = \Delta(\alpha, \delta')$.*

Theorem 3. *Assume that the SBox S does not have any (non-zero) linear structure and that $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ has exactly $2m + 2$ solutions; where $2m + 2 \geq 4$. Then there exist $m + 1$ faults $\{\delta', \delta_1, \ldots, \delta_m\}$ such that the system of equations*

$$S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta), \qquad S(x) \oplus S(x \oplus \delta') = \Delta(\alpha, \delta'),$$
$$S(x) \oplus S(x \oplus \delta_1) = \Delta(\alpha, \delta_1), \quad \ldots, \; S(x) \oplus S(x \oplus \delta_m) = \Delta(\alpha, \delta_m)$$

has a unique solution. Hence, $MinF_S(\alpha) \leq m+2$.

From Theorem 3, we see that it is possible to uniquely recover the input/output value of each SBox with no more than $\frac{\mathrm{DU}_S}{2}+1$ faults (unless there is a linear structure) when attacking the last round. This gives a provable upper bound on the number of faults the attacker needs per SBox (if faults values are judiciously chosen) in order the find out its input uniquely, given that the SBox does not have a linear structure.

Corollary 1 (From Theorem 2, 3). $MinF_S \leq \dfrac{DU_S}{2}+1$.

Remark 2. Although it is theoretically possible, we could not find an SBox with $\mathrm{MinF}_S \geq 3$ (refer to Corollary 1). Whether or not this is a tight bound is left open for future research.

Due to space constraints, the proof for the Lemmas and Theorems can be found in the Supplementary Material (Section A), together with other relevant results and examples.

Remark 3. Lemma 2 and Lemma 3 give another interesting view-point: if an unkeyed SPN permutation is constructed by repeating an SBox with l LS m times (in each round), then the total number of linear structures for the super SBox (which is the round function) is l^m.

In order to better visualize the effect of DFA security with respect to the number of linear structures for SPN ciphers (for a given SBox size), we present detailed information in Table 1 for varying state sizes. Note that the last cases (i.e, a 4×4 SBox with 4 and an 8×8 SBox with 128 linear structures) is the theoretical limit for DFA protection (as any more LS would imply that the SBox is linear). Hence, in theory we can achieve DFA security up to 2^{64} (for a 128-bit state) or 2^{128} (for a 256-bit state) using 4-bit SBoxes; and 2^{112} (for a 128-bit state) or 2^{224} (for a 256-bit state) using 8-bit SBoxes. As a proof of concept, our instantiation of this DFA protection layer will use a 4-bit SBox with 4 LS (which can provide DFA security of 2^{64} computations) and it is described in Section 4.

4 Construction of DFA Resistant Layer and Cipher

With the background given in Section 2, we first look at the problem of maximizing the fault complexity. Note that fault complexity is the highest when the fault is injected at the last round. Usually, for an SPN block cipher, three faults per SBox are sufficient as most block ciphers use an SBox with DU = 4 (except for `GIFT` SBox [8], where DU is 6). In fact, in many cases, only two faults are needed to solve for any input. For example, for the SBoxes chosen in `AES` [28], `PRESENT` [12], `SKINNY-64` [9] and `GIFT` [8], the fault values $\{1, 6\}$ are sufficient to retrieve all inputs uniquely. Thus, it is possible to solve the last round of the above mentioned ciphers with a number of faults equal to just twice the number of SBoxes. It seems hard to force the fault complexity to increase.

Table 1: DFA security for SPN ciphers depending on the number of linear structures in the SBox. Our design `DEFAULT-LAYER` will use a 4×4 SBox with 4 linear structures for a state size of 128 bits, hence ensuring a 2^{64} DFA security.

(a) 4×4 SBox

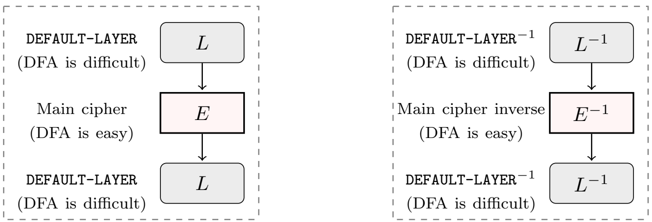
$\sharp$ LS	State Size	DFA Security
2	128	2^{32}
	256	2^{64}
4	128	2^{64}
	256	2^{128}

(b) 8×8 SBox

$\sharp$ LS	State Size	DFA Security
8	128	2^{48}
	256	2^{96}
64	128	2^{96}
	256	2^{192}
128	128	2^{112}
	256	2^{224}

4.1 Ad-hoc DFA Protection Layer (`DEFAULT-LAYER`)

Our approach is to tackle the problem of increasing the search complexity instead. This means that we give the attacker the power to apply as many faults as he wants in total, but the search space for the analysis should remain very large. As we already pointed out (Theorem 1), if an SBox S has non-zero linear structure(s), then the attacker will not be able to uniquely identify the input. Thus, he has to enumerate the remaining key candidates from the input difference – output difference relation.



(a) Encryption **(b)** Decryption

Fig. 1: Main cipher augmented by `DEFAULT-LAYER` to resist DFA

Now, using an SBox with a linear structure is generally considered undesirable for a block cipher design, as it makes the classical differential attacks easier (as explained in Section 2.4). Hence, we arrive at a paradoxical situation: if we want to design a cipher with better resistance against DFA, it becomes weak against classical attacks; and vice-versa. In order to find a middle ground, where the cipher is strong against both DFA and classical attacks, we propose the concept of prepending/appending an extra layer (that uses SBoxes with linear structures) to the underlying cipher (henceforth referred to as *"main cipher"*). Figure 1 visually represents the idea. The layer L, which we name as `DEFAULT-LAYER` and describe in Section 4.3, is prepended and appended to the main cipher E, as in Figure 1(a). For decryption, L^{-1} is both prepended and appended to the main cipher inverse (shown in Figure 1(b)), since the ciphertext $C = L \circ E \circ L(P)$, and the decryption $L^{-1} \circ E^{-1} \circ L^{-1}(C) = (L^{-1} \circ E^{-1} \circ L^{-1}) \circ (L \circ E \circ L)(P) = P$. The idea is that the underlying cipher E will have desirable protection against classical attacks, while the additional layer L will be used to thwart DFA. Since for DFA the attacker has to slowly peel off the outer rounds of the cipher, we only have to protect these rounds against DFA, while the inner cipher will provide all the security we expect from a block cipher in the black-box model (adding a layer L will not weaken its security).

As we assume the attacker can target both the encryption and decryption processes, the model described here can thwart DFA on both. If we assume a constrained model for the attacker, for example where the decryption is done at a server which is physically protected such that it cannot be accessed (as in [6, Section III]), then the prepended layer L in Figure 1(a) and the appended layer L^{-1} in Figure 1(b) can be removed, which will result in better performance.

4.2 Extension to a Full-Fledged Cipher (DEFAULT)

Aside from an ad-hoc layer which is able to protect any cipher from DFA, it is also possible to construct a full-fledged block cipher. This is done by sandwiching the so-called DEFAULT-CORE (this is another keyed permutation described in Section 4.4) with DEFAULT-LAYER. The DEFAULT-CORE contains an SBox that is especially resistant to classical linear attacks, and DEFAULT-LAYER uses an SBox that contains linear structures to resist DFA and its variants. Hence DEFAULT consists of 2 components (for both the encryption and decryption), as can be seen in Figure 1(a), replacing E with DEFAULT-CORE.

DEFAULT-CORE also follows a construction similar to GIFT-128, but we do not reuse GIFT-128 permutation exactly as core permutation because we want to maximize the security against linear attacks, even if that results in relatively low security against differential attacks (which will be partially provided by the DEFAULT-LAYER layers anyway). Thus, we do not use LS SBox, but in contrary we will use an SBox with excellent linear approximation table (LAT) properties.

Therefore, the advantage of using DEFAULT instead of simply a classical cipher protected with DEFAULT-LAYER layers, is that since DEFAULT-CORE has been designed to be especially strong against linear attacks, we can reduce the number of cryptographic operations globally. In other words, we believe DEFAULT strikes a better balance in terms of security/efficiency, while using a classical cipher with DEFAULT-LAYER probably comes with some performance overkill (DEFAULT-LAYER will provide extra differential attack resistance on top of the main cipher, which was not needed since the cipher is assumed to be secure already).

4.3 Construction of DEFAULT-LAYER

We detail the 128-bit version of our proposed DFA protecting layer (DEFAULT-LAYER). It can be used to protect 128-bit block ciphers, but we emphasize that it can be adapted to any block size that is a multiple of 16.

DEFAULT-LAYER is a 28-round keyed permutation[10] that receives a 128-bit message as the state $X = b_{127}b_{126}\dots b_0$, where b_0 is the least significant bit,

[10] We avoid calling it a "cipher" as it is a DFA protecting layer used on top of an actual cipher.

and a 128-bit layer-key from the master key. The state can also be expressed as $X = w_{31}\|w_{30}\|...\|w_0$, where w_i is a 4-bit nibble word. We do not describe the inverse layer here for the sake of brevity, but it can be trivially derived. The round function (denoted by $\mathcal{R}$ henceforth) of `DEFAULT-LAYER` consists of 4 steps (in order): `SubCells` — applying a 4-bit SBox to the state, `PermBits` — permute the bits of the state (same as in `GIFT-128` [8]), `AddRoundConstants` — XORing a 6-bit constant as well as another bit to the state (same as in `GIFT-128`), and `AddLayerKey` — XORing the layer-key to the state. A two consecutive rounds of `DEFAULT-LAYER` are depicted in Supplementary Material (Section C).

`SubCells`. It uses the 4-bit LS SBox $S =$ `037ED4A9CF18B265`. This SBox is applied to every nibble of the state: $w_i \leftarrow S(w_i)$, $\forall i \in \{0, \ldots, 31\}$.

`PermBits`. The bit-permutation is the same as the permutation P_{128} in `GIFT-128` (see Supplementary Material, Section C), which maps bits from bit position i of the internal state to bit position $P_{128}(i)$: $b_{P_{128}(i)} \leftarrow b_i$, $\forall i \in \{0, ..., 127\}$.

`AddRoundConstants`. A single bit "1" and a 6-bit round constant $\mathrm{C} = c_5c_4c_3c_2c_1c_0$ are XORed into the cipher state at bit position 127, 23, 19, 15, 11, 7 and 3 respectively: $w_{127} = w_{127} \oplus 1$, $w_{23} = w_{23} \oplus c_5$, $w_{19} = w_{19} \oplus c_4$, $w_{15} = w_{15} \oplus c_3$. Table 2 shows the round constants (6-bit) for `DEFAULT-CORE` and `DEFAULT-LAYER`. At each round the value is encoded into a 6-bit word and XORed to the cipher state, with c_0 being the least significant bit.

Table 2: Round constants for `DEFAULT`

	Round Constants	♯
`DEFAULT-CORE`	1, 3, 7, 15, 31, 62, 61, 59, 55, 47, 30, 60, 57, 51, 39, 14, 29, 58, 53, 43, 22, 44, 24, 48, 33, 2, 5, 11	28
`DEFAULT-LAYER`	1, 3, 7, 15, 31, 62, 61, 59, 55, 47, 30, 60, 57, 51, 39, 14, 29, 58, 53, 43, 22, 44, 24, 48	24

`AddLayerKey`. The layer-key k is bitwise XORed to the state: $b_i \leftarrow b_i \oplus k_i^j, \forall i \in \{0, ..., 127\}$.

`DEFAULT-LAYER` does not have a key schedule, the layer-key is extracted directly from the master key. If the secret key is more than 128-bit, simply extract the first 128 bits as the layer-key.

4.4 Construction of `DEFAULT-CORE` (and `DEFAULT`)

In order to design the full-fledged cipher, we need to describe the middle part of the cipher (`DEFAULT-CORE`), for which the SBox does not have any (non-zero) linear structure. The design of the core is much alike to the `DEFAULT-LAYER` (hence omitted here for the sake of brevity), except for the SBox, and it has 24 rounds. The SBox of choice here is `196F7C82AED043B5`, based on its very desirable cryptographic properties against linear attacks (see Section 7.2 for details). In a nutshell, the overall design of `DEFAULT` consists of: `DEFAULT-LAYER` (28 rounds), followed by `DEFAULT-CORE` (24 rounds), followed by another `DEFAULT-LAYER` (28 rounds). Hence `DEFAULT` is an SPN block cipher with heterogeneous round structure, consisting of 80 rounds. Therefore, in comparison with Duplicated `GIFT-128` (which contains 80 rounds in total), `DEFAULT` has the same number of rounds. As for the round counter, we use the same from `GIFT-128`, which is refreshed at the beginning of DEFAULT-LAYER/DEFAULT-CORE. More description (such as test vectors) for DEFAULT can be found in Supplementary Material (Section B).

Key Schedule. The 128-bit key (which is the same as the master key for the underlying cipher in case of the ad-hoc layer) is divided into 16-bit words: $k_7||k_6||\ldots||k_0$. Each round key is extracted first and then the next round key is produced by the following update: $k_7||k_6||\ldots||k_0 \leftarrow (k_7||k_6||\ldots||k_0) \ggg_{128} 20$ followed by $k_7 \leftarrow k_7 \ggg_{16} 1$; where $\ggg_x i$ represents an i-bit right rotation within a x-bit word.

5 Design Rationale

The goals of our DEFAULT-CORE/DEFAULT-LAYER designs are clear: (1) to protect against DFA, (2) applicable to different state sizes as well as to wide variety of symmetric key ciphers, and (3) simple and lightweight. During its design, various choices have been made and we discuss those here.

5.1 Design Philosophy

SPN vs Feistel network. Our first decision was to choose between SPN and Feistel network. Although implementing the inverse of Feistel construction is simple and does not require the inverse of its f-function, the non-linearity is introduced to only half of its state in each round and hence usually requires more rounds (though lighter rounds) to achieve the desired security margin. On the other hand, SPN introduces non-linearity to the entire state and thus requires lesser rounds in general. Study is also simpler, so we chose to start with SPN.

Bit Permutation vs Rotational-XOR Diffusion vs Word-mixing Diffusion. For SPN constructions, the diffusion layer is usually either a bit permutation (like in PRESENT and GIFT), a rotational-XOR layer (like in SMS4 [16], ASCON [18]), or a word-mixing diffusion (like in AES and SKINNY). Although the latter two provide a stronger diffusion, they can be costly in hardware and non-trivial to adopt to different block sizes as it might lead to quite different descriptions. In hardware, the bit permutation is basically free to implement as it consists simply of circuit wiring. Moreover, from the design strategy of GIFT, we see that a bit permutation can be adjusted to various state sizes. Therefore, we choose bit permutation over other choices of diffusion layer.

5.2 Structure of the DEFAULT PermBits

We recall here the structure of the PRESENT and GIFT bit permutations as this will be useful later to understand our security guarantees. There are essentially two levels of permutation within the PRESENT or GIFT bit permutation: the group mapping and the SBox grouping.

Group Mapping. The mapping of the output bits from a group of 4 SBoxes to another group of 4 SBoxes in the next round. This is the main difference between the PRESENT and GIFT permutation. For 4-bit SBoxes, we denote the 4 bits as bit 0, 1, 2 and 3, where bit 0 is the least significant bit. Within a group, the PRESENT permutation sends the 4 output bits from the i^{th} SBox (index from

0) to bit i of the 4 SBoxes in the next round, forming a symmetrical structure. Due to this symmetrical structure, PRESENT has many symmetrical differential characteristics for a given fixed input and output differences, which results in a higher differential probability (similar situation for the linear cryptanalysis case). On the other hand, the GIFT permutation sends bit i from the output of the j^{th} SBox (index from 0) to the bit i of the l^{th} SBox in the next round, where $l = i - j \mod 4$. Since bit i of an SBox output is always mapped to bit i of another SBox, it makes the analysis on the propagation of the differences easier and breaks the symmetry. Therefore, we choose GIFT group mapping.

SBox Grouping. The partitioning of the SBoxes into the groups of 4 SBoxes. The SBox grouping for the 64-bit block ciphers PRESENT and GIFT-64 are the same, and the designers of GIFT extended the idea to construct SBox grouping for 128-bit block size. Similar to [8], we denote the SBoxes in round i as $S_0^i, S_1^i, \ldots, S_{g-1}^i$, where $g = n/4$ for block size n. These SBoxes can be grouped in 2 different ways - the Quotient Q and Remainder R groups, defined as $Qx = \{S_{4x}, S_{4x+1}, S_{4x+2}, S_{4x+3}\}$ and $Rx = \{S_x, S_{q+x}, S_{2q+x}, S_{3q+x}\}$, where $q = g/4$, $0 \leq x \leq q-1$. The SBox grouping simply maps SBoxes from Qx^i to Rx^{i+1}, where within this group the 16-bit mapping is defined as the group mapping described above. This is the adaptable component of the bit permutation, as one can see that the SBox grouping is well-defined as long as n is a multiple of 16.

5.3 Selection of the DEFAULT SBoxes

Here we describe the selection process of the LS SBox (used in DEFAULT-LAYER) and the non-LS SBox (used in DEFAULT-CORE) providing high resistance against linear attacks. A summary of various properties of our chosen SBoxes together with SBoxes from other lightweight ciphers (PRESENT, SKINNY-64 and GIFT) are shown in Table 3.

As for the size of the SBox, we decided to choose 4-bit. Although there are better (in terms of DFA security) 8-bit SBoxes (see Table 1), we chose the 4-bit SBoxes for the following main reasons: (1) to lower the cost (similar to GIFT [8]), (2) making the MILP modelling (described in Section 7) more efficient as generating the same for 8-bit SBoxes could be costly [36].

Table 3: Properties of the DEFAULT (LS, Non-LS), PRESENT, SKINNY-64 and GIFT 4-bit SBoxes. DBN is differential branch number, LBN is linear branch number, LS are the linear structures, DU is the differential uniformity, AD is the algebraic degree of the coordinate functions and NL is the non-linearity.

		DBN	LBN	LS	DU	AD		NL
						max	min	
DEFAULT LS	037ED4A9CF18B265	3	3	0, 6, 9, f	16	2	1	0
DEFAULT Non-LS	196F7C82AED043B5	2	2	0	8	3	2	4
PRESENT [12]	C56B90AD3EF84712	3	2	0	4	3	2	4
SKINNY-64 [9]	C6901A2B385D4E7F	2	2	0	4	3	2	4
GIFT [8]	1A4C6F392DB7508E	2	2	0	6	3	2	4

LS SBox. From the list of 302 affine equivalence (AE) classes of SBoxes by De Cannière [15], there are 10 AE classes with non-zero linear structures. Among these 10 AE classes, 8 of them (#293 — #300) have only one non-zero linear structure, AE class #301 has three non-zero linear structures and the last AE class #302 is fully linear (contains the identity permutation). To maximize the number of linear structures and yet to use a non-linear permutation, we chose the AE class #301 (the representative for this AE class in [15] is `1032456789ABCDEF`).

Within this class, we chose an SBox with the following criteria (HW denoting Hamming weight):

1. Both differential and linear branch number 3.
2. Zero diagonal in the DDT and LAT (except $(0, 0)$).
3. In the DDT, $\forall \delta_i \in \mathbb{F}_2^4 \setminus \{0\}$, if $(\delta_i, \delta_o) = 16$, then $HW(\delta_i) \geq 2$, $HW(\delta_o) \geq 2$.
4. In the LAT, $\forall \alpha_i \in \mathbb{F}_2^4 \setminus \{0\}$, if $(\alpha_i, \alpha_o) = 8$, then $HW(\alpha_i) + HW(\alpha_o) \geq 4$.

In other words, first we try to optimize the differential and linear diffusion with branch number 3. Next, we avoid enabling 1-round iterative differential or linear patterns (hence we look for empty diagonals). Then, for any probability 1 differential transition, we make sure that the input and output difference Hamming weight is at least 2 (we could not find an SBox for which such transitions necessarily happen with $HW(\delta_i) \geq 3$, or $HW(\delta_o) \geq 3$, or $HW(\delta_i)+HW(\delta_o) \geq 5$). Lastly, for any full linear transition, we select an SBox that will maximize the Hamming weight of the input and output values. The two last criteria are basically trying to maximize the number of active SBoxes before and after a probability 1 differential or a full linear transition. In total, we found 240 SBoxes candidates that satisfy our selection criteria and we ended up choosing SBox `037ED4A9CF18B265`.

Any of these 240 SBoxes, combined with our `DEFAULT-LAYER` bit permutation, ensures the following properties: for any 5-round differential characteristic,

(P1) there are at least 10 active SBoxes,

(P2) if there are exactly 10 active SBoxes, then each of these active SBoxes has differential probability 2^{-1} (which totals to 2^{-10}),

(P3) if there exists one active SBox with differential probability 1, then there are at least 12 other active SBoxes with differential probability 2^{-1} each (which totals to 2^{-12}).

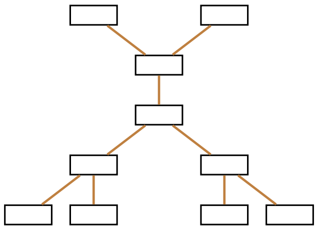
We give a general intuition on how the selection criteria facilitates these properties (we actually do not really need to prove these properties, since we will later be using automated tools to guarantee bounds on the differential characteristics probability). First, observe that all the 240 SBoxes will ensure that $\forall \delta, \Delta \in \mathbb{F}_2^4 \setminus \{0\}$,

(C1) if $\mathrm{DDT}_S[\delta, \Delta] > 0$, then $HW(\delta) + HW(\Delta) \geq 3$,

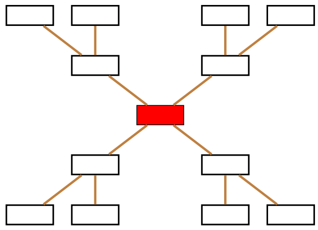
(C2) if $\mathrm{DDT}_S[\delta, \Delta] = 16$, then $HW(\delta) + HW(\Delta) \geq 4$,

(C3) if $\mathrm{DDT}_S[\delta, \Delta] = 16$, then $HW(\delta) \geq 2$ and $HW(\Delta) \geq 2$.

Then, from (C1) one can prove that there will be at least 10 active SBoxes over 5 rounds (P1) (in Figure 2(a)), which is basically Theorem 1 in [12]. By (C2) and the first case in the proof of Theorem 1 in [12], one can show that for such a 10-active SBoxes differential characteristic, none of these SBoxes (in Figure 2(a)) can have a differential probability 1 (P2). If there exists an SBox with differential probability 1, again by (C1) and (C2), there are at least 13 active SBoxes (see Figure 2(b)). Criterion (C3) enforces that only 1 of these 13 active SBoxes can potentially have differential probability 1 (P3).



(a) 10 active SBoxes with differential probability $< 2^{-10}$



(b) 13 active SBoxes with differential probability $< 2^{-12}$

Fig. 2: 5-round differential characteristics (solid lines are active bits, white boxes are active SBoxes and red box is SBox with differential probability 1)

From these properties, we can (conservatively) estimate that the probability of any differential characteristic drops by at least a factor of 2^2 for every additional round.

Non-LS SBox. For this SBox candidate, we focused on the linearity of the SBox as linear attacks will be the most difficult part to protect. Among the 33 AE classes with the lowest maximum linear bias 2^{-2}, the AE classes #32 (represented by `C0A23547691B8DEF`) and #33 (represented by `D0A23547691BC8EF`) have the least number of non-zero entries in the LAT. Statistically speaking, this gives us a higher chance of finding linear branch number 3 SBoxes. However, every 4×4 SBox with linear branch number 3 has at least one non-trivial linear structure (belonging to the AE classes #294, #297, #298, #300, #301, #302 of [15]). Hence, we tried several of those SBoxes and obtained the corresponding linear bias bounds using the automated technique described in Section 7. However, the bounds we obtained were not good enough. Thus, our next strategy was to select an SBox with the following linear properties:

1. $\sharp\{((\alpha_i, \alpha_o)) \mid HW(\alpha_i) = HW(\alpha_o) = 1, (\alpha_i, \alpha_o) \neq 0\} = 1$.
2. Zero diagonal in the LAT (except $(0, 0)$).
3. $\sharp\{((\alpha_i, \alpha_o)) \mid HW(\alpha_i) + HW(\alpha_o) = 3, (\alpha_i, \alpha_o) = \pm 4\} = 13$.
4. $\sharp\{((\alpha_i, \alpha_o)) \mid HW(\alpha_i) + HW(\alpha_o) = 3, (\alpha_i, \alpha_o) = \pm 2\} = 6$.

In other words, first we limit the number of Hamming weight $1 \to 1$ transitions to 1[11]. Next, we avoid having a 1-round iterative linear pattern. Lastly, we

[11] In [31], the authors show that, under their BOGI+ paradigm, when there are at least 9 consecutive rounds, having only 1 Hamming weight $1 \to 1$ transition is a sufficient condition to achieve a theoretic bound of at least 2 active SBoxes per round.

minimize the number of possible Hamming weight $1 \to 2$ and $2 \to 1$ transitions. This is to encourage faster and wider propagation of the linear trail. We finally choose the SBox `196F7C82AED043B5` from the AE class #32.

We note that other considerations could be incorporated in addition to the ones mentioned in this section, such as side-channel attacks resilient criteria [23], but we believe this falls out of the scope of our research that tries to focus on natural immunity to DFA.

5.4 Unbiased Linear Structures

We need an extra security criterion: each bit of the linear structures of S as well as S^{-1} must be unbiased. This is to avoid certain undesirable property of the linear layer. If we assume that the linear structures for S are $\{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}\}$, the two MSBs are always 0. One such SBox is `1032456789ABCDEF` (the representative for class #301 in [15]). It has the property that if the first two bits of its input are known uniquely, then the first two bits of its output are also known uniquely. The attacker may be able to leverage this property by attacking the penultimate round of the cipher/protection layer, with attacking the last round. This issue does not arise when each bit of the linear structures is unbiased (in which case the attacker is not able to find any bit uniquely). In our chosen LS SBox, the linear structures being $\{\mathtt{0}, \mathtt{6}, \mathtt{9}, \mathtt{f}\}$, and that of the inverse SBox being $\{\mathtt{0}, \mathtt{5}, \mathtt{a}, \mathtt{f}\}$, this criterion is indeed satisfied.

6 Security Analysis

Conducting security analysis on `DEFAULT` is quite different from conducting security analysis on block ciphers, despite having similar structure. This is because `DEFAULT-LAYER` is built on top of an existing (and presumably secure against classical attacks) cipher and only assists in providing the desired security against DFA, while `DEFAULT-CORE` is used in conjunction with two instances of `DEFAULT-LAYER`. Although classical attacks do not pose any threat against `DEFAULT-LAYER`, some cryptanalytic techniques could still be applied to `DEFAULT` through DFA. For instance, suppose an attacker injects faults to the output of the main cipher, this difference will only propagate through the `DEFAULT-LAYER` and not the entire cipher, creating some form of differential attack on the `DEFAULT-LAYER` itself. Thus, we need to ensure that `DEFAULT-LAYER` is not vulnerable to classical attacks that could bypass the main cipher using DFA and target `DEFAULT-LAYER` directly. The desired security for the classical attacks are summarized in Table 4 and security evaluation against such attacks are done subsequently in Section 6.2. Detailed discussion on the classical attacks are omitted here for brevity, but interested readers may find it for example in [8, Section 4]. It may be noted that more precise differential and linear bounds are presented in Section 7. The security against DFA and side-channel attacks are evaluated subsequently (Section 6.1 and Section 6.3, respectively).

As `DEFAULT` comprises of `DEFAULT-CORE` and (two layers of) `DEFAULT-LAYER`, we specify which component we are analyzing and for which cryptanalysis technique. The analysis is summarized in Table 5.

Table 4: Security requirement of DEFAULT against classical attacks

	DEFAULT-LAYER	DEFAULT-CORE
Differential, Algebraic	2^{64} Search Complexity	–
Integral, Impossible Diff.	–	No Distinguisher
Linear	2^{32} Search Complexity	2^{64} Search Complexity
Invariant Subspace	–	2^{128} Search Complexity

Table 5: Security analysis of DEFAULT

	DEFAULT-LAYER (28-rounds)	DEFAULT-CORE (24-rounds)	DEFAULT (80-rounds)	Ref.
Differential Fault Attacks (64-bit Security)				
On DEFAULT-LAYER	2^{64}	Bypassed	2^{64}	Sec. 6.1
On DEFAULT-CORE	$\geq 2^{64}$	Negligible	$> 2^{64}$	
Double Fault	Not applicable			
Classical Cryptanalysis (128-bit Security)				
Differential	$\geq 2^{64}$	$> 2^{24}$ (Trivial)	$> 2^{128}$	Sec. 6.2
Linear	$> 2^{40}$	$> 2^{128}$	$> 2^{128}$	
Impossible Diff.	Main cipher	Longest: 6 rnd	Not vulnerable	
Invariant Subspace	Main cipher	Not vulnerable	Not vulnerable	
Algebraic	Main cipher	Not vulnerable	Not vulnerable	

6.1 Differential Fault Attacks

First, we look at DFA on DEFAULT-LAYER, when it is used as a protection layer for other block ciphers. Next, we look at DFA on DEFAULT-CORE or other block ciphers with DEFAULT-LAYER as protection layer.

DFA on DEFAULT-LAYER. Our chosen SBox has 3 non-trivial linear structures: $\mathtt{6}, \mathtt{9}, \mathtt{f}$. Hence, for any input $\alpha \in \{\mathtt{0}, \ldots, \mathtt{f}\}$, the attacker cannot uniquely identify which among $\{\alpha, \alpha \oplus \mathtt{6}, \alpha \oplus \mathtt{9}, \alpha \oplus \mathtt{f}\}$ is the actual input to the SBox. In other words, the attacker will be able to identify one partition of the input: $\{\{\mathtt{0}, \mathtt{6}, \mathtt{9}, \mathtt{f}\}$, $\{\mathtt{1}, \mathtt{7}, \mathtt{8}, \mathtt{e}\}$, $\{\mathtt{2}, \mathtt{4}, \mathtt{b}, \mathtt{d}\}$, $\{\mathtt{3}, \mathtt{5}, \mathtt{a}, \mathtt{c}\}\}$, but will not be able to identify which particular input is correct. Similarly for the output of the SBox, due to the linear structures, the attacker will only able to identify the partition to be one of these $\{\{\mathtt{0}, \mathtt{5}, \mathtt{a}, \mathtt{f}\}, \{\mathtt{1}, \mathtt{4}, \mathtt{b}, \mathtt{e}\}, \{\mathtt{2}, \mathtt{7}, \mathtt{8}, \mathtt{d}\}, \{\mathtt{3}, \mathtt{6}, \mathtt{9}, \mathtt{c}\}\}$ and not a particular output.

In the last round attack of DEFAULT-LAYER, the attacker has to inject faults and analyze each of the 32 SBoxes independently. That means, for each SBox he has to do a brute-force search of 4, leading to a total search complexity of $4^{32} = 2^{64}$.

DFA on DEFAULT-CORE or other block ciphers with DEFAULT-LAYER. Alternatively, the adversary could still try to launch DFA on the main cipher by injecting fault(s) to the last round of it and hope that it will propagate nicely through DEFAULT-LAYER. If so, it boils down to whether the adversary can distinguish the output difference from the main cipher with less than 2^{64} effort, otherwise it is better off attacking DEFAULT-LAYER directly (2^{64}). Using MILP, we found that the maximum differential probability of DEFAULT-LAYER is upper bounded by 2^{-64} (details in Section 7.2). Thus, the attack complexity is too high and this alternative strategy is not worthwhile.

Information-combining DFA on DEFAULT-LAYER. An attacker could apply DFA on multiple rounds and hope to combine these learnt information to further reduce the number of key candidates. For instance, targeting the last two rounds of DEFAULT, or the first and last round of DEFAULT through DFA on both the encryption and decryption processes. There are two possible strategies: 1) combine the information of the input (resp. output) solutions of SBoxes from different rounds, or 2) combine the input and output solutions of the SBoxes.

For the first strategy, since the same layer-key is XORed to every round, one would either get the same 4-bit key candidates (for same nibble positions) or completely independent key bits which does not help in reducing the key candidates. For the second strategy, to combine the 4-bit input and output solutions of the SBoxes, either side has to propagate through the bit permutation. Due to the Group Mapping, 4 SBoxes from each side are involved and the analysis is extended 16-bit. After combining the information together (details omitted for brevity sake), one could only reduce the key candidates from 2^{16} to 2^{12}. This is worse than targeting individual SBoxes which reduces the space from 2^4 to 2^2.

In summary, DEFAULT-LAYER is resistant against information-combining DFA.

6.2 Classical Cryptanalysis

In the following, we apply classical cryptanalysis techniques on DEFAULT. Recall that DEFAULT has a sandwich structure with two DEFAULT-LAYER layers and a DEFAULT-CORE layer in the middle. For most of the cryptanalysis considered, it will be sufficient to show that DEFAULT-CORE is resistant against the attack.

Differential Cryptanalysis. Using MILP, we found that the maximum differential probability of DEFAULT-LAYER is upper bounded by 2^{-64} (details in Section 7.2). Since there are two layers of DEFAULT-LAYER, we already show that there is no meaningful differential characteristic tracing across two layers of DEFAULT with differential probability more than 2^{-128}. In addition, DEFAULT-CORE has 24 rounds and any differential characteristic will involve at least 1 active SBox per round. Thus, this trivially adds an additional factor of 2^{-24} to any differential characteristic. In summary, DEFAULT is not susceptible to differential cryptanalysis.

Linear Cryptanalysis. Using MILP, we found that the absolute linear bias of 11-round DEFAULT-CORE is upper bounded by 2^{-33} (details in Section 7.2). Thus, with a simple concatenation of two 11-round linear characteristics, we can show that there is no meaningful 22-round linear characteristic in DEFAULT-CORE. In addition, there are two layers of DEFAULT-LAYER, which will only make the linear cryptanalysis even harder to realise (even though linear structures are present in the SBox). In summary, DEFAULT is not susceptible to linear cryptanalysis.

Impossible Differential Attacks. We considered the possible effect of impossible differential attacks against `DEFAULT-CORE`. As proposed in [32], we generated MILP instances (Section 7) for all $\binom{128}{1} \times \binom{128}{1} = 16384$ differentials with both the input and output differences of Hamming weight 1 on `DEFAULT-CORE` and check if any of these instances were infeasible, which implies impossible differential. For the 7^{th} round, we observe all instances are feasible (i.e., no impossible differential exists). Therefore, following the philosophy of [32], we believe the full-round `DEFAULT-CORE` is secure against impossible differential attacks.

Invariant Subspace Attacks. In order to simplify the analysis of invariant subspace attacks, we assume that any (affine) subspaces are preserved over the entire `DEFAULT-LAYER`, the `PermBits` and `AddRoundConstants` step. Thus, we focus on subspace transition over the `SubCells` step in `DEFAULT-CORE`, namely the non-LS SBoxes layer.

There is no dimension 3 (affine) subspace transition, and among the dimension 2 transitions most of them can only propagate up to 3 rounds, except one: $\mathtt{5} \oplus \{\mathtt{0}, \mathtt{2}, \mathtt{c}, \mathtt{e}\} \rightarrow \mathtt{0} \oplus \{\mathtt{0}, \mathtt{2}, \mathtt{c}, \mathtt{e}\}$. Notice that this affine subspace will be preserved over the `AddRoundKey` step if each nibble of the round key belongs to $\{\mathtt{5}, \mathtt{7}, \mathtt{9}, \mathtt{b}\}$.

Moreover, observe that every value in $\{\mathtt{5}, \mathtt{7}, \mathtt{9}, \mathtt{b}\}$ has the least significant bit set to 1. After the 1-bit rotation within the 16-bit word, the most significant bit is set to 1. For these round key nibbles to still preserve the subspace transition, it has to be either `9` or `b`. However, after 7 key schedule updates, some of these nibbles have another 1-bit right rotation, which will break the subspace structure. Thus, we believe that no (affine) subspace can be preserved for more than 8 rounds and `DEFAULT` is not vulnerable to invariant subspace attacks.

Algebraic Attacks. In order to evaluate the security of `DEFAULT-CORE` against algebraic attacks, we checked its algebraic properties using Sage[12]. We are able to represent `DEFAULT-CORE` as Boolean expressions up to 4-rounds. We have observed that the minimum number of monomials is 11101, at least 97 variables (out of 128) are involved and the minimum algebraic degree is 8. Furthermore, computing bounds on the maximum algebraic degree for different number of rounds according to the degree estimate given in [13], we can hope to reach maximum degree 127 after 8 rounds.

Integral Attacks. Suppose an attacker repeats the encryption multiple times and injects all possible differential fault values to a specific word in the output of the main cipher. This is similar to collecting a set of inputs (more precisely the output from the main cipher) with specific structure to launch an integral attack. Such model is reported in [29] and [30, Chapter 6.3].

This model is a special case of DFA where all possible faults are considered. Since the attacker does not get any extra information by using all possible faults, `DEFAULT-LAYER` (and hence `DEFAULT`) is resistant against it.

As for `DEFAULT-CORE`, we could reuse some of the security analysis of `GIFT-128` for our design. In particular, the designers of `GIFT` evaluated the longest integral

[12] `http://www.sagemath.org/`

distinguisher for GIFT-128 using the (bit-based) division property [37] to be 11 rounds, and concluded that GIFT-128 is secure against integral attacks. Since DEFAULT-CORE has 28 rounds, we believe that DEFAULT-CORE is secure against integral attacks.

Using the SOLVATORE tool [19], we could find a distinguisher for DEFAULT-LAYER till 12 rounds. Beyond this, no solution is returned in a reasonable time.

6.3 Protection Against Side-Channel Attacks

In essence, DEFAULT-LAYER/DEFAULT is simply a bit permutation based SPN block cipher and, as such, usual side-channels attacks might apply on it. Usual countermeasures such as masking can of course be applied on DEFAULT.

We point out that protecting DEFAULT against side-channels attacks should not make DFA easier. An additional feature of the DEFAULT-LAYER SBox is that it has lower number of AND operations compared to the usual SBoxes used in other cipher designs, hence making it easier to mask [25]. One might argue that the large number of rounds of DEFAULT or DEFAULT-LAYER would be problematic, but implementation trade-offs would partially avoid this issue (implementing 2 or 4 rounds per clock cycle would greatly improve the throughput while moderately increase the area).

6.4 Comparison With CRAFT, FRIET and Duplicated Computation

As stated earlier, CRAFT and duplication are the two most relevant countermeasures when comparing with DEFAULT. Under a single fault adversary, duplication and DEFAULT are all secure against DFA. CRAFT in itself does not protect against DFA but is designed with a consideration to make it cost effective when integrating error detection codes. CRAFT only protects against faults that are detectable by the deployed error detection code and remains vulnerable to faults outside the detection capability. For an error detection codes with minimum distance d (*i.e.* minimum distance between distinct codewords), CRAFT can detect faults altering up to $t(= d-1)$ cells[13] at once (within one cycle). Note that for low cost equipment where injected faults are often random, the probability of getting a fault which is beyond the detection limit of error detection code is non-negligible. With precise fault injection equipment, an adversary could inject specific difference large enough ($\geq t$ cells) to change the code to another valid code and fool the error detection mechanism trivially. On the contrary, DEFAULT is not bounded by any such t.

FRIET adopts a parity check code to detect a *single-limb*[14] fault in the computation. Similar to CRAFT, for faults that alter more than one limb are beyond the detection limit. Again, DEFAULT is not bounded by any such limb.

Regarding duplicate faults, CRAFT claims no security. Duplicated computation was demonstrated to be broken by injecting two identical faults in the redundant execution using state of the art fault injection equipment [33]. DEFAULT is not vulnerable to DFA under duplicate faults as it does not rely on redundancy.

[13] The "cell" is adopted from the CRAFT paper [10] referring to the word size.
[14] The "limb" refers to an array of bits within the internal state of FRIET

6.5 Other Fault Attacks

Although we do not claim security against attacks that uses analysis method 2, for completeness we discuss the security of our design against some of such attacks.

Fault Altering Control/Algorithm Flow. Since our solution is at algorithm level and does not rely on any engineering solutions, it is natural that our security claim holds under the assumption of the correctness of our algorithm. Therefore, we do not claim security against faults that alter the execution sequence of the algorithm. An accomplished attacker could hypothetically skip the execution of `DEFAULT-LAYER` completely with a control flow fault and can target the main cipher with standard DFA.

Hypothetical Multiple Precision Fault Attacks. Consider a *multiple precision fault attack* where the adversary injects a fault to introduce a specific difference just before an SBox and another difference right after the same SBox in an attempt to precisely cancel the difference. When the cancellation is successful, it will result in the same output as a fault-free execution, and the adversary can obtain the possible solutions for that SBox. While this is not effective against our LS SBoxes, it could still target the main cipher which typically does not have any LS. Feasibility of such precise multiple faults have never been demonstrated. In addition, this attack falls under analysis method 2 which is outside of our fault model.

Precise Bit Flipping Attacks. A single bit flip on a specific bit, though much harder to achieve, has been reported in practice by lasers [3]. Despite its precision, bit precision DFA (equivalent to injecting a Hamming weight 1 difference) will still be ineffective against our design. As described in Section 6.1, any input α will still lead to multiple solutions thanks to our LS SBox.

Assume that the adversary can target the logic gate component of the SBox, there could be a statistical attack, but again, we do not make claims against attacks that fall under analysis method 2.

Other non-DFA models. The *Safe Error Attack* (SEA) [22, 38, 39] model has been proposed which utilizes the cases where the faulty and non-faulty outputs are the same. Among the SEA models, one particular model is known as *Ineffective Fault Attack* (IFA) [14]. Another type of fault attack uses statistical information on the output distribution as it has become biased because of fault injection [27,40]. Such analysis often are based upon hostile fault models like stuck-at, permanent or persistent faults which assume a stronger attacker, specially stuck-at faults which are widely used in the fault analysis literature [17, 27]. Stuck-at faults in electronic devices are generally related to defects in devices either at manufacturing or due to high-energy radiation in space electronics. Injecting stuck-at fault intentionally for malicious purpose requires expensive equipment like precise lasers, ion beams, etc. and thus considered under strong adversary capability . In comparison, bit flips or random faults are relatively easier to realise with simple fault injection equipment. A hybrid model – *Statistical Ineffective Fault Attack* (SIFA) [17] is proposed. It relies on both ineffective fault and statistical information of the computation. All these attacks exploit information leakages from statistical biases under analysis method 2, which is beyond our focus.

7 Automated Bounds for Differential and Linear Attacks

In [26], the authors present a method to find optimal differential and linear characteristics based on *Mixed Integer Linear Programming* (MILP), which is then tuned to work with bit permutation based block ciphers in [36].

Indeed, our special SBox with linear structures has probability 1 differential transitions (resp., $\pm\frac{1}{2}$ linear bias). For the differential case, the above mentioned approach will always yield an MEDP bound of $\epsilon_d = 1$ (1 is raised to the power of an integer), which naturally signifies the smallest possible protection against differential attacks (the attack succeeds with only one chosen input difference or two chosen inputs). In case of linear cryptanalysis, it can be shown that the overall bias ϵ_l, considering only $\pm\frac{1}{2}$ biases (and assuming mutual independence of the biases), is $\frac{1}{2}$. This is obtained by substituting $\epsilon_i = \frac{1}{2}\ \forall i$ in [35, Lemma 3.1]. Similar to the differential case, this also leads to the smallest protection against linear attack (the attack succeeds with roughly $1/\epsilon_l^2 = 4$ known inputs). Naturally, we need to devise a way to count precisely the number of probability $\frac{1}{2}$ differential transitions and $\pm\frac{1}{4}$ linear biases.

To overcome this problem, we devise a new strategy which is inspired from the concept of *indicator constraint* used in linear programming (also known as the *big M* method), where a large constant M is chosen.

Due to space constraints, the details of our strategy and description of the MILP modeling can be found in the Supplementary Material (Section D).

7.1 Optimizations

Using the idea described in previous section, we construct the MILP problems and attempt to solve them using the Gurobi[15] solver. Being inspired from [24], we use redundancy in the MILP constraints. Using redundant constraints together with the usual constraints does not change the problem description, but could make the execution faster. As for the choice of the heuristics, we use the idea of *Convex Hull* (CH) [36].

For the differential case, we use the complete set of the CH inequalities, while for the linear case we use the greedy algorithm to select a subset of the complete set of the CH inequalities. The details on generation of the CH inequalities and the greedy algorithm can be found in [36]. We observe that using the heuristics the solution time can be improved by almost a factor of 10 compared to the respective cases where no heuristic was used. For more details on the heuristics, refer to [4].

7.2 Results

For the LS SBox (used in `DEFAULT-LAYER`), the bounds obtained from the corresponding MILP programs are: 2^{-4} at the 5^{th} round for linear, and 2^{-20} at the 7^{th} round for differential. This translates to around 2^8 computations for 5 rounds against classical linear attacks and around 2^{20} computations against differential

[15] `https://www.gurobi.com/`

attacks. Hence, we believe 28 rounds of `DEFAULT-LAYER` is enough to provide a security level of 2^{64} computations against classical differential attacks and of 2^{32} computations against classical linear attacks.

As explained in Section 6.2, we only consider the security against the classical linear attack against `DEFAULT-CORE`. For the non-LS SBox (used in `DEFAULT-CORE`) `196F7C82AED043B5`, the bound obtained from the MILP program for the linear case is 33.00 at the 11^{th} round. Hence, the linear cryptanalysis security at 11 rounds of `DEFAULT-CORE` is around 2^{66} computations. Hence, we conclude `DEFAULT` ensures the required DFA security (of 2^{64} computations) and also the required classical security (of 2^{128} computations).

Table 6: Differential and linear bounds (in $-\log_2$ notation) for LS and non-LS SBoxes

(a) LS SBox: `037ED4A9CF18B265`

Rounds	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Diff.	0	0	2	6	10	15	20
Linear	0	0	0	1	4	-	-

(b) Non-LS SBox: `196F7C82AED043B5`

Rounds	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Linear	1	2	4	6	8	12	16	20	25	30	33

More results regarding this can be found in Table 6 (Table 6(a) for differential and linear bounds for the LS SBox `037ED4A9CF18B265` and Table 6(b) for linear bounds for the non-LS SBox `196F7C82AED043B5`), as obtained from the MILP instances. Those results are obtained from a workstation with 16× Intel Xeon E7-8880 physical cores (shared among multiple users), running Gurobi 8.1 on 64-bit Ubuntu 18.04. Due to the time taken by the solver, it would be difficult to compute the bounds beyond the ones given in Table 6, at least with the current modelling (and with our computing resource).

8 Performance

In this part we state benchmarks for hardware and software implementations of `DEFAULT`. Comparison is done with `GIFT-128` and a duplication-protected `GIFT-128` which runs the same computation twice (in space or time) and compares the output. The output is released only if both computations produce same ciphertext, otherwise it is suppressed. This is the so-called *detective countermeasure* [7]. As a side note, it can be mentioned that the current academic researches have drifted away from the simple detective countermeasure towards more sophisticated error detection code-based or infection-based countermeasures, which would incur higher overheads. If such a sophisticated countermeasure is taken into account, `DEFAULT` provides much better performance.

8.1 Hardware Benchmark

The area and throughput for `DEFAULT`, `GIFT-128` and `AES` are given in Table 7. We also provide the same for `GIFT-128` and `AES` when protected with spatial or temporal duplication, or with `DEFAULT-LAYER`. The code is written in Verilog, and synthesized on Synopsys Design Compiler J-2019 on the TSMC 65nm standard cell library. The area is given in gate equivalents. The throughput is computed for 2 GHz clock frequency. Since `DEFAULT` inherits the key schedule of the protected cipher, we assume the round keys are precomputed for all implementations. For `GIFT-128` with `DEFAULT-LAYER`, we implemented two versions. The first (v1) is a simple combination of `DEFAULT-LAYER` with main cipher, while the second one (v2) takes advantage of the structural similarities between `GIFT-128` and `DEFAULT-LAYER`. For `AES`, we noticed that the area required to implement `DEFAULT-LAYER` is negligible compared to the size of the `AES` circuit. Besides, the `AES` circuit is the bottleneck for clock frequency. Hence, we experimented with 3 different architectures for `DEFAULT-LAYER` i.e. one round ($\times 1$), two round ($\times 2$) and four rounds ($\times 4$) unrolled per clock cycle. In order to put our results into perspective, we implemented two versions of the simple duplication countermeasure for `AES` and `GIFT-128`. The first version is temporal duplication, where the cipher is implemented once and called twice, then the outputs are compared. The second version is spatial duplication, where two instances of cipher are computed in parallel followed by final comparison.

Table 7: ASIC Synthesis Results on the TSMC 65nm library.

Design	**Area** (GE)	**Cycles**	**Throughput** (Mbps)
`DEFAULT-LAYER`	2011	28	9143
`DEFAULT`	3377	80	3200
`GIFT-128` + `DEFAULT-LAYER` v1	3301	96	2667
`GIFT-128` + `DEFAULT-LAYER` v2	3201	96	2667
`GIFT-128`	3067	40	6400
`GIFT-128` temporal duplication	4104	81	3160
`GIFT-128` spatial duplication	6569	41	6244
`AES` + `DEFAULT-LAYER` ($\times 1$)	13430	67	3821
`AES` + `DEFAULT-LAYER` ($\times 2$)	14651	39	6564
`AES` + `DEFAULT-LAYER` ($\times 4$)	17062	25	10240
`AES`	13196	11	23273
`AES` temporal duplication	14232	23	11130
`AES` spatial duplication	26994	12	21333

Our results show that for `GIFT-128`, the area needed to add the `DEFAULT-LAYER` is almost negligible, while the throughput drops by a factor of $2.4\times$. The overall overhead, estimated by the throughput overhead multiplied by the area overhead, is $2.5\times$. The area of our design is significantly smaller than both types of duplication. Spatial duplication has smaller overall overhead ($2.2\times$) at the cost of doubling the area, while temporal duplication has a worse overhead of $2.7\times$.

For `AES`, the cost for adding `DEFAULT-LAYER` ($\times 1$) to `AES` is also negligible, the `DEFAULT-LAYER` ($\times 4$) architecture leads to lowest overhead among the three (particularly $3\times$). Comparing duplication, the overall overhead for temporal and spatial duplication of `AES` is almost the same ($2.25\times$). In terms of low area consideration, our implementations are strictly smaller than duplication. Besides, the drawbacks of simple duplications were discussed in details in Section 6.4, which we believe justifies the cost of our countermeasure.

We have also synthesized our implementations for the Xilinx Kintex 7 FPGA. We fixed the clock frequency to 200 MHz. Due to the nature of FPGA look-up tables (LUTs), they are sometimes under-utilized. This makes it possible to add extra functionality or extra flip-flops to the design for almost no cost. The results are given in Table 8. Our results show that the `DEFAULT-LAYER` can be added to `GIFT-128` for no extra LUTs or flip-flops. The throughput drops by a factor of 2.4×. Both types of duplication lead to drop in throughput and increase in both LUTs and flip-flops. The overall overheads are 4.05× and 2.56× for temporal and spatial duplication, respectively. While the number of flip-flops is doubled in both cases, we only consider it in the overhead in the case of temporal duplication, as they are already merged with the LUTs in the case of spatial duplication.

Table 8: FPGA Synthesis Results on Kintex 7.

Design	**Cycles**	**LUT**	**FF**	**Throughput** (Mbps)
`DEFAULT-LAYER`	28	256	128	914.3
`DEFAULT`	80	256	128	320.0
`GIFT-128` + `DEFAULT-LAYER` v1	96	358	128	266.7
`GIFT-128` + `DEFAULT-LAYER` v2	96	256	128	266.7
`GIFT-128`	40	256	128	640.0
`GIFT-128` temporal duplication	81	384	256	316.0
`GIFT-128` spatial duplication	41	640	256	624.4
`AES` + `DEFAULT-LAYER` (×1)	67	918	128	382.1
`AES` + `DEFAULT-LAYER` (×2)	39	964	128	656.4
`AES` + `DEFAULT-LAYER` (×4)	25	1204	128	1024.0
`AES`	11	528	128	2327.3
`AES` temporal duplication	23	656	256	1113.0
`AES` spatial duplication	12	1184	256	2133.3

In the case of duplication for `AES`, the ×1, ×2 and ×4 unrolled architectures of `DEFAULT-LAYER` have an overhead of 10.58×, 6.47× and 5.18× respectively. Spatial and temporal duplication have overheads of 2.44× and 2.59×. While duplication is about twice as efficient as our solution when it comes to `AES`, this is only specific to `AES` as its base line cost is relatively reduced on FPGAs, taking advantage of the large LUTs available. Moreover, the security features of `DEFAULT` compared to duplication still makes it interesting for `AES` on FPGAs.

8.2 Software Benchmark

The software benchmarks for `GIFT-128`, duplicated `GIFT-128` (in time) and `DEFAULT` are given in Table 9. The relative overheads compared to `GIFT-128` are shown within parenthesis. The clock cycles were measured by utilizing `time()` function from `time.h` library in C, by averaging over multiple executions. Program was running on a single core. Compiler optimizations were disabled to produce a consistent result. Note that the main purpose of this benchmark is to show the relative performance compared to `GIFT` in the same setting. It can be seen that the code size for `DEFAULT` is slightly more compared to duplicated `GIFT-128`, but at the same time `DEFAULT` is faster. We would also like to note that a new efficient software representation of `GIFT` was published recently [1], called the fixslicing technique, drastically reducing the cycles needed for encryption on ARM

Cortex-M family of microcontrollers. The fixsliced implementation of `DEFAULT` would have very similar per-round performances as `GIFT-128`, as the permutation is the same (which is what the fixslicing technique is trying to optimize), while the Sboxes have similar cost. Overall, we expect the overheads to be similar as it scales accordingly to the number of rounds. Generally, this scaling would apply to other optimizations as well.

Table 9: Software benchmarking for `DEFAULT` and `GIFT-128` with/without duplication

		Intel Xeon Silver 4215	Arm Cortex A-53
Speed (Cycles/Bytes)	`GIFT-128`	9.7 (1.000×)	61.3 (1.000×)
	`GIFT-128` Duplicated	21.9 (2.258×)	124.4 (2.029×)
	`DEFAULT`	19.2 (1.979×)	121.9 (1.989×)
Code Size (Bytes)	`GIFT-128`	6624 (1.000×)	5593 (1.000×)
	`GIFT-128` Duplicated	6859 (1.035×)	5818 (1.040×)
	`DEFAULT`	8024 (1.211×)	7085 (1.267×)

9 Conclusion and Future Works

In this paper, we presented the first theoretical study on SBoxes with respect to their properties against differential fault attacks (DFA). We observe that DFA works as a simplified model of differential attacks (DA), yet the properties of an SBox which makes DFA harder, will make DA easier, and vice-versa. Our findings enabled us to propose the first cipher-level countermeasure against DFA. Our construction does not incur too much overhead and is competitive with state-of-the-art in terms of performances, while protecting against a larger spectrum of faults. The core idea is to use a special SBox with linear structures, so that when trying all possible fault values, the attacker is not able to narrow down the search space below square root bound. This work opens up a new paradigm of symmetric-key cipher design, by studying SBoxes with LS, which has not been explored much yet.

Below we summarize the advantages and limitations of our proposal.

+ **First cipher-level protection.** This solves the concern raised against existing DFA countermeasures (Section 2.2). In particular, we remove the DFA protection from the hand of the cipher implementer to the cipher designer.
+ **Scalable to (almost) all symmetric-key primitives as an ad-hoc layer.** Using `DEFAULT-LAYER`, the basic concept we propose can be scaled to ensure a non-trivial DFA security on any symmetric-key primitive. We give a proof of concept for 128-bit state size, but it can be easily adapted to handle any state size that is multiple of 16 bits (by adjusting the number of rounds).
+ **Possibility to get a non-trivial DFA security.** The particular instantiation we propose offers up to $2^{n/2}$ DFA security where $n \geq 128$ is the state size of a block cipher (without jeopardizing its classical security). However, this is not a maximum limit as can be seen from Table 1. Note that, attack complexity of $2^{n/2}$ can be considered impractical for fault attacks.
+ **Protected against duplicate faults.** `DEFAULT` is not vulnerable to duplicate faults, unlike duplication based countermeasure. This remains true

regardless of the number of faults, unlike some error detection based protection where faults are not detected beyond a certain coverage.

+ **Extension to any FA that uses differential analysis method.** The use of LS Sboxes increases the number of solutions for any given differential, which makes any attack under analysis method 1 harder.
+ **No need for external randomness/ protected device.** The commonly referred infective countermeasure [7] uses an external source of randomness. For the protocol level countermeasures, such as [6], a part of the device is assumed to be off limit to the attacker due some device level protection. In our case, there is neither a need for an external source of entropy nor a specially protected device.
− **Not full DFA security.** It is technically possible to achieve almost full DFA security (such as 2^{112} for a 128-bit state, see Table 1). However, it does not seem possible to achieve a full state-size DFA security by this methodology.

We believe our work opens up a new research direction for fault-resilient ciphers and there are a few potential open problems that would be interesting to explore in future studies. To begin with, one can look for a self-inverse SBox that fits our criteria to reduce the hardware cost when both the layer and its inverse are implemented in the same circuit. As the LS SBox has fewer AND operations, future ciphers could be designed while leveraging this. Finally, a solution that would combine fault protection with side-channel resistance would be extremely valuable.

References

1. Adomnicai, A., Najm, Z., Peyrin, T.: Fixslicing: A new gift representation. IACR Cryptology ePrint Archive **2020** (2020) 412 26
2. Aghaie, A., Moradi, A., Rasoolzadeh, S., Shahmirzadi, A.R., Schellenberg, F., Schneider, T.: Impeccable circuits. Cryptology ePrint Archive, Report 2018/203 (2018) 5
3. Agoyan, M., Dutertre, J.M., Mirbaha, A.P., Naccache, D., Ribotta, A.L., Tria, A.: How to flip a bit? In: 2010 IEEE 16th International On-Line Testing Symposium, IEEE (2010) 235–239 22
4. Baksi, A.: New insights on differential and linear bounds using mixed integer linear programming (full version). Cryptology ePrint Archive, Report 2020/1414 (2020) `https://eprint.iacr.org/2020/1414`. 23
5. Baksi, A., Bhasin, S., Breier, J., Jap, D., Saha, D.: Fault attacks in symmetric key cryptosystems. Cryptology ePrint Archive, Report 2020/1267 (2020) 4
6. Baksi, A., Bhasin, S., Breier, J., Khairallah, M., Peyrin, T.: Protecting block ciphers against differential fault attacks without re-keying (extended version). Cryptology ePrint Archive, Report 2018/085 (2018) 5, 11, 28
7. Baksi, A., Saha, D., Sarkar, S.: To infect or not to infect: a critical analysis of infective countermeasures in fault attacks. J. Cryptogr. Eng. **10**(4) (2020) 355–374 4, 5, 24, 28
8. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: CHES 2017. (2017) 321–345 2, 9, 12, 14, 17

9. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO 2016. (2016) 123–153 9, 14
10. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: Craft: Lightweight tweakable block cipher with efficient protection against dfa attacks. IACR Trans. Symmetric Cryptol. **2019**(1) (Mar. 2019) 5–45 4, 5, 21
11. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In Kaliski, BurtonS., J., ed.: CRYPTO '97. Volume 1294 of LNCS. Springer (1997) 513–525 2
12. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: CHES. Volume 4727., Springer (2007) 450–466 9, 14, 15, 16
13. Boura, C., Canteaut, A., Cannière, C.D.: Higher-order differential properties of keccak and *Luffa*. In Joux, A., ed.: FSE 2011. Volume 6733 of LNCS., Springer (2011) 252–269 20
14. Clavier, C.: Secret external encodings do not prevent transient fault analysis. In: CHES 2007. (2007) 181–194 22
15. De Cannière, C.: Analysis and Design of Symmetric Encryption Algorithms. Katholieke Universiteit Leuven, Belgium (2007) PhD Thesis. 6, 15, 16, 17
16. Diffie, W., (translators), G.L.: Sms4 encryption algorithm for wireless networks. Cryptology ePrint Archive, Report 2008/329 (2008) 13
17. Dobraunig, C., Eichlseder, M., Korak, T., Mangard, S., Mendel, F., Primas, R.: SIFA: exploiting ineffective fault inductions on symmetric cryptography. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(3) (2018) 547–572 22
18. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. CAESAR Final Portfolio (2014) `https://ascon.iaik.tugraz.at/`. 13
19. Eskandari, Z., Kidmose, A.B., Kölbl, S., Tiessen, T.: Finding integral distinguishers with ease. Cryptology ePrint Archive, Report 2018/688 (2018) 21
20. Guilley, S., Sauvage, L., Danger, J., Selmane, N.: Fault injection resilience. In: FDTC 2010. (2010) 51–65 5
21. He, W., Breier, J., Bhasin, S., Miura, N., Nagata, M.: Ring oscillator under laser: Potential of pll-based countermeasure against laser fault injection. In: FDTC 2016, IEEE (2016) 102–113 5
22. Joye, M., Quisquater, J., Yen, S., Yung, M.: Observability analysis - detecting when improved cryptosystems fail. In: CT-RSA 2002. (2002) 17–29 22
23. Lerman, L., Veshchikov, N., Picek, S., Markowitch, O.: On the construction of side-channel attack resilient s-boxes. In Guilley, S., ed.: COSADE 2017. Volume 10348 of LNCS., Springer (2017) 102–119 17
24. Li, L., Wu, W., Zheng, Y., Zhang, L.: The relationship between the construction and solution of the milp models and applications. Cryptology ePrint Archive, Report 2019/049 (2019) 23
25. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007) 21
26. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Inscrypt 2011. (2011) 57–76 23
27. Nahid Farhady Ghalaty, Bilgiday Yuce, P.S.: Analyzing the efficiency of biased-fault based attacks. Cryptology ePrint Archive, Report 2015/663 (2015) 22
28. National Institute of Standards and Technology (NIST): Advanced encryption standard (aes (2001) 9
29. Phan, R.C., Yen, S.: Amplifying side-channel attacks with techniques from block cipher cryptanalysis. In: CARDIS 2006. (2006) 135–150 20

30. Sakiyama, K., Sasaki, Y., Li, Y.: Security of Block Ciphers - From Algorithm Design to Hardware Implementation. Wiley (2015) 20
31. Sarkar, S., Sasaki, Y., Sim, S.M.: On the design of bit permutation based ciphers - the interplay among s-box, bit permutation and key-addition. In: IWSEC 2020. Volume 12231 of LNCS., Springer (2020) 3–22 16
32. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: EUROCRYPT 2017. (2017) 185–215 20
33. Selmke, B., Heyszl, J., Sigl, G.: Attack on a DFA protected AES by simultaneous laser fault injections. In: FDTC 2016. (2016) 36–46 5, 21
34. Simon, T., Batina, L., Daemen, J., Grosso, V., Massolino, P.M.C., Papagiannopoulos, K., Regazzoni, F., Samwel, N.: Friet: An authenticated encryption scheme with built-in fault detection. In Canteaut, A., Ishai, Y., eds.: 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. Volume 12105 of Lecture Notes in Computer Science., Springer (2020) 581–611 5
35. Stinson, D.R.: Cryptography - theory and practice. Discrete mathematics and its applications series. CRC Press (2006) 23
36. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In: ASIACRYPT 2014. (2014) 158–178 14, 23
37. Todo, Y.: Structural evaluation by generalized integral property. In: EUROCRYPT 2015. (2015) 287–314 21
38. Yen, S., Joye, M.: Checking before output may not be enough against fault-based cryptanalysis. IEEE Trans. Computers **49**(9) (2000) 967–970 22
39. Yen, S., Kim, S., Lim, S., Moon, S.: A countermeasure against one physical cryptanalysis may benefit another attack. In: ICISC 2001. (2001) 414–427 22
40. Zhang, F., Lou, X., Zhao, X., Bhasin, S., He, W., Ding, R., Qureshi, S., Ren, K.: Persistent fault analysis on block ciphers. IACR Transactions on Cryptographic Hardware and Embedded Systems (2018) 150–172 22

– Supplementary Material –

A Results

Proof of Lemma 1. Since both α and $\alpha \oplus a$ are solutions of $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$, for any δ, so we have $S(\alpha) \oplus S(\alpha \oplus \delta) = \Delta(\alpha, \delta) = S(\alpha \oplus a) \oplus S(\alpha \oplus a \oplus \delta)$. From this, we have, $S(\alpha) \oplus S(\alpha \oplus \delta) = S(\alpha \oplus a) \oplus S(\alpha \oplus a \oplus \delta) \implies S(\alpha \oplus \delta) \oplus S(\alpha \oplus \delta \oplus a) = S(\alpha) \oplus S(\alpha \oplus a)$. As the above relation holds for any δ so we can write $S(x) \oplus S(x \oplus a) = S(\alpha) \oplus S(\alpha \oplus a)$ holds for all x. This means that a is a linear structure of S. □

Proof of Lemma 2. We show that (a_1, a_2) is a linear structure of (S_1, S_2) if and only if a_1, a_2 are linear structures of S_1 and S_2 respectively. If (a_1, a_2) is a linear structure of (S_1, S_2), then for some constant $(c_1, c_2) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, $(S_1(x), S_2(y)) \oplus (S_1(x \oplus a_1), S_2(y \oplus a_2)) = (c_1, c_2)$, for all $x, y \in \mathbb{F}_2^n$. Therefore, $(S_1(x), S_2(0)) \oplus (S_1(x \oplus a_1), S_2(a_2)) = (c_1, c_2)$ for all $x \in \mathbb{F}_2^n$, from which we have $(S_1(x) \oplus S_1(x \oplus a_1), S_2(0)) \oplus S_2(a_2)) = (c_1, c_2)$, that is $S_1(x) \oplus S_1(x \oplus a_1) = c_1$ for all $x \in \mathbb{F}_2^n$. Therefore, a_1 is a linear structure of S_1. Similarly, it can be proved that a_2 is a linear structure of S_2. Conversely if a_1, a_2 are linear structures of S_1 and S_2 respectively, then it is easy to prove that (a_1, a_2) of (S_1, S_2). Thus the total number of linear structures of (S_1, S_2) is $\ell_1 \ell_2$ (including the trivial linear structure $(0, 0)$). □

Proof of Lemma 3. Suppose $a \in \mathbb{F}_2^n$ is a linear structure of $L \circ F$. Then we have $L(F(x)) \oplus L(F(x \oplus a)) = c$, for some constant $c \in \mathbb{F}_2^n$. This implies that $L((F(x) \oplus F(x \oplus a)) = c$, that is $F(x) \oplus F(x \oplus a) = L^{-1}(c)$. Therefore, a is also a linear structure of F.

To prove the converse; that is if a is a linear structure of F, then a is also a linear structure of $L \circ F$. Therefore, a is a linear structure of $L \circ F$ if and only if a is a linear structure of F. □

Definition 7 (Expanded DDT). *Expanded DDT of the SBox S is a matrix having the same dimension as its DDT, where entry corresponding to the input difference δ and output difference Δ is the set of solutions for the equation $S(x) \oplus S(x \oplus \delta) = \Delta$.*

Hence, Expanded DDT gives the actual solutions, whereas DDT only shows the cardinality of each solutions instead.

Theorem 4. *If an $n \times n$ SBox has l non-zero linear structures, then the minimum non-zero value in its DDT $= \min(2^n, 2l + 2)$.*

Proof. Let the l non-zero linear structures are $a_1, \ldots, a_l$. Hence, for $\delta \in \{a_1, \ldots, a_l\}$, the minimum non-zero value in the corresponding rows are 2^n (in fact, this is the only non-zero value in its DDT if the SBox is linear). For $\delta \notin \{a_1, \ldots, a_l\}$, the elements $\alpha, \alpha \oplus \delta, \alpha \oplus \alpha_1, \alpha \oplus \alpha_1 \oplus \delta, \ldots, \alpha \oplus \alpha_l, \alpha \oplus \alpha_l \oplus \delta$ $\in S_\alpha\langle\delta\rangle$. Hence the result follows. □

Theorem 4 indicates that if we want better protection against DFA (by increasing linear structures in the SBox), then at the same time the minimum entry in its DDT will also grow, making it harder to resist against differential attack.

Remark 4. By choosing an SBox for which large number of coordinate functions are affine, it is possible to get more LS. For an $n \times n$ SBox S, if it has c $(\leq n - 2)$ affine coordinate functions, then it can have 2^c linear structures. For example, the 8×8 SBox given by the coordinate functions, $S(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (x_0, x_1, x_2, x_3, x_4, x_5, x_0 x_1 \oplus x_6, x_0 x_1 \oplus x_7)$ has $2^6 = 64$ LS. As a side note, it can be mentioned that the SBox $S(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_0 x_1 \oplus x_7)$ has 64 LS too.

Remark 5. Although (in terms of maximum DFA security) the maximum number of LS an $n \times n$ SBox can have is 2^{n-1} (as 2^n LS would make the SBox linear), so far we are able to find SBoxes with at most 2^{n-2} LS (see Remark 4). Whether or not such SBox exists is thus an open problem. This problem can be considered orthogonal to the *Big APN Problem* [BDMW10].

Remark 6. Having a non-zero linear structure does not imply zero non-linearity. For example, the SBox `0123458967CDEFBA` has a linear structure at `1` but has non-linearity of 2.

Theorem 5. *If $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ has only two solutions, then there exists a δ', such that α is the unique common solution for $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ and $S(x) \oplus S(x \oplus \delta') = \Delta(\alpha, \delta')$.*

Proof. Consider $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ which has only two solutions, α and $\alpha \oplus \delta$. On the contrary assume that for all input differences d, if α is a solution of $S(x) \oplus S(x \oplus d) = \Delta(\alpha, d)$, so is $\alpha \oplus \delta$. Then by Lemma 1, we have that δ is a linear structure of S, which is a contradiction. Thus there will be a δ' such that α is a solution of $S(x) \oplus S(x \oplus \delta') = \Delta(\alpha, \delta')$, but $\alpha \oplus \delta$ is not. □

Remark 7. For simplicity, denote $\Delta(\alpha, \delta)$ by Δ and $\Delta(\alpha, \delta')$ by Δ'. If $|S_\alpha\langle\delta\rangle| = 2$, $S(\alpha) \neq 0$ and $S(\alpha \oplus \delta) \neq 0$, then δ' can be chosen as $\delta' = \alpha \oplus S^{-1}(\Delta)$, such that $S_\alpha\langle\delta\rangle \cap S_\alpha\langle\delta'\rangle = \alpha$. Notice that, in this case, $\Delta' = S(\alpha) \oplus S(\alpha \oplus \delta') = S(\alpha) \oplus \Delta = S(\alpha \oplus \delta)$.

Definition 8 **($D_\alpha\langle z\rangle$).** *For an SBox S with α as the input, $D_\alpha\langle z\rangle$ is defined as: $D_\alpha\langle z\rangle = \{$fault value d : z is a solution to $S(x) \oplus S(x \oplus d) = \Delta(\alpha, d)\}$, where z is an arbitrary input value.*

So, for $d \in D_\alpha\langle z\rangle$, we have, $\Delta(\alpha, d) = S(\alpha) \oplus S(\alpha \oplus d) = S(z) \oplus S(z \oplus d) = \Delta(z, d)$. In other words, the attacker will not be able to distinguish α from z under fault $d \in D_\alpha\langle z\rangle$.

Lemma 4. *Suppose, for the SBox S, $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ has exactly $2m + 2$ solutions, $\{\alpha, \alpha \oplus \delta, \beta_1, \beta_1 \oplus \delta, \ldots, \beta_m, \beta_m \oplus \delta\}$ where $2m + 2 \geq 4$. Then for any fault d ($\neq \delta$), define d^* as $d^* = d \oplus \delta$. We show the following results:*

(i) $d \in D_\alpha\langle\alpha \oplus \delta\rangle \implies d^* \in D_\alpha\langle\alpha \oplus \delta\rangle$;
(ii) $d \in D_\alpha\langle\beta_i\rangle \implies d^* \in D_\alpha\langle\beta_i\rangle$, *for* $i = 1, \ldots, m$;
(iii) $d \in D_\alpha\langle\alpha \oplus \delta\rangle \cap D_\alpha\langle\beta_i\rangle \implies d^* \in D_\alpha\langle\alpha \oplus \delta\rangle \cap D_\alpha\langle\beta_i\rangle$, *for* $i = 1, \ldots, m$.

Proof. Under fault d, we have:

$$\begin{aligned}
\Delta(\alpha, d) &= S(\alpha) \oplus S(\alpha \oplus d) \\
\Delta(\alpha \oplus \delta, d) &= S(\alpha \oplus \delta) \oplus S(\alpha \oplus \delta \oplus d) \\
\Delta(\beta_i, d) &= S(\beta_i) \oplus S(\beta_i \oplus d) \\
\Delta(\beta_i \oplus \delta, d) &= S(\beta_i \oplus \delta) \oplus S(\beta_i \oplus \delta \oplus d)
\end{aligned}$$

If (i) holds, then, $\Delta(\alpha, d) = \Delta(\alpha \oplus \delta, d)$; if (ii) holds, then, $\Delta(\beta_i, d) = \Delta(\beta_i \oplus \delta, d)$; if (iii) holds, then, $\Delta(\alpha, d) = \Delta(\alpha \oplus \delta, d) = \Delta(\beta_i, d) = \Delta(\beta_i \oplus \delta, d)$.

Similarly, under fault $d^* = d \oplus \delta$, we have:

$$\begin{aligned}
\Delta(\alpha, d^*) &= S(\alpha) \oplus S(\alpha \oplus d^*) = S(\alpha) \oplus S(\alpha \oplus \delta \oplus d) \\
\Delta(\alpha \oplus \delta, d^*) &= S(\alpha \oplus \delta) \oplus S(\alpha \oplus \delta \oplus d^*) = S(\alpha \oplus \delta) \oplus S(\alpha \oplus d) \\
\Delta(\beta_i, d^*) &= S(\beta_i) \oplus S(\beta_i \oplus d^*) = S(\beta_i) \oplus S(\beta_i \oplus \delta \oplus d) \\
\Delta(\beta_i \oplus \delta, d^*) &= S(\beta_i \oplus \delta) \oplus S(\beta_i \oplus \delta \oplus d^*) = S(\beta_i \oplus \delta) \oplus S(\beta_i \oplus d)
\end{aligned}$$

So, using previous relations, if (i) holds, then, $\Delta(\alpha, d^*) = \Delta(\alpha \oplus \delta, d^*)$; if (ii) holds, then, $\Delta(\beta_i, d^*) = \Delta(\beta_i \oplus \delta, d^*)$; if (iii) holds, then, $\Delta(\alpha, d^*) = \Delta(\alpha \oplus \delta, d^*) = \Delta(\beta_i, d^*) = \Delta(\beta_i \oplus \delta, d^*)$. □

Theorem 6. *Let the SBox S does not have any (non-zero) linear structure and $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ has exactly $2m + 2$ solutions; where $2m + 2 \geq 4$. Then there exist $m + 1$ faults $\{\delta', \delta_1, \ldots, \delta_m\}$, such that the system of equations*

$$\begin{aligned}
S(x) \oplus S(x \oplus \delta) &= \Delta(\alpha, \delta) \\
S(x) \oplus S(x \oplus \delta') &= \Delta(\alpha, \delta') \\
S(x) \oplus S(x \oplus \delta_1) &= \Delta(\alpha, \delta_1) \\
&\vdots \\
S(x) \oplus S(x \oplus \delta_m) &= \Delta(\alpha, \delta_m)
\end{aligned}$$

has a unique solution. Hence, $MinF_S(\alpha) \leq m + 2$.

Proof. For the input α and for the fault δ, the output difference is $S(\alpha) \oplus S(\alpha \oplus \delta) = \Delta(\alpha, \delta)$. Suppose $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$, has exactly $2m + 2$ solutions: $\{\alpha, \alpha \oplus \delta, \beta_1, \beta_1 \oplus \delta, \ldots, \beta_m, \beta_m \oplus \delta\}$. We will show that there exists a set of $m + 1$ faults $\{\delta', \delta_1, \ldots, \delta_m\}$ such that $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$; $S(x) \oplus S(x \oplus \delta') = \Delta(\alpha, \delta')$; $S(x) \oplus S(x \oplus \delta_1) = \Delta(\alpha, \delta_1)$; $\ldots$; and $S(x) \oplus S(x \oplus \delta_m) = \Delta(\alpha, \delta_m)$ have only one common solution α; i.e., $S_\alpha\langle\delta\rangle \cap S_\alpha\langle\delta'\rangle \cap S_\alpha\langle\delta_1\rangle \cap \cdots \cap S_\alpha\langle\delta_m\rangle = \{\alpha\}$. So, to uniquely identify α, one needs not more than $m + 2$ faults: $\{\delta, \delta', \delta_1, \ldots, \delta_m\}$.

Since, all of $\{\alpha, \alpha \oplus \delta, \beta_1, \beta_1 \oplus \delta, \ldots, \beta_m, \beta_m \oplus \delta\}$ are solutions of $\Delta(\alpha, \delta)$; we have:

$$\begin{aligned} \Delta(\alpha, \delta) = S(\alpha) \oplus S(\alpha \oplus \delta) = \Delta(\alpha \oplus \delta, \delta) \\ = \Delta(\beta_i, \delta) = S(\beta_i) \oplus S(\beta_i \oplus \delta) = \Delta(\beta_i \oplus \delta, \delta);\ \text{for } i = 1, \ldots, m. \end{aligned} \tag{1}$$

Consider the two sets of faults $D_\alpha\langle\alpha \oplus \delta\rangle$ and $D_{\alpha\beta} = \bigcup_{i=1}^m D_\alpha\langle\beta_i\rangle$. Note that $\delta \in D_\alpha\langle\alpha \oplus \delta\rangle \cap (\bigcap_{i=1}^m D_\alpha\langle\beta_i\rangle)$. Hence, none of the sets is empty. Define Ω_S to be the set of all $2^n - 1$ faults. We now treat rest of the proof in two cases.

Case 1. $D_\alpha\langle\alpha \oplus \delta\rangle \cup D_{\alpha\beta} \neq \Omega_S$.

Consider a $\delta' \in \Omega_S \setminus (D_\alpha\langle\alpha \oplus \delta\rangle \cup D_{\alpha\beta})$, then the equations, $S(x) \oplus S(x \oplus \delta) = \Delta(\alpha, \delta)$ and $S(x) \oplus S(x \oplus \delta') = \Delta(\alpha, \delta')$ will have α as a solution and $\beta_i \oplus \delta$'s could be other solution(s). If no $\beta_i \oplus \delta$ appears as a solution, then the two faults δ and δ' would be enough to determine α uniquely.

Now suppose that, for a set of i's, each of $\beta_i \oplus \delta$ is also included in the solution. Then, for each i, there must be one fault $\delta_i \in \Omega_S$ such that $\beta_i \oplus \delta$ is not a solution of $S(x) \oplus S(x \oplus \delta_i) = \Delta(\alpha, \delta_i)$; i.e., $\beta_i \oplus \delta \notin S_\alpha\langle\delta_i\rangle$. Otherwise if both α and $\beta_i \oplus \delta$ were common solutions of $S(x) \oplus S(x \oplus d) = \Delta(\alpha, d)$ for all input difference d; then by Lemma 1, $\alpha \oplus \beta_i \oplus \delta$ would be a linear structure of S; which is a contradiction. Therefore, these (at most $m + 2$) faults; δ, δ' and δ_i's will uniquely determine α.

Case 2. $D_\alpha\langle\alpha \oplus \delta\rangle \cup D_{\alpha\beta} = \Omega_S$.

First, we prove, $D_{\alpha\beta}$ is not a subset of $D_\alpha\langle\alpha \oplus \delta\rangle$. If possible, assume $D_{\alpha\beta} \subseteq D_\alpha\langle\alpha \oplus \delta\rangle$. Since, $D_\alpha\langle\alpha \oplus \delta\rangle \cup D_{\alpha\beta} = \Omega_S$ by assumption; we have $D_\alpha\langle\alpha \oplus \delta\rangle = \Omega_S$, implying δ is a linear structure; which is a contradiction. Hence, $D_{\alpha\beta} \setminus D_\alpha\langle\alpha \oplus \delta\rangle$ is non-empty.

Consider the fault: $\delta' \in D_{\alpha\beta} \setminus D_\alpha\langle\alpha \oplus \delta\rangle$. So, $S_\alpha\langle\delta\rangle \cap S_\alpha\langle\delta'\rangle$ can contain β_i's or $\beta_i \oplus \delta$'s along with α (but not $\alpha \oplus \delta$).

Next, we prove, $\beta_i \oplus \delta \notin S_\alpha\langle\delta\rangle \cap S_\alpha\langle\delta'\rangle$ by showing $\beta_i \oplus \delta \notin S_\alpha\langle\delta'\rangle$ for any i. Otherwise, if $\beta_i \oplus \delta \in S_\alpha\langle\delta'\rangle$ for some i, then, $\Delta(\alpha, \delta') = \Delta(\beta_i \oplus \delta, \delta')$; i.e.,

$$\begin{aligned} S(\alpha) \oplus S(\alpha \oplus \delta') &= S(\beta_i \oplus \delta) \oplus S(\beta_i \oplus \delta \oplus \delta') \\ \implies S(\alpha \oplus \delta') \oplus S(\beta_i \oplus \delta \oplus \delta') &= S(\alpha) \oplus S(\beta_i \oplus \delta) \\ &= S(\alpha \oplus \delta) \oplus S(\beta_i), \text{using Equation (1)} \\ \implies S(\alpha \oplus \delta) \oplus S(\alpha \oplus \delta') &= S(\beta_i) \oplus S(\beta_i \oplus \delta \oplus \delta') \\ &\implies \delta \oplus \delta' \in D_\alpha\langle\alpha \oplus \delta\rangle \cap D_\alpha\langle\beta_i\rangle \end{aligned}$$

Notice from Lemma 4 that, if $\delta \oplus \delta' \in D_\alpha\langle\alpha \oplus \delta\rangle \cap D_\alpha\langle\beta_i\rangle$, then so will be $(\delta \oplus \delta') \oplus \delta = \delta'$, which is a contradiction, as $\delta' \in D_{\alpha\beta} \setminus D_\alpha\langle\alpha \oplus \delta\rangle$.

So, with faults δ and δ', assume we have α and β_i's in the solution, for a set of i's. Now, consider fault(s) $\delta_i \in \Omega_S \setminus D_\alpha\langle\beta_i\rangle$. Consequently, the intersection, $S_\alpha\langle\delta\rangle \cap S_\alpha\langle\delta'\rangle \cap S_\alpha\langle\delta_i\rangle$, cannot contain any of $\{\alpha \oplus \delta, \beta_i, \beta_i \oplus \delta\}$. Besides, such fault δ_i exists; otherwise, we will have, $D_\alpha\langle\beta_i\rangle = \Omega_S$, implying $\alpha \oplus \beta_i$ is a linear structure; which is a contradiction. □

From Theorem 6, we see that, it is possible to uniquely recover the input/output value of each SBox with not more than $\frac{\mathrm{DU}_S}{2} + 1$ faults (unless there is a linear structure), when attacking the last round. This gives a provable upper bound for the attacker that he needs at most (if faults values are judiciously chosen) $\frac{\mathrm{DU}_S}{2} + 1$ faults per SBox in order the find out its input uniquely, given the SBox does not have a linear structure.

Corollary 1 (From Theorem 5, 6). $MinF_S \leq \dfrac{DU_S}{2} + 1$.

Remark 8. Although it is theoretically possible, we cannot find an SBox with $\mathrm{MinF}_S \geq 3$ (refer to Corollary 1). Whether or not this is the tight bound is left open for future research.

Theorem 7. *[Extension of [LCFS17, Appendix A.1]] Consider an $n \times n$ SBox S with input α. Suppose, upon applying two distinct faults δ_1 and δ_2, α is not uniquely retrieved; since, $\alpha \oplus \delta \in S_\alpha\langle\delta_1\rangle \cap S_\alpha\langle\delta_2\rangle$ ($\delta \neq \delta_1, \delta_2$).*

(i) If $\delta = \delta_1 \oplus \delta_2$:
$\alpha, \alpha \oplus \delta_1, \alpha \oplus \delta_2, \alpha \oplus \delta_1 \oplus \delta_2 \in S_\alpha\langle\delta_1\rangle \cap S_\alpha\langle\delta_2\rangle \cap S_\alpha\langle\delta_1 \oplus \delta_2\rangle$.
Hence, $|S_\alpha\langle\delta_1\rangle|$, $|S_\alpha\langle\delta_2\rangle|$ and $|S_\alpha\langle\delta_1 \oplus \delta_2\rangle| \geq 4$.

(ii) Else:
$\alpha, \alpha \oplus \delta_i, \alpha \oplus \delta, \alpha \oplus \delta \oplus \delta_i \in S_\alpha\langle\delta_i\rangle$ for $i = 1, 2$.
$\alpha, \alpha \oplus \delta, \alpha \oplus \delta_1, \alpha \oplus \delta_2, \alpha \oplus \delta \oplus \delta_1, \alpha \oplus \delta \oplus \delta_2 \in S_\alpha\langle\delta\rangle$.
Hence, $|S_\alpha\langle\delta_1\rangle|, |S_\alpha\langle\delta_2\rangle| \geq 4$ and $|S_\alpha\langle\delta\rangle| \geq 6$.

Proof. We have,

$$\begin{aligned}
S(\alpha) \oplus S(\alpha \oplus \delta_1) &= \Delta(\alpha, \delta_1) = \Delta(\alpha \oplus \delta, \delta_1) = S(\alpha \oplus \delta) \oplus S(\alpha \oplus \delta \oplus \delta_1), \\
S(\alpha) \oplus S(\alpha \oplus \delta_2) &= \Delta(\alpha, \delta_2) = \Delta(\alpha \oplus \delta, \delta_2) = S(\alpha \oplus \delta) \oplus S(\alpha \oplus \delta \oplus \delta_2), \\
\implies S(\alpha \oplus \delta_1) \oplus S(\alpha \oplus \delta_2) &= S(\alpha \oplus \delta \oplus \delta_1) \oplus S(\alpha \oplus \delta \oplus \delta_2).
\end{aligned}$$

(i) Given, $\delta = \delta_1 \oplus \delta_2$. Now we have, $S(\alpha) \oplus S(\alpha \oplus \delta_1) = S(\alpha \oplus \delta_2) \oplus S(\alpha \oplus \delta_1 \oplus \delta_2) \implies \alpha \oplus \delta_2, \alpha \oplus \delta_1 \oplus \delta_2 \in S_\alpha\langle\delta_1\rangle$.
Again, $S(\alpha) \oplus S(\alpha \oplus \delta_2) = S(\alpha \oplus \delta_1) \oplus S(\alpha \oplus \delta_1 \oplus \delta_2) \implies \alpha \oplus \delta_1, \alpha \oplus \delta_1 \oplus \delta_2 \in S_\alpha\langle\delta_2\rangle$.
Also, $S(\alpha) \oplus S(\alpha \oplus \delta_1) = S(\alpha \oplus \delta_2) \oplus S(\alpha \oplus \delta_1 \oplus \delta_2) \implies S(\alpha) \oplus S(\alpha \oplus \delta_1 \oplus \delta_2) = S(\alpha \oplus \delta_1) \oplus S(\alpha \oplus \delta_2)$ $\implies \alpha \oplus \delta_1, \alpha \oplus \delta_2 \in S_\alpha\langle\delta_1 \oplus \delta_2\rangle$.

(ii) $\delta \neq \delta_1 \oplus \delta_2$.
We get, $S(\alpha) \oplus S(\alpha \oplus \delta_i) = S(\alpha \oplus \delta) \oplus S(\alpha \oplus \delta \oplus \delta_i) \implies \alpha \oplus \delta, \alpha \oplus \delta \oplus \delta_i \in S_\alpha\langle\delta_i\rangle$; for $i = 1, 2$.
Next, $S(\alpha) \oplus S(\alpha \oplus \delta) = S(\alpha \oplus \delta_1) \oplus S(\alpha \oplus \delta \oplus \delta_1) \implies \alpha \oplus \delta_1, \alpha \oplus \delta \oplus \delta_1 \in S_\alpha\langle\delta\rangle$. Similarly, $\alpha \oplus \delta_2, \alpha \oplus \delta \oplus \delta_2 \in S_\alpha\langle\delta\rangle$. ◻

Example 1. The SBox `80A23517496BCDEF` (the representative of class # 293 in [De 07]) has a linear structure at $a = 2$, which can be seen from its DDT in Table 8(a). Zeros, and the rows-columns corresponding to $\delta = 0$ and $\Delta = 0$ are not shown here for the sake of better clarity. Table 8(b) gives the $S_0\langle\delta\rangle$ for varying δ on S.

Table 8: Effect of DFA on SBox `80A23517496BCDEF`

(a) DDT

δ \ Δ	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	4					4		4					4		
2		16													
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								4	4			4			4
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	

(b) Effect of fault for input $\alpha = 0$

δ	$S_0\langle\delta\rangle$	$\lvert S_0\langle\delta\rangle\rvert$
1	0, 1, 2, 3	4
2	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f	16
3	0, 1, 2, 3	4
4	0, 2, 4, 6	4
5	0, 2, 5, 7	4
6	0, 2, 4, 6	4
7	0, 2, 5, 7	4
8	0, 2, 8, a	4
9	0, 2, 9, b	4
a	0, 2, 8, a	4
b	0, 2, 9, b	4
c	0, 2, c, e	4
d	0, 2, d, f	4
e	0, 2, c, e	4
f	0, 2, d, f	4

Table 9: Expanded DDT for SBox `20135467A98BCDEF`

δ \ Δ	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	4567cdef	0123	89ab												
2		89abcdef	01234567												
3	012389ab	4567	cdef												
4				13579bdf		8ace	0246								
5					13469bce	0257	8adf								
6				02468ace		9bdf	1357								
7					02578adf	1346	9bce								
8								03678bef	12459acd						
9								45cd	67ef	128b	039a				
a										03679acd	12458bef				
b								129a	038b	45ef	67cd				
c												37bf	159d	06ac	248e
d												149c	36be	258f	07ad
e												068e	24ac	379d	15bf
f												25ad	078f	14be	369c

Example 2. Take the SBox `20135467A98BCDEF` (# 288 in [De 07]). Notice from the expanded DDT (Table 9) that, for fault $\delta = 1$ (column 3), we have, $S_8\langle 1\rangle = S_9\langle 1\rangle = S_a\langle 1\rangle = S_b\langle 1\rangle = \{8, 9, a, b\}$.

Example 3. Consider the previous SBox `20135467A98BCDEF` (expanded DDT is in Table 9). Assume the input, $\alpha = 8$. If an attacker inserts the fault values $\delta_1 = d$ and then $\delta_2 = 1$, then he is able to uniquely retrieve the input $\alpha = 8$; since $S_8\langle d\rangle \cap S_8\langle 1\rangle = \{2, 5, 8, f\} \cap \{8, 9, a, b\} = \{8\}$. So, $\mathrm{MinF}_{\texttt{20135467A98BCDEF}}(8) = 2$.

Example 4 (Theorem 7(ii)). Consider the SBox `20135467A98BCDEF` whose expanded DDT is in Table 9. Let, $\alpha = 8, \delta_1 = 1, \delta_2 = 4$ with $\delta = 2 \neq \delta_1 \oplus \delta_2$. We have $|S_{\delta_1}\langle\alpha\rangle| = |\{8, 9, a, b\}| = 4$ and $|S_{\delta_2}\langle\alpha\rangle| = |\{8, a, c, e\}| = 4$ and $|S_{\delta}\langle\alpha\rangle| = |\{8, 9, a, b, c, d, e, f\}| \geq 6$.

B Test Vectors for `DEFAULT`

In Table 10, we provide a few test vectors for the 80-round cipher `DEFAULT` (i.e., with the `DEFAULT-LAYER` – `DEFAULT-CORE` – `DEFAULT-LAYER` construction).

Table 10: Test vectors for `DEFAULT` (full cipher with 80 rounds)

	Key	00000000000000000000000000000000
1	Plaintext	00000000000000000000000000000000
	Ciphertext	02ec558a8f65dc7e53b326a1de9ade51
	Key	33333333333333333333333333333333
2	Plaintext	33333333333333333333333333333333
	Ciphertext	14df4c184c70c5243ffe4c4044b04d1f
	Key	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
3	Plaintext	55555555555555555555555555555555
	Ciphertext	00db5016a9bebf70f21ea1dd7a6fb56a
	Key	fedcba9876543210fedcba9876543210
4	Plaintext	fedcba9876543210fedcba9876543210
	Ciphertext	bbfa59b71f67bd80a8b647fe72d327a9
	Key	d728395021c3c50433b4ff70ee7cc270
5	Plaintext	47115d03a7ad328d998e1957f166ec5c
	Ciphertext	bc27f7ad16133a77d0455fdfe7a70a56

C Visual Representation of Two Rounds of `DEFAULT-LAYER`

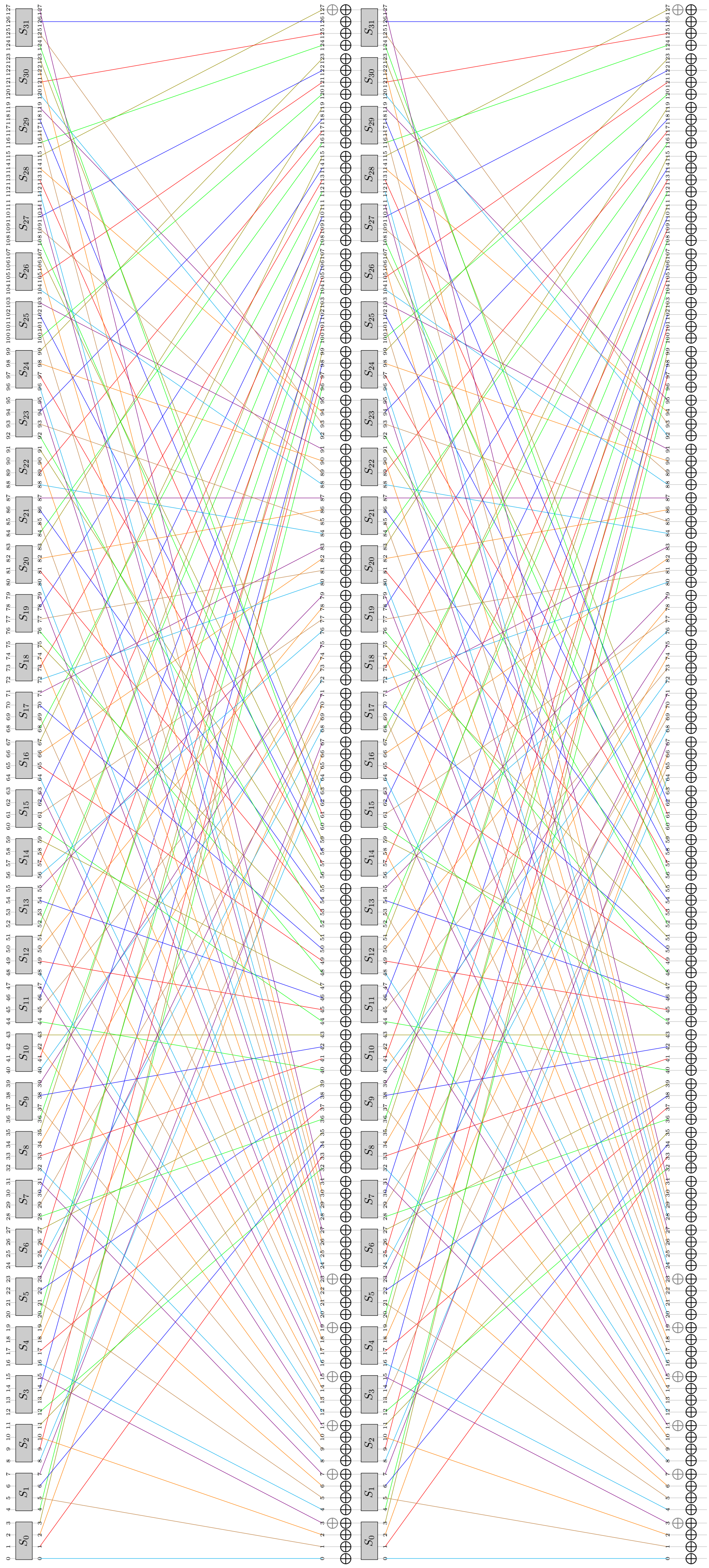
We recall in Table 11 the `GIFT-128` bit permutation that is used in our constructions.

The structure of `DEFAULT-LAYER` for two rounds is shown (out of 28 rounds in `DEFAULT-LAYER`) in the Figure 5. The structure for the `DEFAULT-CORE` is also similar. The SBoxes are numbered (from

Table 11: Specifications of `GIFT-128` Bit Permutation.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P_{128}(i)$	0	33	66	99	96	1	34	67	64	97	2	35	32	65	98	3
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P_{128}(i)$	4	37	70	103	100	5	38	71	68	101	6	39	36	69	102	7
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P_{128}(i)$	8	41	74	107	104	9	42	75	72	105	10	43	40	73	106	11
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P_{128}(i)$	12	45	78	111	108	13	46	79	76	109	14	47	44	77	110	15
i	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
$P_{128}(i)$	16	49	82	115	112	17	50	83	80	113	18	51	48	81	114	19
i	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
$P_{128}(i)$	20	53	86	119	116	21	54	87	84	117	22	55	52	85	118	23
i	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
$P_{128}(i)$	24	57	90	123	120	25	58	91	88	121	26	59	56	89	122	27
i	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
$P_{128}(i)$	28	61	94	127	124	29	62	95	92	125	30	63	60	93	126	31

0 to 31), so are the bit positions (from 0 to 127), for better clarity. The colored lines indicate the permutation (linear) layer. By $\bigoplus$, it is indicated that the corresponding key bits are XORed. The round constant additions are shown by $\bigoplus$ (bits at 3, 7, 11, 15, 19, 23), and the bit at 127 is flipped in each round.



7

Fig. 5: Structure of `DEFAULT-LAYER` (two rounds only)

D MILP modeling

As our special SBox with linear structures has probability 1 differential transitions (resp., $\pm\frac{1}{2}$ linear bias). For the differential case, the above mentioned approach will always yield an MEDP bound of $\epsilon_d = 1$ (1 is raised to the power of an integer), which naturally signifies the smallest possible protection against differential attacks (the attack succeeds with only one chosen input difference or two chosen inputs). In case of linear cryptanalysis, it can be shown that the overall bias ϵ_l, considering only $\pm\frac{1}{2}$ biases (and assuming mutual independence of the biases), is $\frac{1}{2}$. This is obtained by substituting $\epsilon_i = \frac{1}{2}$ $\forall i$ in [Sti06, Lemma 3.1]. Similar to the differential case, this also leads to the smallest protection against linear attack (the attack succeeds with roughly $1/\epsilon_l^2 = 4$ known inputs). Naturally, we need to devise a way to count precisely the number of probability $\frac{1}{2}$ differential transitions and $\pm\frac{1}{4}$ linear biases.

To overcome this problem, we devise a new strategy which is inspired from the concept of *indicator constraint* used in linear programming (also known as the *big M* method), where a large constant M is chosen. In our case, it is sufficient to choose M being equal to twice the SBox size ($= 8$), similar to [AST+17]. We would like to note that the MILP modeling used here is the first-of-its-kind for its compatibility with LS SBoxes. Our basic idea is to minimize the number of active SBoxes with differential probability $\frac{1}{2}$ (resp., $\pm\frac{1}{4}$ linear bias) while keeping the probability 1 differential transitions (resp., $\pm\frac{1}{2}$ linear bias) unrestricted for a particular number of rounds. The problem can be formulated as an MILP problem whose solution can be obtained by a standard solver. Upon getting the solution, let us denote the number active SBoxes with differential probability 1 by o and that of differential probability $\frac{1}{2}$ by h. Then, the MEDP can be computed as $1^o \times (\frac{1}{2})^h = 2^{-h}$ and hence the attacker needs at least 2^h chosen differences [Sti06, Chapter 3.4], which translates to 2^{1+h} chosen inputs for the layer. To compute the MELP, let us assume the number of active SBoxes with $\pm\frac{1}{2}$ bias is o and that with $\pm\frac{1}{4}$ is h. Then, we substitute with $\epsilon_i = \frac{1}{2}$ or with $\epsilon_i = \frac{1}{4}$ accordingly to get $\epsilon_l = 2^{o+h-1} \times (\frac{1}{2})^o \times (\frac{1}{4})^h$ $= 2^{-h-1}$. Hence the attacker needs roughly $1/\epsilon_l^2 = 2^{2h+2}$ known inputs [Sti06, Chapter 3.3]. Of course, the probability 1 differential transitions as well as the $\pm\frac{1}{2}$ linear biases do not play any role in the search complexity.

In the following, we describe the MILP modeling in details for the differential case. The MILP formulation for the linear case is much alike, hence we skip the details for conciseness (the main differences is that in the linear case the absolute values for the biases are considered). More information about this modeling can be found in [Bak20].

Assume that there are q_p transitions for a given probability p ($1 \geq p > 0$). For example, there are three probability 1 transitions for an SBox with three non-zero LS, hence $q_1 = 3$.

First, for the i^{th} SBox ($i = 0, 1, \ldots, 31$) at the j^{th} round ($j = 0, 1, \ldots, \eta - 1$), we create the following Boolean variables:

$Q_{i,j}$	to indicate it is active;
$Q^p_{i,j}$	to indicate if it takes a probability p trail;
$Q^p_{i,j,l}$, for $l = 0, \ldots, q_p - 1$	to indicate which among the q_p trails (probability p each) is chosen;
$\boldsymbol{x}_{i,j} = (x^0_{i,j}, x^1_{i,j}, x^2_{i,j}, x^3_{i,j})$	to indicate the input difference;
$\boldsymbol{y}_{i,j} = (y^0_{i,j}, y^1_{i,j}, y^2_{i,j}, y^3_{i,j})$	to indicate the output difference.

Next, we set the constraints for each SBoxes:

$MQ_{i,j} \geq \sum_{l=0}^{3} x^l_{i,j} + \sum_{l=0}^{3} y^l_{i,j}$	to check if it is active;
$Q_{i,j} = \sum_p Q^p_{i,j}$	to keep track which probability p trail if active;

For each probability transition p, do:

$Q^p_{i,j} = \sum_{l=0}^{q_p-1} Q^p_{i,j,l}$	to check precisely which p probability trail is chosen.

After this, each $Q^p_{i,j,l}$ is used to model respective transitions. For example, the probability 1 transition $(\mathtt{6}, \mathtt{a})$ is the $l = 2$ trail, it is modelled as: $MQ^1_{i,j,2} \geq (x^0_{i,j}) + (1 - x^1_{i,j}) + (1 - x^2_{i,j}) + (x^3_{i,j}) + (1 - y^0_{i,j}) + (y^1_{i,j}) + (1 - y^2_{i,j}) + (y^3_{i,j})$. Basically, each negative literal is taken as is, and each positive literal is subtracted from 1, then added together.

Also, we have to set a non-zero initial input difference to at least one variable at the beginning ($j = 0$): $\sum_{i=0}^{31} \sum_{l=0}^{3} x^l_{i,0} \geq 1$.

The last set of constraints comes from the bit permutation layer. For each round from 1 to $\eta - 1$ (for $j = 1, \ldots, \eta - 1$), 128 equality constraints are inserted. For example, the second entry in the permutation $(1 \rightarrow 33)$ is modeled as $x^1_{8,j} = y^1_{0,j-1}$.

Finally, we formulate our MILP problem with the objective function:

$$\text{Minimize } \sum_{i=0}^{31} \sum_{j=0}^{\eta-1} \sum_{p<1} (-\log_2 p) \times Q^p_{i,j}.$$

References

AST+17. Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP modeling for (large) s-boxes to optimize probability of differential characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4):99–129, 2017. 8

Bak20. Anubhab Baksi. New insights on differential and linear bounds using mixed integer linear programming (full version). Cryptology ePrint Archive, Report 2020/1414, 2020. https://eprint.iacr.org/2020/1414. 8

BDMW10. KA Browning, JF Dillon, MT McQuistan, and AJ Wolfe. An apn permutation in dimension six. *Finite Fields: theory and applications*, 518:33–42, 2010. 1

De 07. Christophe De Cannière. *Analysis and Design of Symmetric Encryption Algorithms*. Katholieke Universiteit Leuven, Belgium, 2007. PhD Thesis. 4, 5

LCFS17. Benjamin Lac, Anne Canteaut, Jacques Fournier, and Renaud Sirdey. *DFA on LS-Designs with a Practical Implementation on SCREAM*, pages 223–247. Springer International Publishing, Cham, 2017. 4

Sti06. Douglas R. Stinson. *Cryptography - theory and practice*. Discrete mathematics and its applications series. CRC Press, 2006. 8