Attacks and Countermeasures in Persistent Fault Model

Viet Sang Nguyen, Vincent Grosso and Pierre-Louis Cayrel

Laboratoire Hubert Curien, Université Jean Monnet, Saint-Étienne, France

Abstract. Persistent fault attacks have recently emerged as a significant area of research in embedded cryptography. In a persistent fault model, fault injection targets constants stored in memory, such as ROM. This type of fault persists across multiple encryptions and only disappears when the device is reset. Previous works in the literature assume that a table of S-box elements is stored in the memory and consider a model where fault injection results in a *biased faulty S-box*. This means that one or several elements appear multiple times while one or several others disappear, leading to non-uniform distributions of ciphertext words that can be exploited by efficient statistic methods. Few countermeasures were proposed to detect such biases in the faulty S-box. However, the current fault model does not account for other severe consequences of persistent faults. Our work aims to address this gap.

In this work, we extend the previous model in two ways. First, we consider persistent faults causing a swap of two or three S-box elements (*non-biased faulty S-box*). We demonstrate, using the PRESENT cipher, that an attacker can bypass existing countermeasures and recover the key by applying a linear attack. Second, we show that S-box is not the only target for fault injection, as assumed by most of previous works. We consider a persistent fault induced on a round constant of the AES cipher and demonstrate that the key can be efficiently recovered by applying a differential fault attack. Notably, we reduce the typical statistical analysis of previous works, which requires from few hundreds to few thousands of ciphertexts, to a differential analysis needing only two correct-faulty ciphertext pairs. Finally, we present a more efficient countermeasure which can detect persistent faults that the existing countermeasures cannot.

Keywords: Fault Attacks · Persistent Fault Analysis · Countermeasures

1 Introduction

Attacking implementations through fault injection is an active research area in embedded cryptography. These attacks exploit execution errors in cryptographic algorithms, typically caused by an attacker injecting faults, to recover secret information. The consequences of such faults can be disastrous, rendering the system completely insecure. A fault attack consists of two phases: fault injection and fault analysis. In the first phase, the attacker induces faults into the target device in order to disturb the algorithm's execution. Faults can be induced using various techniques, such as laser pulses, electromagnetic pulses or clock glitches. In the second phase, the attacker collects the faulty outputs from the device and analyzes them to recover the secret key.

The idea of fault attack was first introduced by Boneh *et al.* [BDL97] with an application to RSA. Subsequently, Biham and Shamir [BS97] proposed Differential Fault Analysis

E-mail: viet.sang.nguyen@univ-st-etienne.fr (Viet Sang Nguyen), vincent.grosso@univ-st-etienne.fr (Vincent Grosso), pierre.louis.cayrel@univ-st-etienne.fr (Pierre-Louis Cayrel)

Date of this document: 2024-09-17.

(DFA) with an application to DES. Since then, DFA has become one of the most common fault attack, applicable to many block ciphers such as AES [PQ03, DLV03, KQ08, TMA11], DES [Riv09], PRESENT [WW10]. Over time, many other effective fault attacks have been developed using different techniques. Collision Fault Analysis (CFA) involves searching for a plaintext whose corresponding ciphertext collides with some faulty ciphertexts. Ineffective Fault Analysis (IFA) [Cla07] aims to find a plaintext such that the faulty intermediate value still results in a correct ciphertext. Statistical Fault Analysis (SFA) [FJLT13] recovers the secret key using statistical models with only faulty ciphertexts. Differential Fault Intensity Analysis (DFIA) [GYTS14] combines the principles of Differential Power Analysis and fault injection. Fault Sensitivity Analysis (FSA) [LSG+10] does not use values of faulty ciphertexts, instead, it exploits detectable characteristics, *e.g.*, the clock frequency at which a faulty operation begins to occur. Statistical Ineffective Fault Analysis (SIFA) [DEK+18] operates in the presence of detection-based and ineffective countermeasures, exploiting the non-uniform distribution of intermediate values that lead to fault-free ciphertexts. Fault Template Attack (FTA) *et al.* [SBR+20] leverages the data-dependent nature of fault activation and propagation through a combinational circuit.

Depending on the duration of the effect, faults can be classified into three categories: *transient faults*, *persistent faults* and *permanent faults*. A transient fault affects the execution in a very short period, typically during a single encryption. This means that a transient fault causes errors in only one execution and does not persist in subsequent executions. Most, if not all, of the fault attacks mentioned above are proposed within the transient fault settings. A permanent fault, on the other hand, has a lasting effect on the target and cannot be erased. Persistent faults, on which we focus in this work, fall between the other two categories. A fault of this type persists across different executions but is erased once the device is reset. Compared to attacks based on transient faults, a persistent fault attack has several advantages: (1) the attacker does not need to inject faults in a precise timing or location with live synchronization of different executions; (2) fault injection and ciphertext collection can be done in two different phases; (3) it can bypass some redundancy-based countermeasures.

The notion of persistent faults was introduced by Schmidt *et al.* [SHP09] with an application to attacking AES. Recently, persistent fault analysis (PFA) has gained significant attention since the work of Zhang *et al.* [ZLZ+18] at CHES 2018. In this work, the authors developed a dedicated model for the persistent fault setting and proposed a statistical technique to recover the key in AES-128. The model assumes that substitution-permutation network (SPN)-based ciphers are realized by table-based implementations and that the faults affect one or multiple constants, typically S-box elements, stored in memory (*e.g.*, ROM). Specifically, the faults result in a *biased faulty S-box*, where one or several S-box elements appear more frequently while one or several others disappear. These faults persist across different encryptions until the device is reset. Building on this model of Zhang *et al.* [ZLZ+18], many follow-up works have either aimed to reduce the number of encryptions required for the analysis phase [CGR20, XZY+21, ZZJ+20, SBH+22, ZFL+22, ZHF+23] or to apply this model to attack different (protected) ciphers [PZRB19, GPT19, TL22, ZFL+22].

We recall that the above persistent fault attacks are all based on the model of Zhang *et al.* [ZLZ+18], *i.e.*, based on a biased faulty S-box. In [ZLZ+18], Zhang *et al.* also pointed out that the popular transient-fault countermeasures based on spatial redundancy and temporal redundancy can be circumvented by persistent faults. This raised the need for countermeasures specifically designed to detect persistent faults. Since then, several countermeasures have been proposed [TBG23, CB19], employing different techniques but sharing the same principal: detecting biases in the faulty S-box. From these observations, we raised the following research questions:

1. *Is it possible to bypass the existing countermeasures with a* non-biased faulty S-box *(i.e., the faulty S-box where two or several elements are swapped because of the faults'*

effect)?

2. *Is it possible to bypass the existing countermeasures by faulting other constants rather than the S-box elements? Saying differently, are the S-box elements the only ones that can be faulted in the persistent fault model?*
3. *If the answers for the above two questions are positive, can we still perform some analyses to recover the key?*
4. *If the answers for the above three questions are positive, can we develop a stronger countermeasure?*

Contributions. In this paper, we provide the positive answers to the four questions outlined above. Specifically, we extend the persistent fault model of Zhang *et al.* [ZLZ$^+$18] in two ways, including (1) persistent faults resulting in a swap of two or three S-box elements (*non-biased faulty S-box*) and (2) persistent faults induced on other constants rather than S-box elements. In this extended model, we show that an attacker can bypass the existing countermeasures dedicated to detect biases in an S-box [TBG23, CB19]. Regarding the first extension, we demonstrate, using the PRESENT cipher, that an attacker still can recover the key by applying a linear analysis after bypassing these countermeasures. Regarding the second extension, we investigate a persistent fault induced into a round constant of the AES cipher and show that an attacker can recover the key by applying a differential fault analysis. This also reduces the statistical analyses of many previous works following the Zhang *et al.*'s model, which require from few hundreds to few thousands of ciphertexts, to a differential analysis needing only two correct-faulty ciphertext pairs. Finally, we present a stronger countermeasure in the extended model, addressing vulnerabilities that the existing ones cannot cover.

To the best of our knowledge, this work is the first to study a non-biased faulty S-box and faults applied to constants other than S-box elements. Our focus is on the persistent fault model and the analysis phase, rather than the fault injection phase in hardware. We aim to highlight potential risks in scenarios where existing countermeasures only pay attention to detect the biases in a faulty S-box (as the current analyses in the literature only focus on a biased faulty S-box). Furthermore, by extending the previous model, we hope to open up new research directions for persistent fault attacks from both hardware and cryptographic perspectives.

Outline. The paper is organized as follows. Section 2 presents related works in the field. Section 3 describes the extended persistent fault model used for our analyses. Section 4 explains how an attacker can bypass the existing countermeasures in the extended model. Section 5 details the linear attack on the full-round PRESENT cipher with a non-biased faulty S-box. Section 6 presents the differential fault attack on the AES-128 cipher with a persistent fault induced in a round constant. Section 7 presents a robust countermeasure. Finally, Section 8 concludes our work and offers some perspectives for future research.

2 Related works

In this section, we present an overview of existing persistent fault models, analyses, countermeasures and practical injection techniques.

2.1 Persistent fault model and analysis

At CHES 2018, Zhang *et al.* [ZLZ$^+$18] introduced a model dedicated to persistent faults and a novel analysis called *persistent fault analysis* (PFA). In this model, the S-box is assumed to be implemented as a lookup table and stored in memory. A single fault on an S-box element v alters this value to the faulty value $v' \neq v$. Since S-box is a permutation, v no longer appears in the S-box, while v' appears twice as often. Consequently, one value will never be observed in each ciphertext word, and another value will be observed twice as often. This results in a non-uniform probability distribution for each ciphertext word. If an attacker collects a sufficiently large number of ciphertexts, he can recover the last round key through statistical analysis. We note that, both the fault value (*i.e.*, $v \oplus v'$) and the fault location (*i.e.*, the position of v) are assumed to be known in this model.

Many follow-up works have been proposed based on the same model introduced by Zhang *et al.* [ZLZ+18]. Carré *et al.* [CGR20] reduced the number of ciphertexts needed for the analysis on AES by applying maximum likelihood estimation. Pan *et al.* [PZRB19] showed that PFA can break higher-order masking schemes with a single persistent fault and showcased on the masked implementations of the AES and PRESENT ciphers. Note that to apply PFA, they assumed that (part of) the masked S-box computation is realized as a lookup table. Gruber *et al.* [GPT19] applied PFA to the authenticated encryption schemes OCB, DEOXYS and COLM. Xu *et al.* [XZY+21] enhanced PFA by extending the analysis to deeper middle rounds. Caforio and Banik [CB19] constructed PFA on generic Feistel schemes, with an additional requirement for the model that an attacker can collect both correct and faulty ciphertexts, *i.e.*, encrypt fixed plaintexts twice.

At CHES 2020, Zhang *et al.* [ZZJ+20] relaxed the assumption of knowing the fault value and the fault location for the case of a single fault with applications on the AES and PRESENT ciphers. For multiple faults, both [ZLZ+18] and [ZZJ+20] (double faults) presented analysis methods, however, the values and locations of the faults need to be known in both works. This assumption for the case of multiple faults was then relaxed by Engels *et al.* [ESP20] and Soleimany *et al.* [SBH+22] with applications on the AES and LED ciphers. Zheng *et al.* [ZLZ+21] and Zhang *et al.* [ZHF+23] proposed a collision analysis and chosen-plaintext analysis, respectively, which operate under a relatively relaxed model that does not require any information about the fault value, the fault location, or the number of faults.

2.2 Countermeasures

In [ZLZ+18], Zhang *et al.* showed that traditional fault detection approaches, such as temporal redundancy (*e.g.*, run the encryption twice, then compare the two ciphertexts) and spatial redundancy (*e.g.*, use two encryption modules in hardware, then compare the two outputs) are ineffective against persistent fault. This is due to the nature of persistent faults, which persist across multiple executions until the device is rebooted. Consequently, both executions use the same faulty S-box and produce identical outputs. This underscores the necessity for countermeasures specifically designed to detect persistent faults.

The above PFAs are either in a single-fault model or a multiple-faults model, however, they all rely on the assumption that the faulty S-box is biased. In the literature, few countermeasures have been proposed to detect persistent faults using the same principle of identifying such biases. Caforio et Banik [CB19] compared the S-box inputs pairwise and then compared the corresponding S-box outputs pairwise. If two different input values pass through the S-box and result in the same output value, a bias (fault) is detected. Recently, Tissot *et al.* [TBG23] observed that a permutation (S-box) forms a unique set of cycles (see Figure 2 for an example of PRESENT's S-box). If there is a bias (fault) in an S-box, this set of cycles will change. To the best of our knowledge, these are the two state-of-the-art countermeasures dedicated to detect persistent faults in an S-box.

2.3 Persistent fault injection

While transient faults [BDL97, ABF$^+$03] and permanent faults [YMH03] have been investigated since the late of nineties and early of twenties, persistent faults have only garnered attention over the past decade. Schmidt *et al.* [SHP09] reported that irradiating ultraviolet (UV) for a few minutes can flip bits from 0 to 1 in various types of non-volatile memory (EPROM, EEPROM, FLASH). They also demonstrated a real attack on AES by faulting an S-box element. In 2014, Kim *et al.* [KDK$^+$14] exposed persistent bit flips on DRAM using the rowhammer technique, successfully inducing errors in most DRAM modules from major manufacturers. In [ZZJ$^+$20], Zhang *et al.*'s experiment on the SRAM of an ATmega163L microcontroller showed that a single laser pulse can flip two adjacent bits. Soleimany *et al.* [SBH$^+$22] experimented with electromagnetic fault injection (EMFI) on an STM32F407VG microcontroller. The EM pulse more likely affects multiple S-box elements (3-5 elements for the LED S-box and 4-6 elements for the AES S-box). Notably, Selmke *et al.* [SBHS15] demonstrated their experiments of flipping bits *at precise locations* into 90 nm and 45 nm SRAM-cells. This can be considered an important step closer to achieving multiple precise faults in memory.

3 Persistent fault model

In this section, we present the persistent fault model used in our subsequent analyses. As discussed earlier, most, if not all, previous PFAs are based on a biased faulty S-box. Moving towards a new research direction, we investigate PFA with a non-biased faulty S-box and PFA with another faulty constant (*e.g.*, a round constant) rather than S-box elements. We now formalize a model for an attacker to obtain the required faulty values.

Our model has some similarities to those in previous works. We describe the model as follows:

– Like any other PFA, the target cipher is assumed to be realized in a table-based implementation. S-box elements are stored as a lookup table in memory such as ROM, SRAM, DRAM, depending on the devices. Additionally, we assume that other algorithm constants (*e.g.*, round constants in AES) are stored as a lookup table in memory.

– An attacker can inject a single or multiple (typically, 2 or 3) precise faults (*i.e.*, known values and known locations) into the above values stored in memory. These faults persist across different encryptions until the device is reset. This allows the attacker to obtain a non-biased faulty S-box (Section 5), and to fault a specific round constant (Section 6). Note that the assumption of knowing a single or multiple fault values and locations was also used in the models of Zhang *et al.* [ZLZ$^+$18] and the model of Zhang *et al.* [ZZJ$^+$20] (for double faults only).

– An attacker has access both plaintexts and ciphertexts. This access enables the attacker to perform linear analysis with a non-biased faulty S-box (Section 5). Note that this assumption loses the ciphertext-only advantage of some PFAs (*e.g.*, Zhang *et al.* [ZLZ$^+$18], Soleimany *et al.* [SBH$^+$22]), however, it was also used in some other PFAs (*e.g.*, Zhang *et al.* [ZHF$^+$23], Zheng *et al.* [ZLZ$^+$21], Caforio and Banik [CB19]).

– If necessary, the attacker can encrypt a plaintext twice, once with faults and once without, to obtain a correct-faulty ciphertext pair. This enables the attacker to perform a differential analysis (Section 6). Note that this assumption was also used in the model of Caforio and Banik [CB19].

Table 1: S-box of PRESENT cipher (also used in LED cipher).

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\mathbf{S}[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

4 Bypassing countermeasures against biased S-boxes

In this section, we show how an attacker in our model can bypass the state-of-the-art countermeasures proposed Caforio and Banik [CB19] and Tissot *et al.* [TBG23]. Recall that there are two main extensions in our model: a non-biased faulty S-box and a fault on another constant (*e.g.*, a round constant of AES). For the latter extension, it is clear that an attacker can bypass these countermeasures since they are only designed to detect faults in an S-box. We now present how to bypass the two countermeasures using a non-biased faulty S-box.

4.1 Caforio and Banik's countermeasure [CB19]

Description of countermeasure. Suppose that one or several S-box elements are altered by faults. It turns out that at least two elements at indexes i and j bear the same value, $\mathbf{S}[i] = \mathbf{S}[j]$, where $i \neq j$. In other words, if two different values enter the S-box layer and result in the same value at the two outputs, an error must be thrown. The main idea of this countermeasure is to compare pairwise the input values and then compare pairwise the corresponding output values. The comparison bit vectors of the inputs and the outputs are then compared one more time to check if there exists an error in the S-box. Figure 1 depicts the construction of this countermeasure.

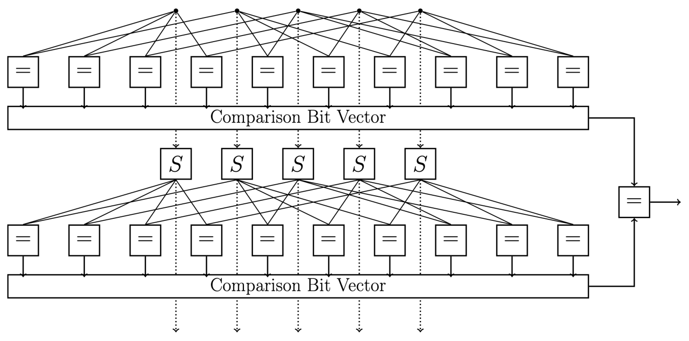

Figure 1: Pairwise comparison network [CB19]

Bypassing countermeasure. We notice that this countermeasure is dedicated to detecting biases in an S-box. Given a non-biased faulty S-box (a permutation) where two elements are swapped, the countermeasure can obviously be bypassed. Indeed, we have $\mathbf{S}[i] \neq \mathbf{S}[j]$ for any $i \neq j$ since the faulty S-box remains a permutation. Note that an attacker can also bypass this countermeasure with more values swapped, but targeting the case where only two values are swapped minimizes the effort for fault injection.

Let us take the 4-bit PRESENT's S-box (which is also used in the LED cipher) [BKL+07] as an example, as given in Table 1. There are 120 possible faulty S-boxes with two elements swapped that cannot be detected by this countermeasure.

4.2 Tissot *et al.*'s countermeasure (BALoo) [TBG23]

Description of countermeasure. The main idea is to detect biases in an S-box by exploiting the fact that an S-box can be decomposed into a product of disjoint cycles. Let $\mathbf{S}$ be an S-box of n elements, $\mathbf{S}[i] = a_i$ for $0 \leq i \leq n-1$. Two elements a_{i_1} and a_{i_2} are the two consecutive elements in a cycle if $a_{i_2} = \mathbf{S}[a_{i_1}]$, where $0 \leq i_1 \neq i_2 \leq n-1$. We say that $(a_{i_1}, a_{i_2}, \ldots, a_{i_\ell})$ is a cycle of length ℓ if $\mathbf{S}[a_{i_1}] = a_{i_2}, \ldots, \mathbf{S}[a_{i_\ell}] = a_{i_1}$, where $0 \leq i_1, i_2, \ldots, i_\ell \leq n-1$ and $i_1, i_2, \ldots, i_\ell$ are pairwise different.

An interesting fact observed by the authors of BALoo is that the number of cycles and their corresponding lengths are distinctive for a given permutation (S-box). The idea of the countermeasure is to verify the lengths of these cycles, given their starting indexes, before executing the cryptographic algorithm. BALoo ensures the detection of any bias in an S-box, as any bias will disrupt the distinctive set of cycles.

As an example, Figure 2 shows the four cycles of the PRESENT's S-box [BKL+07]. We denote these cycles as $(0, 12, 4, 9, 14, 1, 5)$, $(2, 6, 10, 15)$, $(3, 11, 8)$ and $(7, 13)$ corresponding to the lengths of $7, 4, 3$ and 2.

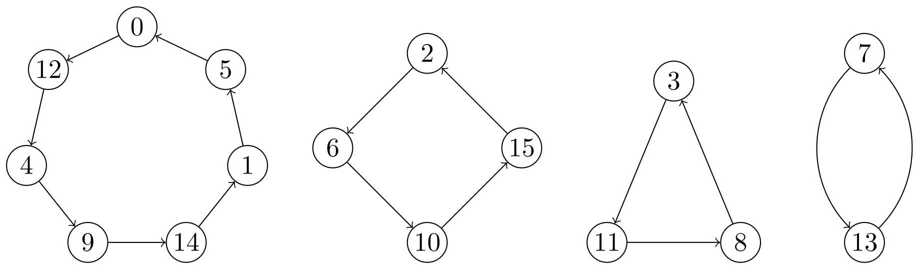

Figure 2: Cycles in BALoo of the PRESENT's S-box

Bypassing countermeasure. We notice that the core idea of BALoo is to verify the number of cycles and their corresponding lengths, given the starting indexes of each cycle. To bypass it, the faulty S-box must maintain the same cycle structure as the original. A possibility for an attacker to achieve this is to fault several values within a cycle such that some consecutive values swap their positions. The number of faulted values required depends on the length ℓ of the targeted cycle. We consider the following cases:

- $\ell > 3$: At least 3 consecutive values in a cycle should replace their positions with each other. To minimize the effort during the injection phase, we consider the case where only 3 consecutive values in a cycle swap their positions. Without loss of generality, let a_0, a_1, a_2, a_3 be consecutive values in a cycle of the original S-box $\mathbf{S}$, that is, $\mathbf{S}[a_0] = a_1, \mathbf{S}[a_1] = a_2, \mathbf{S}[a_2] = a_3$. If the faults change the values at positions indexed by a_0, a_1 and a_2 such that $\mathbf{S}'[a_0] = a_2, \mathbf{S}'[a_2] = a_1, \mathbf{S}'[a_1] = a_3$ (where $\mathbf{S}'$ is the faulty S-box), the attacker can then bypass the BALoo countermeasure with $\mathbf{S}'$.
- $\ell = 3$: There is only one possible faulty S-box of this case. Without loss of generality, let a_0, a_1, a_2 be the 3 consecutive values in a cycle, that is, $\mathbf{S}[a_0] = a_1, \mathbf{S}[a_1] = a_2, \mathbf{S}[a_2] = a_0$. The only faulty cycle is $\mathbf{S}'[a_0] = a_2, \mathbf{S}'[a_2] = a_1, \mathbf{S}'[a_1] = a_0$.
- $\ell < 3$: If faults cause any changes in the cycle of length 2 or 1, the countermeasure will detect the faults since the number of total cycles will increase for $\ell = 2$ and decrease for $\ell = 1$ (given fixed starting indexes). Hence, there is no possible faulty S-box of this case.

In short, an attacker need to swap at least 3 values in order to bypass BALoo. Let us take the S-box of the PRESENT cipher as an example. There are 11 possible faulty

S-boxes for $\ell > 3$ (the first two cycles in Figure 2) and 1 faulty S-box for $\ell = 3$, hence, 12 faulty S-boxes in total that cannot be detected by BALoo. A faulty S-box that an attacker can target, for instance, is:

$$(0, 12, 4, 9, 14, 1, 5), (2, 10, 15, 6), (3, 11, 8), (7, 13),$$

where the faults make 6 become 10, 10 become 15 and 15 become 6.

Another possibility for an attacker to bypass this countermeasure is to fault several values across different cycles. This requires swapping at least 4 values. For instance, a faulty S-box of the PRESENT cipher evading detection in this manner might be

$$(0, 12, 4, 9, 14, 6, 5), (2, 1, 10, 15), (3, 11, 8), (7, 13),$$

where the faults make 1 become 6, 6 become 1, 5 become 10 and 10 become 5. In this work, we aim to minimize the effort for fault injection, so we focus on the bypass where only 3 values are swapped.

5 Linear attack in persistent fault model

Recall an attacker can bypass the state-of-the-art countermeasures [TBG23, CB19] with two extensions in our model: a non-biased faulty S-box and a fault on another constant rather than S-box elements. We now show that, after bypassing these countermeasures with a non-biased faulty S-box, an attacker can still recover the key using a linear attack. We choose the PRESENT cipher to demonstrate the attack, which is also the target cipher for many PFAs in the literature [ZZJ+20, ZFL+22, PZRB19, CB19].

In this section, we briefly recall the PRESENT cipher and the principle of linear attack. We next show the faulty S-boxes that an attacker should target to achieve the best complexity for key recovery. We then present the steps of the linear attack on the cipher with faulty S-boxes, following the work of Flórez-Gutiérrez and Naya-Plasencia [FN20]. A toy example for the key recovery on a small version of PRESENT can be found in Appendix B.

5.1 Description of PRESENT

At CHES 2007, Bogdanov *et al.* proposed PRESENT as an ultra-lightweight block cipher [BKL+07]. The cipher has two versions, PRESENT-80 and PRESENT-128, corresponding to the master key of 80 bits and of 128 bits, respectively. It has a block size of 64 bits and consists of 31 rounds, plus the addition of whitening key at the output. As depicted in Figure 3, each round is the composition of three transformations:

– **addRoundKey**: A 64-bit round key is XORed bitwise to the current state.

– **sBoxLayer**: Each 4-bit nibble of the state is substituted using a fixed 4-bit S-box lookup table.

– **pLayer**: A fixed bitwise permutation is applied to the current state.

The key schedule uses the master key (80 or 128 bits) to generate the round keys. This is the only difference between the two PRESENT versions.

5.2 Estimation of attack complexity and success probability

Linear attack exploits the occurrences with high probability of linear expressions involving plaintext bits, ciphertext bits and subkey bits due to the weaknesses of an S-box. The core idea of our PFA is to weaken the S-box by faults (in addition to bypassing the

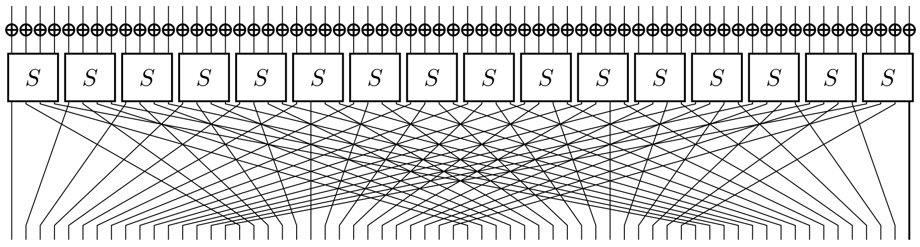

Figure 3: Graphic representation of one round of PRESENT [Jea16].

countermeasures) to facilitate the linear attack. Our goal now is to estimate the attack complexity and success probability to prove that an attacker can recover the key with a non-biased faulty S-box.

In this work, we use the *multiple linear attacks* proposed by Flórez-Gutiérrez and Naya-Plasencia [FN20]. The main idea of [FN20] is to combine multiple *1-bit linear approximations*. The background of the linear attack can be found in Appendix A, or in [FN20, BN16]. Here, we briefly recall estimation formulas proposed by Blondeau and Nyberg [BN16], which are also used by Flórez-Gutiérrez and Naya-Plasencia in their work [FN20].

We consider an attack on a cipher with an n-bit block size ($n = 64$ for PRESENT) using ℓ 1-bit linear approximations and N *distinct* known plaintext-ciphertext pairs (repeated plaintexts are not allowed, thus $N \leq 2^n$). The right-key statistic approximately follows a normal distribution $\mathcal{N}(\mu_R, \sigma_R)$, where

$$\begin{aligned}\mu_R &= B\ell + N \cdot \mathrm{Exp}_K(C(K)),\\ \sigma_R^2 &= 2B^2\ell + 4BN \cdot \mathrm{Exp}_K(C(K)) + N^2 \cdot \mathrm{Var}_K(C(K)),\end{aligned}$$

and

$$B = \frac{2^n - N}{2^n - 1}.$$

The notion of *capacity*, denoted $C(K)$, is to represent the combined information of ℓ linear approximations [BDQ04]. The calculation of $\mathrm{Exp}_K(C(K))$ and $\mathrm{Var}_K(C(K))$ can be found in Appendix A. The wrong-key statistic approximately follows a normal distribution $\mathcal{N}(\mu_W, \sigma_W)$, where

$$\begin{aligned}\mu_W &= B\ell + N\ell \cdot 2^{-n},\\ \sigma_W^2 &= 2\ell(B + N \cdot 2^{-n})^2.\end{aligned}$$

We denote k the part of the key that an attacker aims to recover and $|k|$ is its bit length. Instead of discovering the right key directly, many linear attacks aim at gaining advantage over the exhaustive search. To measure the effectiveness of linear attacks, Selçuk [Sel08] introduced the notion of *advantage*, denoted a. If the correct key guess of k is ranked as the i-th candidate among $2^{|k|}$ possibilities by a key-ranking statistic, we say the attack obtains $a = (|k| - \log i)$-bit advantage over the exhaustive search. The best case, where the correct key guess is the first candidate ($i = 1$ and thus $a = |k|$), corresponds to obtaining a $|k|$-bit advantage over a $|k|$-bit key.

As in [BN16], the success probability P_S of a multiple linear attack is computed as

$$P_S = \Phi\left(\frac{\mu_R - \mu_W - \sigma_W \cdot \Phi^{-1}(1 - 2^{-a})}{\sigma_R}\right) \tag{1}$$

where Φ is the cumulative distribution function and Φ^{-1} is the quantile function of the standard normal distribution. We use Equation 1 to visualize the dependence of the advantage a on the number of plaintext-ciphertext pairs N, given a value of success probability P_S.

5.3 Targeted faulty S-boxes

Since the success probabilities and data complexities of the attacks with different faulty S-boxes are not the same, we aim to determine the faulty S-box that offers the best success probability and data complexity for an attacker exploiting persistent faults.

To achieve this, we use the notion of *Estimated Linear Potential*, denoted ELP. Intuitively, ELP quantifies the sum of correlation contributions of all 1-bit linear approximation paths from an input bit to an output bit. Roughly speaking, ELP measures how biased the linear approximation between this input bit and output bit is, and thus how easy it is to exploit this approximation. The ELP value can be computed from a (non-biased faulty) S-box. A higher ELP value indicates lower attack complexity and is thus better for an attacker. A formal definition and computation method for ELP can be found in Appendix A.

Recall that an attacker can bypass the countermeasure of Caforio and Banik [CB19] by swapping 2 values and the countermeasure of Tissot *et al.* [TBG23] by swapping 3 values in the S-box. We thus consider only the faulty S-boxes from these two cases ($\binom{16}{2}$ with two values swapped and $\binom{16}{3}$ with 3 values swapped) and select the best one for each case. The details of the computation for this selection are provided in in Appendix A. Table 2 presents the best faulty S-boxes with 2 values swapped ($\mathbf{S}'$) and with 3 values swapped ($\mathbf{S}''$), along with their ELP values. For comparison, we also include the original S-box ($\mathbf{S}$). We observe that the ELP values of $\mathbf{S}'$ and $\mathbf{S}''$ are significantly higher than that of the original S-box, which is expected to substantially decrease the attack complexity.

Table 2: Target faulty S-boxes with two swaps ($\mathbf{S}'$) and three swaps ($\mathbf{S}''$)

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	$\log_2$(ELP)
$\mathbf{S}[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2	-74.0
$\mathbf{S}'[x]$	C	5	6	B	9	0	A	3	D	E	F	8	4	7	1	2	-41.4
$\mathbf{S}''[x]$	C	5	8	B	9	0	A	D	3	6	F	E	4	7	1	2	-33.7

5.4 Attack on full-round PRESENT

We consider PRESENT-80 for our analysis. Following the multiple linear attack of Florez-Gutierrez and Naya-Plasencia [FN20], we choose $\ell = 128$ linear approximations and set the success probability $P_S = 0.95$. By Equation 1, the relationship between the advantage a and the number of distinct plaintext-ciphertext pairs N is depicted in Figure 4.

We can observe that the attack using the faulty S-boxes $\mathbf{S}'$ and $\mathbf{S}''$ requires significantly fewer data compared to the attack on the original S-box $\mathbf{S}$. For instance, to achieve an advantage of $a = 10$, the attack on the cipher with $\mathbf{S}'$ and $\mathbf{S}''$ need $2^{44.0}$ and $2^{35.1}$ plaintext-ciphertext pairs. In contrast, the attack on the cipher with $\mathbf{S}$ requires $2^{74.0}$ pairs, which is practically infeasible given that there are at most 2^{64} distinct plaintexts available for PRESENT (since the block size is 64 bits).

We briefly recall the steps of the attack algorithm and compute its complexity in terms of time and memory. In this attack, we make guess for parts of the first two and the last two round keys associated with the chosen approximations. For further details of the algorithm and complexity analysis, we refer the readers to the original paper by Flórez-Gutiérrez and Naya-Plasencia [FN20]. Given N pairs of data and ℓ approximations, an attacker performs the following computations:

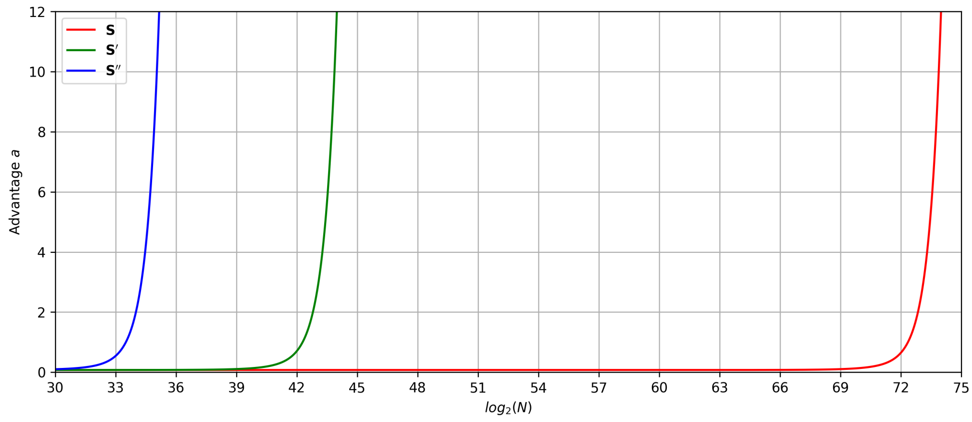

Figure 4: Advantage for attack on full-round PRESENT with different S-boxes. The success probability is fixed to $P_S = 0.95$. The number of linear approximations is $\ell = 128$.

1. For each approximation indexed by i, where $i \in [1, \ell]$, construct a matrix A_i that stores the count of each distinct associated value in the approximation extracted from the ciphertext corresponding to each value extracted from the plaintext.
2. For each matrix A_i, apply the Fast Walsh-Hadamard Transform (FWHT) to every row and column.
3. For each approximation indexed by i, where $i \in [1, \ell]$:
   (a) Compute the first (resp. second) eigenvalue vector based on the guessed first (resp. last) two round subkeys.
   (b) Compute the experimental correlations q_i from A_i and the two eigenvalue vectors.
4. For each guess $\hat{k}$ of the subkey k, compute the ranking value $Q_{\hat{k}}$ from $\hat{k}$ and all experimental correlations q_i, where $i \in [1, \ell]$. The correct guess corresponds to the largest value of Q.

Table 3 presents a comparison of linear attacks on the faulty S-boxes versus attack on the original S-box. In the last column, we include the estimated time required to collect the necessary data, assuming a device operating at 100 MHz and requiring 496 cycles per encryption.[1] We see that the data collection times for our attacks are much more realistic (2.8 years corresponds to $\mathbf{S}'$ and 2.1 days corresponds to $\mathbf{S}''$) than those of [FN20]. For the complexity of time and memory, we note that each step of the algorithm can be parallelized and realized independently. By this way, we can take advantage of hardware to accelerate the key recovery. The key focus of our analysis is on providing a realistic time required to collect data for the analysis.

For further comparison, our approximation corresponds to an average case of the two fast implementations for PRESENT by Reis *et al.* [RAL17] on ARM cortex-M3 and the ARM Cortex-A53 processors. These implementations respectively require 2116 cycles and 1052 cycles at clock speeds of 144 MHz and 2 GHz. Thus, for the Cortex-A53 (resp. Cortex-M3), we estimately need 107 days to collect data for $\mathbf{S}'$ (resp. 8.3 years) and 5 hours to collect data for $\mathbf{S}''$ (resp. 6.3 days).

[1]We assume that the number of clock cycles for an encryption is dominated by S-box lookup operations. Each of these operations costs 1 cycle. PRESENT-80 consists of 31 rounds, each has 16 S-box lookup operations. Thus, $31 \times 16 = 496$

Table 3: Comparison of linear attacks with faulty S-boxes. Note that the attack of [FN20] is on reduced-round cipher and our attack is on full-round cipher. The data collection time is estimated on a 100 MHz device with the assumption that an S-box lookup operation takes 1 cycle, thus $31 \times 16 = 496$ cycles per encryption.

Source	S-box	P_S	#Rounds	#Approx.	Time	Memory	Capacity	Data	Collect. Time (est.)
[FN20]	$\mathbf{S}$	0.95	27	128	2^{72}	2^{44}	$2^{-54.8}$	$2^{63.4}$	$2^{20.8}$ years
This work	$\mathbf{S}'$	0.95	31	128	2^{70}	2^{44}	$2^{-37.2}$	$2^{44.0}$	2.8 years
This work	$\mathbf{S}''$	0.95	31	128	2^{70}	2^{44}	$2^{-28.4}$	$2^{35.1}$	2.1 days

We provide in Appendix B a toy example for this attack on a small version of PRESENT to show that our estimation is correct.

5.5 Discussion

Compared to previous PFAs (*e.g.*, [ZLZ+18, ZZJ+20, SBH+22]), which typically require hundreds to thousands of ciphertexts, our attack demands a quite large number of plaintext-ciphertext pairs. As a result, our approach may appear less effective for key recovery in terms of time and data complexity. However, it is important to emphasize that our primary goal is to demonstrate a potential risk: an attacker could still recover the key even when the state-of-the-art countermeasures [TBG23, CB19] are in place. This finding underscores that these countermeasures might not be entirely sufficient to prevent persistent fault attacks.

The core idea of exploiting S-box weaknesses is also fundamental to other attacks, *e.g.*, differential cryptanalysis. In this work, linear attack is a specific method we use to illustrate the effectiveness of PFA within the extended model involving a non-biased faulty S-box. Future research could explore various other attacks in the context of persistent faults, including differential attacks and more advanced forms of linear attacks, as potential venues for further investigation.

6 Differential fault attack in persistent fault model

Recall an attacker can bypass state-of-the-art countermeasures [TBG23, CB19] using two extensions in our model: causing a non-biased faulty S-box and causing faults in constants other than S-box elements. In Section 5, we demonstrated how an attacker can still recover the key using a linear attack after bypassing the countermeasures with a non-biased faulty S-box. In this section, we investigate the latter extension. Specifically, we show that by faulting the 8-th round constant in the AES-128 cipher, an attacker can recover the key using a differential fault analysis (DFA).

We note that the effect of this fault attack is similar to Kim's work [Kim12], which presented a DFA with a fault in the AES key schedule. The main difference is that Kim considered a *transient fault* induced into the first column of the 8-th round in the key schedule, whereas we consider a *persistent fault* induced into the 8-th round constant (assuming that round constants are stored as a lookup table in memory). Our analysis is somewhat simpler since the fault value remains the same across encryptions due to the nature of a persistent fault. Nonetheless, we can directly apply the Kim's analysis [Kim12] to recover the last round key. Therefore, we refer to the original analysis of Kim [Kim12] for the analysis. In this section, we only present the effect of a persistent fault on the 8-th round constant, which is the main difference from [Kim12].

6.1 Description of AES

AES [DR05] is a block cipher with block size of 128 bits. A block (also called a *state*) is an array of 4×4 bytes indexed from 0 to 15. In our attack, we consider the 128-bit key variant. It consists of 10 rounds where each round is the composition of the following transformations:

- **SubBytes** (SB): A non-linear substitution is applied to each of the 16 bytes of the state.
- **ShiftRows** (SR): A circular shift is applied on each row of the state.
- **MixColumns** (MC): A linear bijection is applied on each column of the state.
- **AddRoundKey** (AK): A 16-byte round key is XOR-ed to the state.

6.2 Fault propagation on AES-128

We consider a persistent fault induced into the 8-th round constant in the key schedule. The fault value does not need to be known to the attacker. Suppose that the attacker obtains N pairs of correct-faulty ciphertexts encrypted with the same plaintexts. We now describe how the differential effects of the fault propagate through the last three rounds of AES encryption.

Let a denote the difference at the output of the associated XOR operation caused by the fault. Figure 5 illustrates the propagation of a through the 8-th, 9-th and 10-th round keys. Due to the SB transformation, this difference a results in two additional differential values, denoted b and c, in the 9-th and 10-th round keys. Since the fault is persistent, the values a, b and c remain the same across all N correct-faulty ciphertext pairs.

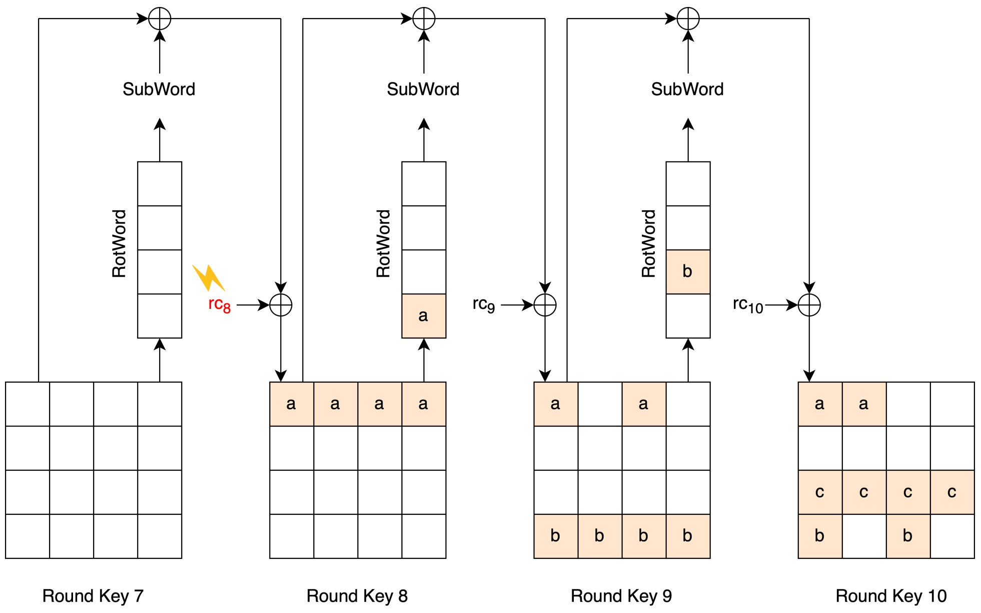

Figure 5: Differential propagation in the key schedule. This figure omits the XOR operations to recursively generate the 2-nd, 3-rd and 4-th columns of each round key for a succinct illustration.

Figure 6 depicts the influence of the differences a, b and c in the last three rounds of AES encryption. First, a spreads to the first row of the 8-th round key, and thus the output state of the AK in the 8-th round (state (5) in Figure 6). Next, the SB transformation causes the changes on the differences of the first row from (a, a, a, a) to (e, f, g, h) (state (6) in Figure 6). Since SB is a non-linear transformation, the differences e, f, g, and h are plaintext-dependent and vary among different correct-faulty ciphertext pairs. Then, the MC transformation spreads them to the four cells of its corresponding column. Now, there is a differential relation of the four cells in each column (state (8) in Figure 6), *e.g.*, $(2e, e, e, 3e)$ for the first column. The AK of the 9-th round key then adds a to two cells of the first row and b to four cells of the last row (state (9) in Figure 6).

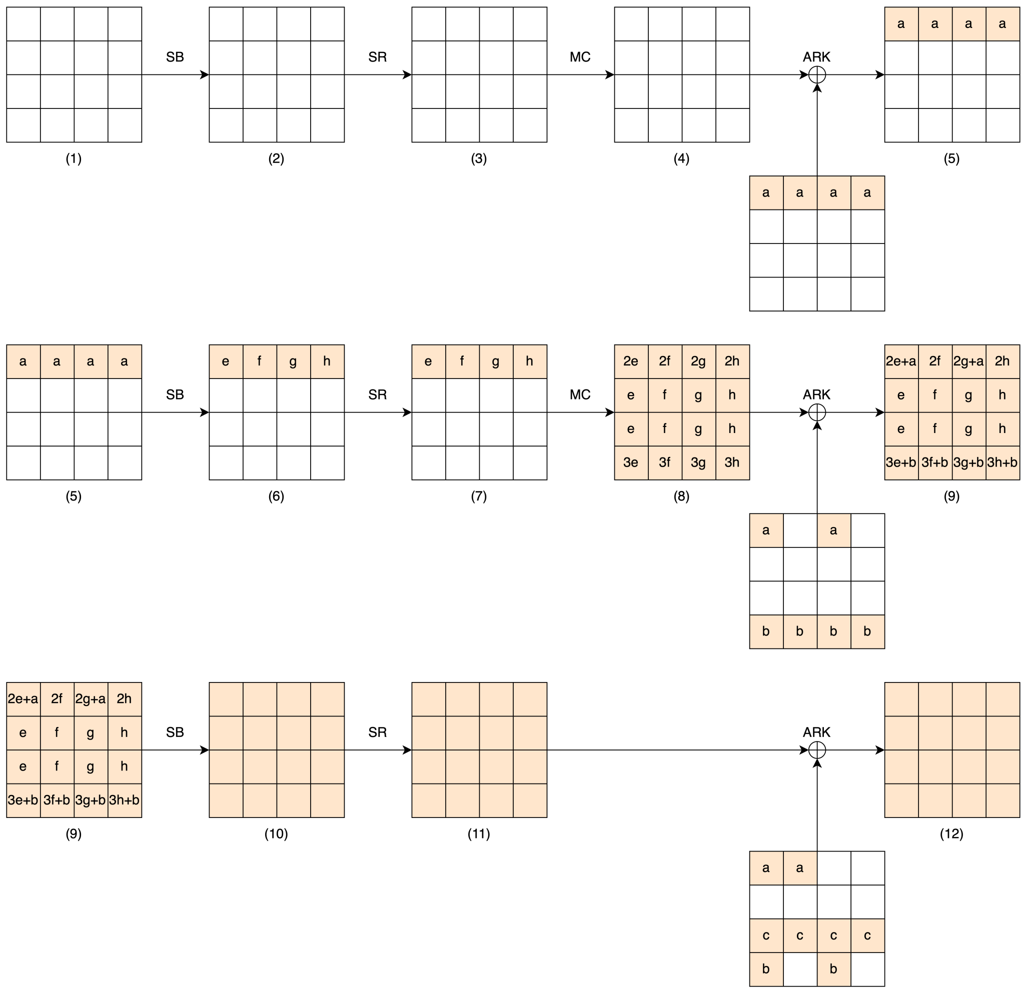

Figure 6: Differential propagation in the last three rounds of AES.

Given a pair of correct-faulty ciphertexts, we make hypotheses for bytes of the last round key and for the differences a, b, and c (we only need to make hypotheses for 2 bytes at a time, see [Kim12]). We then compute backwards to the state at the beginning of the last round (state (9) in Figure 6). The correct hypothesis will lead to a match for the differential relation in this state. For the algorithm of the analysis, we refer to the paper of Kim [Kim12].

According to the analysis by Kim [Kim12], $N = 2$ pairs of correct-faulty ciphertexts can reduce the search space of the key to 2^{24} candidates. However, Kim [Kim12] considered a transient fault where a, b and c are different for each pair of ciphertexts. In contrast, we can take advantage of a persistent fault where the differences a, b and c remain the same for N pairs. Our simulation shows that with $N = 3$ pairs, we can obtain a single candidate that turns out to be the correct key, whereas with $N = 2$ pairs, we obtain around 20 candidates.

6.3 Discussion

By this analysis, we reduce the PFAs based on statistical analyses from previous works (*e.g.*, [ZLZ+18, ZZJ+20, SBH+22]), which require hundreds to thousands ciphertexts, to a DFA needing only 2 pairs of correct-faulty ciphertexts. This approach is much more effective for the key recovery in terms of time and data complexity. However, DFA does not work in the ciphertexts-only context as the previous analyses. Our attack works under an assumption that an attacker can encrypt a plaintext twice, once with the fault present and once without, to collect a pair of correct-faulty ciphertexts. In practice, an attacker can collect a set of correct ciphertexts before injecting faults. In fact, this assumption was also used in the work of Caforio and Banik [CB19].

Another advantage of our analysis is that the assumption from previous works, that the key schedule is not affected by the faults, can be relaxed. Recall that the PFA of Zhang *et al.* [ZLZ+18] and many follow-up works [ZZJ+20, SBH+22, PZRB19, CGR20, XZY+21, GPT19, TL22] assumed that round keys are pre-computed so that the faulty S-boxes do not affect the key schedule. This assumption ensures that the recovered last round key is correct and allows for easy reversal of the master key. Nevertheless, this is not always true in practice. For instance, each round key is usually computed on-the-fly in smartcard implementations. Recently, Zhang *et al.* [ZHF+23] attempted to address this issue by reducing the key search space using a chosen-plaintext based PFA. Our DFA, which relies on a fault in a round constant instead of S-box elements, is not susceptible to this limitation.

7 Robust countermeasure

In Section 4, we demonstrated that the state-of-the-art countermeasures [TBG23, CB19] can be bypassed by exploiting non-biased faulty S-boxes and faults in other constants (*e.g.*, a round constant in AES). This vulnerability allows for key recovery, as detailed in Section 5 and Section 6. The shortcomings of these countermeasures arise from their exclusive focus on detecting biases in the S-box alone. In this section, we present a countermeasure that addresses this limitation by offering a more comprehensive approach to fault detection.

The goal of our countermeasure is to detect any persistent faults in the S-box, any other constants, and any unforeseen locations that result in faulty ciphertexts. That means the correctness of the encryption functionality must be ensured. The core idea of our countermeasure is to use several correct plaintext-ciphertext pairs (called the *references*) to verify the correctness of the encryption. These reference pairs could be produced and hardcoded in memory by the manufacturer. We note that this idea was briefly mentioned by Zhang *et al.* [ZZJ+20] in their discussion. Here, we work out the idea in detail to show how we can apply this to different ciphers.

Let $(m_1, c_1), \ldots, (m_\ell, c_\ell)$ be these ℓ reference pairs, where $m_1, \ldots, m_\ell$ are random plaintexts. The number of pairs ℓ should be minimized to reduce memory cost, but it must be sufficient to ensure that the encryptions of these plaintexts access all S-box elements, allowing us to detect any fault in the S-box. Hence, the number of pairs may differ for different ciphers. For instance, a single encryption of a random plaintext in PRESENT-80 is likely to access all 16 S-box elements with high probability since the S-box operation is used 527 times during the encryption (31×16 times in the main algorithm and 31 times in the key schedule). However, this is not the case for AES-128, as its encryption accesses the S-box 200 times, while it has 256 S-box elements.

We now formally calculate the value of ℓ for an arbitrary cipher with a random fixed key. Let n be the number of elements in an S-box, s_e and s_k be the number of times the S-box operation is used during a single encryption and a single key schedule, respectively. In ℓ encryptions, there are $s_e \cdot \ell + s_k$ S-box operations in total. The probability that there exists an S-box element which is never accessed in all $s_e \cdot \ell + s_k$ S-box operations is

$$\left(\frac{n-1}{n}\right)^{s_e \cdot \ell + s_k}.$$

The probability p that every S-box element is accessed at least once in $s_e \cdot \ell + s_k$ S-box operations is therefore

$$p = 1 - \left(\frac{n-1}{n}\right)^{s_e \cdot \ell + s_k}.$$

Figure 7 shows the probability p corresponding to different numbers of encryptions ℓ for PRESENT-80 ($s_e = 496, s_k = 31, n = 16$) and AES-128 ($s_e = 160, s_k = 40, n = 256$). For PRESENT, $\ell = 1$ encryption is sufficient to ensure that all S-box elements are accessed at least once with a probability $p \approx 1$. For AES, the required number of encryptions ℓ is higher to achieve a probability close to 1. In particular, $p \approx 0.98$ when $\ell = 6$ and $p \approx 1.00$ when $\ell = 9$.

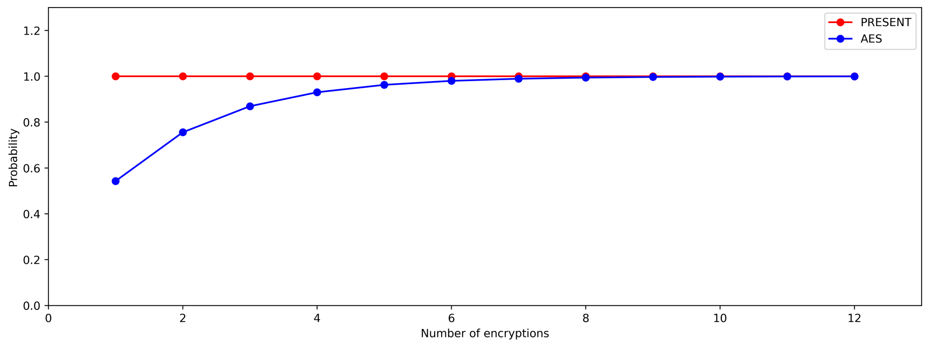

Figure 7: Probability p of different number of encryptions ℓ for PRESENT and AES

For AES, storing 9 reference pairs (288 bytes) in memory is costly. This cost can be reduced by using each ciphertext as the plaintext for the next encryption, *i.e.*, $m_2 = c_1, m_3 = c_2, \ldots, m_9 = c_8$. In other words, we can reduce the storage requirement by 128 bytes, needing to store only 160 bytes for $m_1, c_1, c_2, \ldots, c_9$. This reduction means we need only 62.5% of the space typically required for the S-box.

The next question is when to apply this countermeasure. In [TBG23], Tissot *et al.* proposed activating their verification once every certain number of encryptions to minimize time performance loss. This approach is based on the observation that statistical PFAs require the collection of hundreds or thousands of ciphertexts. They apply the countermeasure occasionally to prevent an attacker from collecting enough ciphertexts for analysis. However, an attacker can still collect few faulty ciphertexts before the faults are detected. In [CB19], Caforio and Banik sacrificed the fault detection performance to improve chip area performance in their ASIC implementation. This trade-off also allows an attacker to collect few faulty ciphertexts before the faults are detected. In Section 6, we have shown that an attacker can use DFA to recover the key with only 2 correct-faulty ciphertexts pairs, indicating that the countermeasures in [TBG23, CB19] do not ensure the security.

For those reasons, we propose applying the verification of our countermeasure immediately after the device reboots. If a persistent fault is detected (*i.e.*, if the ciphertext returned by an encryption does not match its reference), the device should either reboot again to erase the fault or return random outputs thereafter.

Table 4: Comparison of our countermeasure with the existing ones.

	[TBG23]	[CB19]	Ours
Biased S-box	✓	✓	✓
S-box with 2 elements swapped (linear attack)	✓	✗	✓
S-box with 3 elements swapped (linear attack)	✗	✗	✓
Fault on round constant (differential attack)	✗	✗	✓

Table 4 compares the fault detection capabilities of our countermeasure with those presented in [TBG23, CB19]. As shown, our countermeasure is able detect persistent faults in many cases that the others cannot.

8 Conclusion and perspectives

In this paper, we proposed two extensions to the previous persistent fault model: a non-biased faulty S-box and a fault affecting constants other than S-box elements. We demonstrated that state-of-the-art countermeasures can be bypassed using these extensions. Additionally, we showed that an attacker can still recover the key through either a linear attack or a differential attack even after bypassing the countermeasures. Furthermore, we reduced the traditional statistical PFAs, which demand hundreds to thousands of ciphertexts, to a DFA that only needs 2 correct-faulty ciphertext pairs. Finally, we presented a more robust countermeasure and showed that it is more efficient than the existing ones.

By this work, we aim to highlight the potential risks associated with the existing countermeasures that focus solely on detecting biases in the S-boxes. We also hope that our extended model can open up new research directions in persistent fault attacks from both hardware and cryptographic perspectives.

Still, there are further works that need to be fulfilled:

– Conduct experiments on hardware to assess the difficulty of obtaining a non-biased faulty S-box and faulting a round constant.

– Develop analyses that exploit a non-biased faulty S-box more effectively than the linear attack presented in Section 5. In our extended model, we assume that an attacker can encrypt a plaintext twice to obtain a correct-faulty ciphertext pair. This assumption could open up more efficient (differential) analyses.

– Implement the proposed countermeasure on hardware to verify its efficiency.

References

[ABF+03] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and Jean-Pierre Seifert. Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 260–275. Springer, Heidelberg, August 2003. doi:10.1007/3-540-36400-5_20.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 37–51. Springer, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_4.

[BDQ04] Alex Biryukov, Christophe De Cannière, and Michaël Quisquater. On multiple linear approximations. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 1–22. Springer, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_1.

[BKL$^+$07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, Heidelberg, September 2007. doi:10.1007/978-3-540-74735-2_31.

[BN16] Céline Blondeau and Kaisa Nyberg. Improved parameter estimates for correlation and capacity deviates in linear cryptanalysis. *IACR Trans. Symm. Cryptol.*, 2016(2):162–191, 2016. https://tosc.iacr.org/index.php/ToSC/article/view/570. doi:10.13154/tosc.v2016.i2.162-191.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Heidelberg, August 1997. doi:10.1007/BFb0052259.

[BTV18] Andrey Bogdanov, Elmar Tischhauser, and Philip S. Vejre. Multivariate profiling of hulls for linear cryptanalysis. *IACR Trans. Symm. Cryptol.*, 2018(1):101–125, 2018. doi:10.13154/tosc.v2018.i1.101-125.

[CB19] Andrea Caforio and Subhadeep Banik. A study of persistent fault analysis. In Shivam Bhasin, Avi Mendelson, and Mridul Nandi, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 13–33, Cham, 2019. Springer International Publishing.

[CGR20] Sébastien Carré, Sylvain Guilley, and Olivier Rioul. Persistent fault analysis with few encryptions. In Guido Marco Bertoni and Francesco Regazzoni, editors, *COSADE 2020*, volume 12244 of *LNCS*, pages 3–24. Springer, Heidelberg, April 2020. doi:10.1007/978-3-030-68773-1_1.

[Cho10] Joo Yeon Cho. Linear cryptanalysis of reduced-round PRESENT. In Josef Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *LNCS*, pages 302–317. Springer, Heidelberg, March 2010. doi:10.1007/978-3-642-11925-5_21.

[Cla07] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 181–194. Springer, Heidelberg, September 2007. doi:10.1007/978-3-540-74735-2_13.

[DEK$^+$18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: Exploiting ineffective fault inductions on symmetric cryptography. *IACR TCHES*, 2018(3):547–572, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7286. doi:10.13154/tches.v2018.i3.547-572.

[DLV03] Pierre Dusart, Gilles Letourneux, and Olivier Vivolo. Differential fault analysis on A.E.S. In Jianying Zhou, Moti Yung, and Yongfei Han, editors, *Applied Cryptography and Network Security, First International Conference, ACNS 2003. Kunming, China, October 16-19, 2003, Proceedings*, volume 2846 of *Lecture Notes in Computer Science*, pages 293–306. Springer, 2003. doi:10.1007/978-3-540-45203-4_23.

[DR05] Joan Daemen and Vincent Rijmen. Rijndael/aes. In Henk C. A. van Tilborg, editor, *Encyclopedia of Cryptography and Security*. Springer, 2005. doi:10.1007/0-387-23483-7_358.

[ESP20] Susanne Engels, Falk Schellenberg, and Christof Paar. SPFA: SFA on multiple persistent faults. In *17th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2020, Milan, Italy, September 13, 2020*, pages 49–56. IEEE, 2020. doi:10.1109/FDTC51366.2020.00014.

[FJLT13] Thomas Fuhr, Eliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on AES with faulty ciphertexts only. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 108–118, 2013. doi:10.1109/FDTC.2013.18.

[FN20] Antonio Flórez-Gutiérrez and María Naya-Plasencia. Improving key-recovery in linear attacks: Application to 28-round PRESENT. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 221–249. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_9.

[GPT19] Michael Gruber, Matthias Probst, and Michael Tempelmeier. Persistent fault analysis of OCB, DEOXYS and COLM. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2019, Atlanta, GA, USA, August 24, 2019*, pages 17–24. IEEE, 2019. doi:10.1109/FDTC.2019.00011.

[GYTS14] Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont. Differential fault intensity analysis. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 49–58, 2014. doi:10.1109/FDTC.2014.15.

[Jea16] Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.

[KDK$^+$14] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 361–372, 2014. doi:10.1109/ISCA.2014.6853210.

[Kim12] Chong Hee Kim. Improved differential fault analysis on aes key schedule. *IEEE Transactions on Information Forensics and Security*, 7(1):41–50, 2012. doi:10.1109/TIFS.2011.2161289.

[KQ08] Chong Hee Kim and Jean-Jacques Quisquater. New differential fault analysis on AES key schedule: Two faults are enough. In Gilles Grimaud and François-Xavier Standaert, editors, *Smart Card Research and Advanced Applications, 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008, London, UK, September 8-11, 2008. Proceedings*, volume 5189 of *Lecture Notes in Computer Science*, pages 48–60. Springer, 2008. doi:10.1007/978-3-540-85893-5_4.

[LSG$^+$10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 320–334. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-15031-9_22.

[Mat94] Mitsuru Matsui. The first experimental cryptanalysis of the data encryption standard. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 1–11. Springer, Heidelberg, August 1994. doi:10.1007/3-540-48658-5_1.

[NSZW09] Jorge Nakahara Jr., Pouyan Sepehrdad, Bingsheng Zhang, and Meiqin Wang. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 58–75. Springer, Heidelberg, December 2009.

[Nyb95] Kaisa Nyberg. Linear approximation of block ciphers (rump session). In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 439–444. Springer, Heidelberg, May 1995. doi:10.1007/BFb0053460.

[Ohk09] Kenji Ohkuma. Weak keys of reduced-round PRESENT for linear cryptanalysis. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *SAC 2009*, volume 5867 of *LNCS*, pages 249–265. Springer, Heidelberg, August 2009. doi:10.1007/978-3-642-05445-7_16.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 77–88. Springer, Heidelberg, September 2003. doi:10.1007/978-3-540-45238-6_7.

[PZRB19] Jingyu Pan, Fan Zhang, Kui Ren, and Shivam Bhasin. One fault is all it needs: Breaking higher-order masking with persistent fault analysis. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, 2019. doi:10.23919/DATE.2019.8715260.

[RAL17] Tiago B. S. Reis, Diego F. Aranha, and Julio Cesar López-Hernández. PRESENT runs fast - efficient and secure implementation in software. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 644–664. Springer, Heidelberg, September 2017. doi:10.1007/978-3-319-66787-4_31.

[Riv09] Matthieu Rivain. Differential fault analysis on DES middle rounds. In Christophe Clavier and Kris Gaj, editors, *CHES 2009*, volume 5747 of *LNCS*, pages 457–469. Springer, Heidelberg, September 2009. doi:10.1007/978-3-642-04138-9_32.

[SBH$^+$22] Hadi Soleimany, Nasour Bagheri, Hosein Hadipour, Prasanna Ravi, Shivam Bhasin, and Sara Mansouri. Practical multiple persistent faults analysis. *IACR TCHES*, 2022(1):367–390, 2022. doi:10.46586/tches.v2022.i1.367-390.

[SBHS15] Bodo Selmke, Stefan Brummer, Johann Heyszl, and Georg Sigl. Precise laser fault injections into 90 nm and 45 nm SRAM-cells. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 193–205. Springer, 2015. doi:10.1007/978-3-319-31271-2_12.

[SBR$^+$20] Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay. Fault template attacks on block ciphers exploiting fault propagation. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 612–643. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_22.

[Sel08] Ali Aydin Selçuk. On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, January 2008. doi:10.1007/s00145-007-9013-7.

[SHP09] Jörn-Marc Schmidt, Michael Hutter, and Thomas Plos. Optical fault attacks on AES: A threat in violet. In *2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 13–22, 2009. doi:10.1109/FDTC.2009.37.

[TBG23] Pierre-Antoine Tissot, Lilian Bossuet, and Vincent Grosso. BALoo: First and efficient countermeasure dedicated to persistent fault attacks. In Alessandro Savino, Michail Maniatakos, Stefano Di Carlo, and Dimitris Gizopoulos, editors, *29th International Symposium on On-Line Testing and Robust System Design, IOLTS 2023, Crete, Greece, July 3-5, 2023*, pages 1–7. IEEE, 2023. doi:10.1109/IOLTS59296.2023.10224871.

[TL22] Honghui Tang and Qiang Liu. MPFA: An efficient multiple faults-based persistent fault analysis method for low-cost FIA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(9):2821–2834, 2022. doi:10.1109/TCAD.2021.3117512.

[TMA11] Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. Differential fault analysis of the advanced encryption standard using a single fault. In Claudio A. Ardagna and Jianying Zhou, editors, *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication - 5th IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings*, volume 6633 of *Lecture Notes in Computer Science*, pages 224–233. Springer, 2011. doi:10.1007/978-3-642-21040-2_15.

[WW10] Gaoli Wang and Shaohui Wang. Differential fault analysis on PRESENT key schedule. In Muren Liu, Yuping Wang, and Ping Guo, editors, *2010 International Conference on Computational Intelligence and Security, CIS 2010, Nanning, Guangxi Zhuang Autonomous Region, China, December 11-14, 2010*, pages 362–366. IEEE Computer Society, 2010. doi:10.1109/CIS.2010.84.

[XZY+21] Guorui Xu, Fan Zhang, Bolin Yang, Xinjie Zhao, Wei He, and Kui Ren. Pushing the limit of PFA: Enhanced persistent fault analysis on block ciphers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(6):1102–1116, 2021. doi:10.1109/TCAD.2020.3048280.

[YMH03] Sung-Ming Yen, Sang-Jae Moon, and JaeCheol Ha. Permanent fault attack on the parameters of RSA with CRT. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *ACISP 03*, volume 2727 of *LNCS*, pages 285–296. Springer, Heidelberg, July 2003. doi:10.1007/3-540-45067-X_25.

[ZFL+22] Fan Zhang, Tianxiang Feng, Zhiqi Li, Kui Ren, and Xinjie Zhao. Free fault leakages for deep exploitation: Algebraic persistent fault analysis on lightweight block ciphers. *IACR TCHES*, 2022(2):289–311, 2022.

[ZHF+23] Fan Zhang, Run Huang, Tianxiang Feng, Xue Gong, Yulong Tao, Kui Ren, Xinjie Zhao, and Shize Guo. Efficient persistent fault analysis with small number of chosen plaintexts. *IACR TCHES*, 2023(2):519–542, 2023. doi:10.46586/tches.v2023.i2.519-542.

[ZLZ+18] Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren. Persistent fault analysis on block ciphers. *IACR TCHES*, 2018(3):150–172, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7272. doi:10.13154/tches.v2018.i3.150-172.

[ZLZ+21] Shihui Zheng, Xudong Liu, Shoujin Zang, Yihao Deng, Dongqi Huang, and Changhai Ou. A persistent fault-based collision analysis against the advanced encryption standard. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(6):1117–1129, 2021. doi:10.1109/TCAD.2021.3049687.

[ZZJ+20] Fan Zhang, Yiran Zhang, Huilong Jiang, Xiang Zhu, Shivam Bhasin, Xinjie Zhao, Zhe Liu(0), Dawu Gu, and Kui Ren. Persistent fault attack in practice. *IACR TCHES*, 2020(2):172–195, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8548. doi:10.13154/tches.v2020.i2.172-195.

A Background on linear attack

Linear approximation. We denote E the block cipher of r rounds with block size of n bits and expanded key size of $n(r+1)$ bits. Given a fixed expanded key $K \in \mathbb{F}_2^{n(r+1)}$ and a plaintext $x \in \mathbb{F}_2^n$, we denote $y = E_K(x) = E(x, K)$ the ciphertext of x. We also have $x = E_K^{-1}(y) = E^{-1}(y, K)$. A linear attack can be performed when a statistical property is observed from a part E'_K of the cipher. We rewrite the encryption function as $E_K(x) = (H_K \circ E'_K \circ G_K)(x)$ where G_K and H_K are the first and the last rounds, respectively. E'_K is composed of $r' < r$ iterations of the round function.

In this work, we only consider the classical linear cryptanalysis as proposed by Matsui in [Mat94]. Let $u, v \in \mathbb{F}_2^n$ be the input mask and output mask, the Boolean function $u \cdot x \oplus v \cdot E'_K(x)$ is called a *linear approximation* over E'_K. For a vectorial Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$, the correlation of the linear approximation $u \cdot x \oplus v \cdot f(x)$ is the following quantity:

$$\begin{aligned}&\mathrm{cor}(u \cdot x \oplus v \cdot f(x)) = \\ &\frac{1}{2^n}\left(\#\{x \in \mathbb{F}_2^n | u \cdot x \oplus v \cdot f(x) = 0\} - \#\{x \in \mathbb{F}_2^n | u \cdot x \oplus v \cdot f(x) = 1\}\right).\end{aligned} \tag{2}$$

For the sake of brevity, we denote $\mathrm{cor}(u, v)(K)$ as the correlation of the linear approximation over E'_K, we have:

$$\mathrm{cor}(u, v)(K) = \mathrm{cor}(u \cdot x \oplus v \cdot E'_K(x)). \tag{3}$$

A path of linear approximations which concatenates over multiple rounds of E'_K is called a *linear trail*. Let γ be a tuple of r' Boolean vectors, $\gamma = (\gamma_0, \ldots, \gamma_{r'})$, where $\gamma_i \in \mathbb{F}_2^n$ for $i \in \{0, \ldots, r'\}$, $\gamma_0 = u$ and $\gamma_{r'} = v$. A sequence of γ defines a linear characteristic, or a linear trail, of the linear approximation (u, v) and the correlation $\rho(u, v, \gamma)$ of the characteristic γ is computed by the following equation:

$$\rho(u, v, \gamma) = \prod_{i=1}^{r'} \mathrm{cor}(\gamma_{i-1} \cdot z_{i-1} + \gamma_i \cdot g(z_{i-1})), \tag{4}$$

where $g(\cdot)$ is the round function and z_{i-1} is the state input of the i-th round function. Then, the correlation over r' rounds of E'_K with the input mask u and output mask v can be written as:

$$\mathrm{cor}(u, v)(K) = \sum_{\substack{\gamma \\ \gamma_0 = u, \gamma_{r'} = v}} (-1)^{\gamma \cdot K} \rho(u, v, \gamma). \tag{5}$$

For the input mask u and output mask v, the set of all the linear trails γ contributing to the correlation $\mathrm{cor}(u, v)(K)$ is called the *linear hull* of (u, v), as introduced by Nyberg in [Nyb95]. The *Estimated Linear Potential* (ELP) of this linear hull is defined as:

$$\mathrm{ELP}(u, v) = \sum_{\substack{\gamma \\ \gamma_0 = u, \gamma_{r'} = v}} \rho(u, v, \gamma)^2. \tag{6}$$

Multiple linear attack. To represent the combined information of ℓ linear approximations, the notion of *capacity*, denoted $C(K)$, was introduced in [BDQ04]. It is defined as the sum of ELP of the ℓ linear approximations. The expected value $\mathrm{Exp}_K(C(K))$ and the variance $\mathrm{Var}_K(C(K))$ of the capacity $C(K)$ is computed as

$$\mathrm{Exp}_K(C(K)) = \sum_{i=1}^{\ell} \mathrm{ELP}(u_i, v_i) + \ell \cdot 2^{-n},$$

$$\mathrm{Var}_K(C(K)) = 2 \cdot \sum_{i=1}^{\ell} \left(\mathrm{ELP}(u_i, v_i) + 2^{-n}\right)^2.$$

Matrix method for ELP computation. The ELP plays an important role in the estimation of the complexity. Many previous linear cryptanalysis on PRESENT [BTV18, Cho10, NSZW09, Ohk09] exploit the 1-bit linear approximation of the S-box, which can effectively used to constructs 1-bit linear trail over multiple rounds using the piling-up lemma. A *1-bit linear trail* is a linear trail where the input and output masks of linear approximations of all intermediate rounds are of Hamming weight one. Let m be the bit size of the S-box. The following steps describe the matrix method to compute the ELP of linear hulls between any input and output masks of Hamming weight one from the 1-bit linear approximation of the S-box:

1. Compute the $m \times m$ matrix **LAT**, where:

$$\mathbf{LAT}[u', v'] = \#\{x \in \mathbb{F}_2^m | u' \cdot x \oplus v' \cdot \mathbf{S}[x] = 0\} - 2^{m-1},$$

   and $u', v' \in \mathbb{F}_2^m$ are 1-bit masks of the S-box.

2. Compute the $m \times m$ matrix **T** either by following Equation 2 or by deriving from:

$$\mathbf{T}[u', v'] = \mathbf{LAT}[u', v']/2^{m-1}.$$

3. Given **T** from the previous step, compute the $n \times n$ matrix **M** of square correlations of all 1-bit linear trails over one round. For 1-bit masks $u, v \in \mathbb{F}_2^n$, we denote i and j the positions of the single bit 1 in u and v, respectively. For every $i \in \{0, \ldots, n-1\}$, there are m possible 1-bit linear approximation through the S-box, thus m possible positions of j, *i.e.*, $j = \mathbf{P}[h_i(l)]$ where **P** is the permutation of the cipher, $l \in \{0, \ldots, m-1\}$ and h_i is a function depending on the permutation **P** for a given i. Then:

$$\mathbf{M}[i, j] = \mathbf{T}[i \mod m, l].$$

4. Compute the $n \times n$ matrix **ELP** of all possible linear hulls between any 1-bit masks over r' rounds:

$$\mathbf{ELP} = \mathbf{M}^{r'}.$$

Table 5a resumes the 1-bit linear approximations **LAT** of the PRESENT's S-box ($b = 64, m = 4$). The 4×4 correlation matrix **T** can be derived by $\mathbf{T}[u', v'] = \mathbf{LAT}[u', v']/8$. Based on the PRESENT's permutation **P**, the function h is defined as $h_i(l) = 4 \times \lfloor i/4 \rfloor + l$, where $l \in \{0, \ldots, 3\}$. Then, the 64×64 matrix **M** is computed as $\mathbf{M}[i, j] = \mathbf{T}^2[i \mod 4, l]$ for $i \in \{0, \ldots, 63\}$ and $j = h_i(l)$. An illustrative example of this method can be found in the Appendix C of [BN16].

Table 5 shows 1-bit linear approximation tables **LAT**s of the faulty S-boxes. We can observe that, for the original S-box, 8 non-zero biased approximations in **LAT** holds the probability $1/2 \pm 2^{-3}$, while there exists several considerably higher biased approximations (-4 and 4) corresponding to the probability $1/2 \pm 2^{-2}$ in $\mathbf{LAT}'$ and $\mathbf{LAT}''$. This is

Table 5: 1-bit linear approximation tables

$u'\backslash v'$	1	2	4	8
1	0	0	0	0
2	0	2	-2	2
4	0	-2	-2	-2
8	0	2	0	-2

(a) **LAT** for **S**

$u'\backslash v'$	1	2	4	8
1	0	2	-2	-2
2	0	4	-4	0
4	0	0	-4	-4
8	0	0	2	0

(b) $\mathbf{LAT}'$ for $\mathbf{S}'$

$u'\backslash v'$	1	2	4	8
1	0	2	2	-2
2	0	2	-2	4
4	0	-2	-2	-2
8	0	4	2	-4

(c) $\mathbf{LAT}''$ for $\mathbf{S}''$

expected to be the main reason making the ELP higher, hence the lower complexity of the linear attack.

We only needs to approximate $r' = 27$ rounds (out of 31 rounds for PRESENT) to mount a key recovery attack [FN20]. For each possibility, we compute the 64×64 matrix **ELP** over 27 rounds, $\mathbf{ELP} = \mathbf{M}^{27}$, following the steps presented above, then select the highest value in this matrix as the ELP of the S-box. We choose the faulty S-box with the highest ELP as the best target for the attacker among all possible faulty S-boxes.

B Example on SMALLPRESENT

To show the practicality of the key recovery, we provide a toy example to demonstrate the attack on the small version of PRESENT, named SMALLPRESENT. This cipher is composed of 6 rounds and has block size $n = 16$. The key size is 32 bits where the first half is used for the first, fifth, sixth, seventh round keys and the second haft is used for the second, third, fourth round keys.

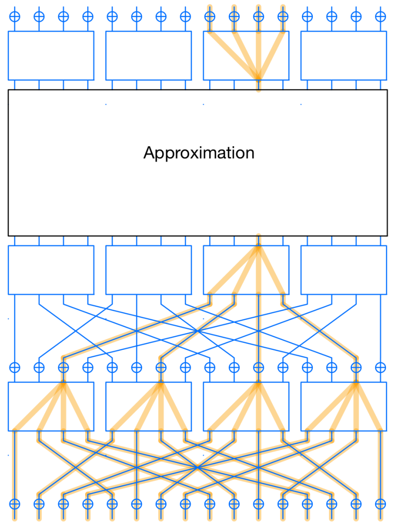

Figure 8: A linear approximation on SMALLPRESENT.

We perform linear approximations for 3 internal rounds and make guess of the subkey of the first and the last two rounds associated to the approximations. For example, Figure 8 shows the 1-bit approximation of the bit at index 5 (input of the approximation) to the bit at index 5 (output of the approximation). In this case, we need to make guess of 16 out of 32 bits for the key.

We compute the values of ELP over 3 rounds and choose $\ell = 4$ approximations whose

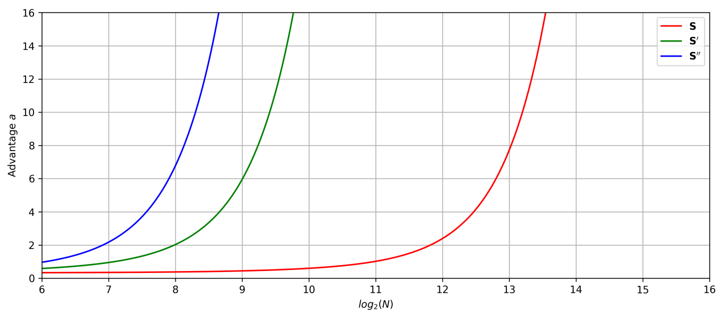

Figure 9: Advantage for attack on SMALLPRESENT with different S-boxes. The success probability is fixed to $P_S = 0.80$. The number of linear approximations is $\ell = 4$.

values of ELP are high. These approximations give the estimation of data complexity and advantage as depicted in Figure 9. If we choose the advantage $a = 16$ (out of 16 bits to be guessed), it means that we expect the first candidate (corresponding to the highest rank by the metric of the attack) should be the right key. Table 6 shows the number of data required for the attacks with different S-boxes and their actual success probability from the experiment. As we can see, the data required to perform the attacks with the faulty S-boxes is significantly less that with the original S-box, while the success probability is still maintained. The actual success probabilities in our experiment are slightly less than the expected ones. This can be explained by the variances in the estimation and the number of repetitions.

Table 6: Comparison of linear attacks on SMALLPRESENT with different S-boxes. We repeat each attack 100 times to calculate the actual success probability (Actual P_S).

S-box	#Approx.	Capacity	Advantage	Data	Expected P_S	Actual P_S
$\mathbf{S}$	4	$2^{-8.4}$	16	$2^{13.78}$	0.80	0.77
$\mathbf{S}'$	4	$2^{-4.3}$	16	$2^{9.90}$	0.80	0.74
$\mathbf{S}''$	4	$2^{-3.5}$	16	$2^{8.77}$	0.80	0.75