SCADFA: Combined SCA+DFA Attacks on Block Ciphers with Practical Validations

Sikhar Patranabis, Nilanjan Datta, Dirmanto Jap, Jakub Breier, Shivam Bhasin, and Debdeep Mukhopadhyay

Abstract—We present the first practically realizable side-channel assisted fault attack on any block-ciphers having bit-permutation with optimal diffusion, that can retrieve the round key efficiently using random nibble faults. The attack demonstrates how side-channel leakage can allow the adversary to precisely determine the fault mask resulting from a nibble fault injection instance. We first demonstrate the viability of such attack model via side-channel analysis experiments on top of a laser-based fault injection setup, targeting a PRESENT-80 and GIFT-128 (two popular block-ciphers based on bit-permutation having optimal diffusion) implementation on an ATmega328P microcontroller. Subsequently, we present a differential fault analysis (DFA) exploiting the knowledge of the output fault mask in the target round to recover multiple last round keys nibbles independently and in parallel. We show that the combined attack can recover the last round key of PRESENT-80 and GIFT-128 with 4 random nibble fault injections in the best case. In the average case, the number of random nibble faults required for PRESENT-80 and GIFT-128 are 9-18 and 6-9 respectively.

Keywords: DFA, DPA, PRESENT, GIFT, combined attacks, fault attacks, side-channel analysis, bit-permutation, optimal diffusion

1 Introduction

Lightweight cryptography has become a necessity with the widespread use of small computing devices that are interconnected and used for various simple control tasks – also known as Internet of Things (IoT). Ciphers used for this purpose are designed with the goal of low computational and area overhead so that they could fit easily into IoT devices and consume less power than their traditional counterparts, developed mostly for use in personal computers and workstations. Since IoT components can be deployed almost in any environment, including publicly available areas, physical attacks are a potential threat, including Side-Channel Attacks (SCA) and Fault Attacks (FA). Therefore, it becomes a necessity to evaluate the strength of lightweight ciphers against these types of attacks.

Recent trends in constructing lightweight designs inspired a whole new family of SPN based block ciphers that use bit-permutations as opposed to maximum distance separable (MDS) layers for achieving the necessary diffusion characteristics. Bit-permutations have zero-cost implementation in hardware as they can be realized solely via wiring. PRESENT [1] is the first and one of the most popular lightweight block ciphers following this paradigm. Several new proposals for lightweight block ciphers such as Rectangle [2] and GIFT [3] have also opted for bit-permutations for efficiency in hardware.

- *Sikhar Patranabis, Nilanjan Datta and Debdeep Mukhopadhyay are with the Department of Computer Science and Engineering, IIT Kharagpur, India.*
  E-mail: {sikhar.patranabis,nilanjan.datta@iitkgp.ac.in}@iitkgp.ac.in, debdeep@cse.iitkgp.ac.in.
- *Dirmanto Jap and Shivam Bhasin are with Temasek Labs, NTU Singapore.*
  E-mail: {djap,sbhasin}@ntu.edu.sg
- *Jakub Breier is with School of Computer Science and Engineering, NTU Singapore.*
  E-mail: jbreier@jbreier.com

1.1 SCA and DFA Attacks on Block-ciphers

Side-Channel Analysis (SCA). Side-channel analysis (SCA) constitutes one of the greatest threats to the security of cryptographic implementations for real-world applications. Several side-channels such as timing, power and electromagnetic emission have been successfully exploited from the devices implementing cryptographic algorithms to recover the secret key [4], [5], [6], [7]. One of the most powerful among them is the Differential Power Analysis (DPA) attack, which consists of a four step procedure. In the first step, the attacker records the power consumption of the device while it is performing cryptographic operations at different instances in time (*samples*) for a number of random inputs (*traces*). Next, they compute the value of a chosen intermediate variable for all the possible key candidates. Note that in a DPA attack we use divide-and-conquer approach and hence the brute-force search here is limited to a part of the full key (*e.g.* 8-bits in case of AES or 4-bits in case of PRESENT). Then, they map the hypothetical intermediate values computed in Step 2 to the hypothetical power values using a suitable power model (*e.g.* Hamming weight, Hamming distance). Finally, a statistical distinguisher (such as distance-of-means and Pearson's correlation coefficient) is used to distinguish between the correct and wrong key candidates.

Fault Analysis (FA). Alongside SCA, fault analysis (FA) attacks also pose a serious threat in modern cryptographic implementations. In this type of attacks, the targeted device is forced to operate under some abnormal operating conditions by injecting faults through modifications of the power supply, clock source by injecting glitches. This essentially produces erroneous outputs, resulting in revealing secret information such as internal states of a cipher to the entire secret key. This implies that several hardware components like smart cards, mobile devices and many other devices

associated with cryptographic applications requires fault resistance.

One of the most popular fault based attack, called *differential fault analysis* (DFA) has been applied on DES by Biham *et al*. Later extensive DFA has been done on popular block ciphers such as AES, LED, PRESENT. Several differential fault attacks have been mounted on AES [8], [9]. In [10], Piret *et al*. introduced a generic fault attack applicable on a class of SPN based block cipher. They have been able to break the AES-128 with only 2 faulty cipher texts, assuming the fault occurs between the antepenultimate and the penultimate Mix Column stage. One of the most optimal fault analysis on AES has been reported by Tunstall *et al*. [11] where a single random byte fault is induced at the input of the eighth round to reduce the key space to 2^{32} in the first step and a mere 2^8 after the second step. Several DFAs have been applied on PRESENT too. The first one, published by Wang *et al*. [12] required 64 pairs of correct and faulty cipher text on average, with a computational complexity of 2^{29}. Later, Zhao et al. [13] utilized a fault-propagation pattern-based DFA, targeting PRESENT and PRINT. This attack on PRESENT required 8/16 cipher text pairs on average, reducing the key search space to $2^{14.7}/2^{21.1}$ in average. Bagheri et al. [14] showed attacks utilizing single bit-flip and single nibble fault models. The first attack obtains the last sub key with 48 cipher texts pairs, while the second attack reveals the key with 18 cipher text pairs on average. Breier and He [15] proposed a multiple fault attack, targeting four nibbles at once, being able to recover the secret key in just 2 encryptions. DeSantis et al. [16] presented a cipher text-only attack, which requires only two cipher text pairs in the best case. Ghalaty et al. [17] attacked PRESENT and LED ciphers with Differential Fault Intensity Analysis (DFIA), showing that both ciphers can be broken with a practically feasible number of fault injections.

Differential behavioral analysis (DBA [18]) is a combined SCA with safe-error attacks. A combined SCA and FA targeting first AES key addition was proposed in [19]. Roche et al. proposed a DFA on AES key schedule in [20] by injecting faults in pen-ultimate round key computation, further improved in [21]. All these attacks were demonstrated in simulated settings. Combinations of fault sensitivity analysis (FSA) with collision correlation attack (CCA) [22] and CPA were also proposed [23].

1.2 Our Contribution

Central to our work are block ciphers based on bit-permutations. We have observed that for these block ciphers, the knowledge of the output fault mask in an earlier round can be exploited to significantly reduce the entropy of the input fault mask at a later round. In this work we try to exploit this possible vulnerability of bit-permutation to mount a generic fault attack. The main contributions of this paper are two folded:

1) We propose SCADFA, a very efficient practically realizable attack on any block ciphers having bit-permutation with optimal diffusion. SCADFA, as the name implies, is a combination of side-channel analysis (SCA) and differential fault attack (DFA). We consider a relaxed fault model corresponding to a given target round, and assume that the adversary uses side-channel leakage to precisely determine the resulting fault mask. We present a theoretical analysis to establish that the attack is capable of recovering multiple key nibbles in parallel under the same fault injection instance, implying that the attack is not only feasible but also efficient.

2) We show the applicability of our attack by demonstrating SCADFA attacks on PRESENT and GIFT, two of the most popular block-ciphers based on bit-permutation. We demonstrate the combined attack on an ATmega328P micro-controller based implementation of PRESENT and GIFT-128 using a laser-driven fault injection setup. Our experiments demonstrate that the proposed attack recovers

   - 64 bits of the last round key of PRESENT using approximately 9-18 fault injections in the average case,
   - 32 bits of the last round key of GIFT-64 using approximately 9-18 fault injections in the average case,
   - 64 bits of the last round key of GIFT-128 using 6-9 fault injections in the average case.

1.3 Significance of the Work

As mentioned already, our attack is a generic one and applies to any block cipher having bit permutation with optimal diffusion. To the best of our knowledge, this is the first time where such a generic fault attack (side-channel assisted) has been exploited, covering the complete class of bit permutation based block-ciphers. Prior attacks on block ciphers have mostly been limited to either SCA or FA. This work depicts an inherent vulnerability of bit-permutation based block ciphers, and opens a scope to build light-weight block ciphers having side channel and fault resistant. The presented results are important in context of the recently launched NIST competition for lightweight cryptography, where bit permutation is a potential choice for cipher construction.

1.4 Road Map of the Paper

The remainder of the paper is organized as follows. In Sec. 2, we provide a basic overview of block ciphers having bit-permutations and briefly describe PRESENT and GIFT. In Sec. 3 we first briefly discuss the fault model and why side-channel information is required for the attack. Then we describe the formal algorithm of SCADFA attack and show how key recovery can be done very efficiently using our algorithm. Sec. 4 and 5 deals with the practical application of our attack by showing attack on PRESENT and GIFT, respectively. All the experimental results are provided in Sec. 6 and further discuusion in Sec. 7. Finally, we conclude in Sec. 8 with interesting open problems.

2 PRELIMINARIES

2.1 Block Ciphers

A block cipher is a keyed permutation *Enc* that takes a *plain text* M of size d-bits and produce a *cipher text* C of d-bits using a *key* K of k-bits. Formally, we can express this encryption function as a Boolean mapping:

$$C := Enc(M, K) \colon \{0,1\}^d \times \{0,1\}^k \rightarrow \{0,1\}^d$$

As mentioned by Shannon, any block cipher should have two properties namely *confusion* and *diffusion*. Confusion refers that each bit of the ciphertext should depend on several parts of the key, obscuring any connection between the two where as diffusion means that if we change a single bit of the plaintext, then (statistically) half of the bits in the ciphertext should change, and similarly, if we change one bit of the ciphertext, then approximately one half of the plaintext bits should change. Block ciphers are typically implemented in any one following ways:

□ **Feistel Network.** In this type of structure, in every round the input plain text is usually divided into two parts. Then a round function F is applied to one of the parts and the output is xor-ed with other part of the plain text and the two parts are then swapped. The underlying *F-function* in Feistel network need not be invertible. DES, CLEFIA etc. are few examples of Feistel based block cipher.

□ **Substitution Permutation Network (SPN).** This is another popular structure of block ciphers. This type block cipher has a very simple design principle where the round function is simply composed of:

- Substitution Layer. This typically contains identical small S-Boxes. The S-Boxes are non-linear functions and creates the necessary confusion.
- Permutation Layer. The substitution layer is followed by a permutation layer. This layer used to provide the necessary diffusion. Typical choice is a byte-permutations followed by a mix column type operation. However for lightweight applications, a simple bit-permutation can be used in the permutation layer.
- Sub-Key Addition Layer. Finally a sub-key is xored.

AES, LED, PRESENT, GIFT etc. are few examples of SPN based block cipher. Now we will briefly describe PRESENT and GIFT as we will illustrate the SCADFA attacks on these two.

2.2 Overview of PRESENT

PRESENT is a very popular 64-bit SPN based block cipher with key size of length 80 or 128 bits, exclusively designed to be used in light-weight applications. Each round of PRESENT is composed of:

- a layer of 16 identical small 4-bit S-Boxes (Table 1),
- a bit permutation layer (shown in Figure 1) and
- a sub-key addition.

Overall, PRESENT consists of 31 rounds and at the end of round 31, a post-whitening XOR with the round key is performed, so 32 round keys are generated in total. The key schedule for PRESENT comprises of a rotation, S-Box look-up and round counter addition, thus making it invertible.

TABLE 1: The PRESENT S-Box

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

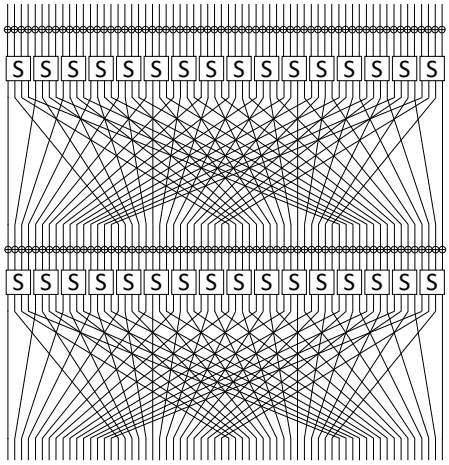

Fig. 1: Structure of two rounds of PRESENT

2.3 Overview of GIFT

It is another very popular SPN based block cipher with bit permutation. GIFT has two variants: GIFT-64 (64 bit) and GIFT-128 (128 bit). GIFT-64 has consists of $r = 28$ rounds and GIFT-128 has consists of $r = 40$ rounds. Both versions of GIFT support keys of length 128 bits. Each round of GIFT is composed of:

- a layer of 16 (or 32) identical small 4-bit S-Boxes (Table 2),
- a bit permutation layer (shown in Figure 2 for GIFT-128) and
- a sub-key addition.

TABLE 2: The GIFT S-Box

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

3 GENERALIZED SCADFA ATTACKS ON BPOD BLOCK-CIPHERS

In this section, we first define Bit-Permutation with Optimal Diffusion (in short BPOD). As the name implies, BPOD is a SPN type of block-cipher construction with bit-permutation that achieves optimal diffusion.

An (n, m) BPOD block-cipher is a $d = n^2.m$ bit block cipher, that uses n-bit S-Boxes. In other words, there are nm groups of S-Boxes, each of n-bits.

3.1 Properties of BPOD Block Ciphers

First we list down all the necessary as well as sufficient properties of BPOD block-ciphers to have an optimal diffusion:

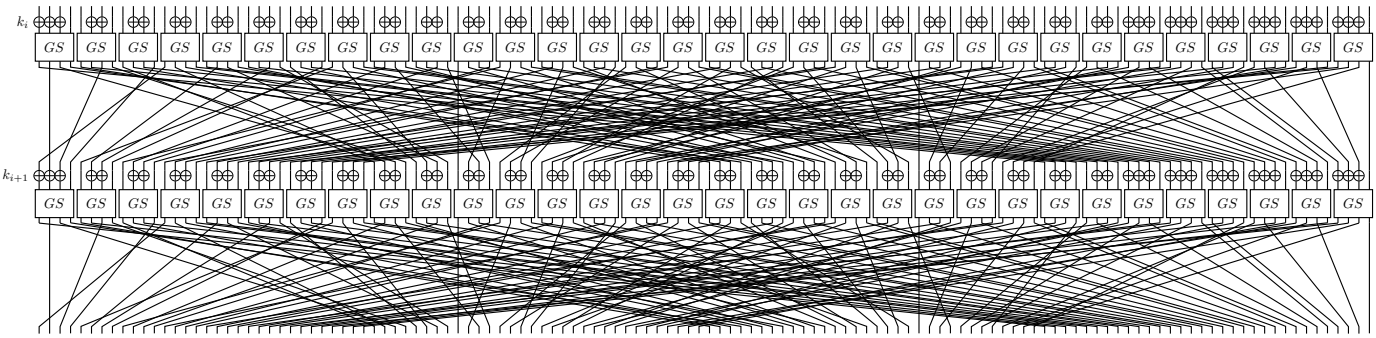

Fig. 2: Structure of two rounds of GIFT-128

- The input of an S-Box in round r comprises output bits from n different S-Boxes in round $r-1$.
- The output of an S-Box in round r is distributed across the inputs of n different S-Boxes in round $r+1$.
- The output of the S-Box group $\{n.i, n.i+1, \ldots, n.i+(n-1)\}$ in round r entirely constitutes the input for the S-Box group $\{i, i+m, \ldots, i+(n-1).m)\}$ in round $r+1$, where $i \in \{0, 1, \ldots, (m-1)\}$.
- Full diffusion occurs in $\xi = \lceil \log_n n^2.m \rceil$ rounds. Note that, for any choices of m, n with $n \leq m < n^2$, we have $\xi = 3$ or 4.

3.2 Fault Model and Fault Location

Our generalized attack works on random chunk fault model. Usual choices of a chunk would be $n = 4$ and $n = 8$, which essentially makes it random nibble fault model and random byte fault model, respectively. Nibble or byte faults have been demonstrated to be practically achievable using traditional fault injection techniques such as clock and voltage glitches [24], [25] as well as more advanced injection methods such as electromagnetic (EM) pulse or laser pulse injection [26], [27]. The fault is injected at the input of round $(r-3)$ during the BPOD encryption operation. We use the term *output fault mask* to denote the differential Δ_{out} of the correct and faulty block at output of S-Box operation in round $(r-3)$. Here we would like to emphasis that many fault attacks (such as the one used in [15] during a fault attack on PRESENT) demand fault injections on specific locations for the desired output fault mask. Our attack, on the other hand, assumes only a random chunk fault, without any specific requirements on the nature of the corresponding output fault mask. This makes our fault model more relevant in the context of real-world fault attacks.

3.3 Determining Fault Mask using SCA

The objective of introducing side-channel leakage in our analysis is to deterministically obtain the output fault mask Δ_{out} introduced in round $(r-3)$. Note that since our attack assumes a random fault in a single nibble, the output fault mask Δ_{out} is unknown. So, we need this side-channel analysis to determine Δ_{out}. We do so by the following method:

- Since the fault is injected in a particular nibble during round $(r-3)$, leakage traces corresponding to the fault-free and faulty computations in round $(r-2)$ are collected.
- The difference (in side-channel measurements) of correct and faulty execution are computed. The bit-permutation pattern makes it possible to determine the exact value of Δ_{out}.

3.4 Fault Propagation and Mounting the Attack

We now present the fault propagation characteristics corresponding to our attack in details. As already mentioned, the attack targets a single chunk in round $(r-3)$ of BPOD. Here we state the main result regarding the fault propagation in the following theorem:

Theorem 3.1. For any (n, m) BPOD construction with r rounds, if the Hamming weight of the output fault mask of the target chunk in round $(r-3)$ is x, then the fault will spread to at least n^2 many chunks in the input of round r and the Hamming weight of the input fault mask of any chunk in round r must be less than or equals x. Additionally, if the Hamming weight of the output fault mask of the target chunk in round $(r-3)$ of BPOD is x then the input fault mask of any chunk in round r takes at most 2^x values.

Proof. We present the fault mask characteristics in each of rounds $(r-3), (r-2), (r-1)$ and r as follows:

- Suppose the adversary injects a output fault in round $(r-3)$ at chunk $n.i+j$, where $i \in \{0, 1, \ldots, (m-1)\}$ and $j \in \{0, 1, \ldots, (n-1)\}$, and suppose the output fault mask has Hamming weight $x \in \{0, 1, \ldots, (n-1)\}$.
- According to the generic properties of BPOD, this fault will be propagated in round $(r-2)$ to x many chunks among the following (depending on the exact permutation): $i, i+m, \ldots, i+(n-1).m$ respectively, with an input fault mask of Hamming Weight 1.
- Each of the above input faults of round $(r-2)$ will be propagated to round $(r-1)$ to n many chunks ($a, a+m, \ldots, a+(n-1).m$ for some $1 \leq a < m$) with an input fault mask of Hamming Weight 1. Overall there will be exactly $n.x$ many chunks in round $(r-1)$ with an input fault mask of Hamming weight 1. This is due to the fact that (i) the output of an S-Box propagates to distinct n S-Boxes in the next round and (ii) the output of two S-Boxes $i + k_1.m$ and $i +$

$k_2.m$ (where $1 \leq k_1 \leq k_2 < n$) can not propagate to same S-Box in the next round. Note that, from any group of chunks $\{m.k, \ldots, m.k + (m-1)\}$, the fault affects a maximum of x many chunks.

- Using the above note and the generic properties of BPOD, it is easy to see that the input faults of round $(r-1)$ will be propagated to round r to at least n^2 many chunks with each chunk having an input fault mask of Hamming Weight equals or less than x. Here we use the fact that the faults corresponding to the S-Boxes $a, a+m, \ldots, a+(n-1).m$ propagates to exactly n^2 many distinct S-Boxes in one round.
- If $m = c.n$, then it essentially means that the fault will potentially spread to exactly n^2 many chunks to the inputs of round r with an Hamming weight x.

Thus, each nibble in round r has an input fault mask of Hamming Weight at most x. Additionally, since at most x bits of each input fault mask are 1, and the faulty bits are uniquely determined from the given permutation, each input fault mask in round r can take 2^x values. This completes the proof of theorem 3.1.

Observation 3.2. Note that, if $m = c.n$ and the output fault is injected in chunk $\{n^2.i, n^2.i+1, \ldots, n^2.i+(n^2-1)\}$ in round $(r-3)$ for some $i \leq m$, then the input fault can affect only the following chunks: $\{i, c+i, \ldots, (n^2-1).c+i\}$.

3.5 Key Recovery

The fault propagation characteristics described above can now be used to recover multiple key nibbles in parallel. For clarity of presentation, we consider a slightly modified version of BPOD, where we ignore the bit-permutation operation of round r. In the absence of the final bit-permutation layer, for any nibble with a non-zero input fault mask, the output is directly XOR-ed with the last round key nibble and output as the ciphertext. Thus, given a correct ciphertext nibble C and a faulty ciphertext nibble C', corresponding to a non-zero input fault mask β, we have the following differential relation involving the corresponding final round key nibble K:

$$S^{-1}\left[C \oplus K\right] \oplus S^{-1}\left[C' \oplus K\right] = \beta$$

where S^{-1} denotes the inverse S-Box operation. Assuming that the underlying BPOD S-Box has good differential uniformity, the expected number of values of K that satisfy the above equation is one. Now, assuming a non-zero output fault mask with Hamming weight x for the target nibble in round $(r-3)$, there are $2^x - 1$ possible values of the input fault mask β in round r (according to Theorem 3.1), which in turn gives rise to $2^x - 1$ possible differential relations as described above. Hence, any value of x in the set $\{1, 2, \ldots, (n-1)\}$ reduces the entropy of the key nibble K by a factor of approximately 2^{n-x} on an average, and is hence expected to allow recovering K uniquely after $n/(n-x)$ fault injections (under the assumption that all faults help recover unique key bits).

For PRESENT-64 and GIFT-64, optimal diffusion takes $\xi = 3$ rounds. In other words, if a fault is injected in round $(r-3)$, then it spreads to every nibble of the state by round r. For GIFT-128, any fault injected in round $(r-3)$ spreads to half of the cipher state by round r. To cover all nibbles, the attacker only needs to inject two faults.

From an attack efficiency point of view, it is desirable that the injected fault spreads to as many state nibbles as possible, since this allows the attacker to recover the maximum possible key nibbles with only few fault injections. Taking all these facts into consideration, round $(r-3)$ seems to be the ideal attack point. Hence, *faults with output mask of Hamming weight 4 are not useful for our attack*.

It is also worth observing that we do not require separate fault injection instances for recovering each key nibble. On the contrary, each fault injection instance in round $(r-3)$ is expected to yield multiple faulty nibbles in round r, and each of these nibbles may be analyzed independently and in parallel for key recovery. This inherent parallelism makes the attack very efficient and reduces the overall number of fault injections necessary to recover d bits of the last round key. The attack efficiency is also supported by experimental results in Section 6.

4 Application of SCADFA to the PRESENT Block Cipher

PRESENT is a very popular $d = 64$ bit BPOD construction with parameters $n = 4$ and $m = 4$. It consists of $r = 31$ rounds. PRESENT supports keys of length 80 and 128 bits. Here we will focus only on the 80 bit key length version, denoted by PRESENT-80. However the attack can be trivially extended to 128-bit version as well.

From the bit permutation for PRESENT, it is easy to verify that the output of the S-Box group $\{4.i, \ldots, 4.i+3\}$ in round r entirely constitutes the input for the S-Box group $\{i, i+4, i+8, i+12\}$ in round $r+1$, where $i \in \{0,1,2,3\}$. More precisely, for $i, j, l \in \{0,1,2,3\}$, the l^{th} bit in the output of S-Box $4i+j$ in any round is precisely the j^{th} bit in the input of S-Box $i+4l$ in the next round, albeit after XOR-ing with the corresponding round key bit.

4.1 Fault Model and Fault Location for SCADFA on PRESENT

As $n = 4$, we will consider random nibble fault model. The fault is injected at the input of round 28 during a PRESENT encryption operation. The timing of fault injection is based on the fact that PRESENT is bit-permutation-based with optimal diffusion. As already explained in Section-3.1, for an (n, m) BPOD block-cipher, full diffusion occurs in $\xi = \lceil \log_n n^2.m \rceil$ rounds. Additionally, for any choice of m, n with $n \leq m < n^2$, we have $\xi = 3$ or 4. For PRESENT-64, optimal diffusion takes $\xi = 3$ rounds. In other words, if a fault is injected in round $(r-3)$, then it spreads to every nibble of the state by round r.

From an attack efficiency point of view, it is desirable that the injected fault spreads to as many state nibbles as possible, since this allows the attacker to recover the maximum possible key nibbles with only few fault injections. Taking all these facts into consideration, round $(r-3)$ seems to be the ideal attack point. Since PRESENT has 31 rounds, the fault is injected in round 28.

We would like to point out that recent attacks on PRESENT, such as that presented by Breier et al. in [15],

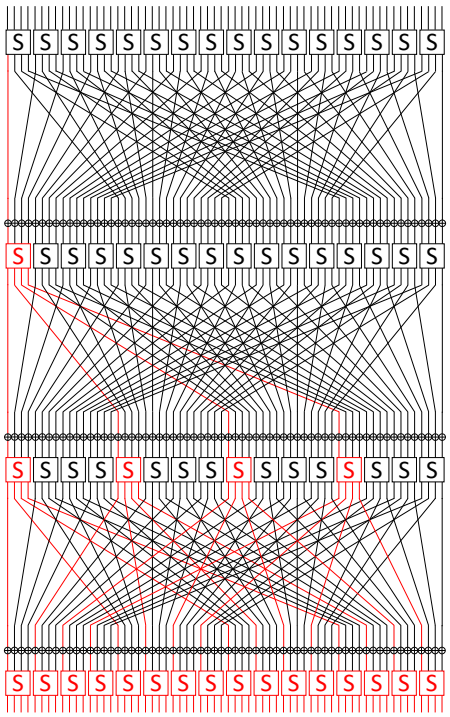

Fig. 3: Fault propagation in PRESENT for the output fault mask 0011

assume some specific instances of nibble faults that result in a desired output fault mask. Our attack, on the other hand, assumes a random nibble fault, without any specific requirements on the nature of the corresponding output fault mask. This makes our fault model more relevant in the context of real-world fault attacks.

4.2 Concrete Examples of Fault Propagation in PRESENT

In this section, we present two concrete instances of our proposed SCADFA on PRESENT, for fault masks with Hamming weight 1 and 2, respectively. Besides serving to elucidate the nature of the fault propagation, these instances also validate that Theorem 3.1 indeed holds with respect to the bit permutation of PRESENT.

4.2.1 Fault Mask with Hamming weight 1

Suppose the output fault mask of the target nibble in round 28 of PRESENT is 0001. This implies that in round 29, nibble 0 has an input fault mask of 0001. Unfortunately, the corresponding output fault mask is non-deterministic: all one can infer is that each of the nibbles 0, 4, 8 and 12 in round 30 have an input fault mask of 0000 (implying no fault propagation) or 0001 (implying fault propagation). The output fault masks in round 30 are non-deterministic; however, one can easily make the following observations:

- If the input fault mask for nibble 0 in round 30 is 0001, then the input fault mask for nibbles 0, 4, 8 and 12 in round 31 are either 0000 or 0001. On the other hand, if the input fault mask for nibble 0 in round 30 is 0000, then the input fault mask for nibbles 0, 4, 8 and 12 in round 31 is definitely 0000.
- If the input fault mask for nibble 4 in round 30 is 0001, then the input fault mask for nibbles 1, 5, 9 and 13 in round 31 are either 0000 or 0001. The case of input fault mask 0000 follows analogously.
- If the input fault mask for nibble 8 in round 30 is 0001, then the input fault mask for nibbles 2, 6, 10 and 14 in round 31 are either 0000 or 0001. The case of input fault mask 0000 follows analogously.
- Finally, if the input fault mask for nibble 12 in round 30 is 0001, then the input fault mask for nibbles 3, 7, 11 and 15 in round 31 are either 0000 or 0001. The case of input fault mask 0000 follows analogously.

Thus, for each of the nibbles in round 31, the input fault mask is either 0000 or 0001, and has Hamming weight at most 1, which is in accordance with Theorem 3.1. Figure 3 illustrates the fault propagation characteristics for the above example.

4.2.2 Fault Mask with Hamming weight 2

Now, suppose the output fault mask of the target nibble in round 28 of PRESENT is 0011. This implies that in round 29, nibble 0 and nibble 4 has an input fault mask of 0001. We can further infer the following (see Figure 4 for an illustration):

- Each of the nibbles 0, 4, 8 and 12 in round 30 have an input fault mask of 0000 (implying no fault propagation) or 0001 (implying fault propagation).
- Similarly, each of the nibbles 1, 5, 9 and 13 in round 30 have an input fault mask of 0000 (implying no fault propagation) or 0001 (implying fault propagation).

The analysis in case of round 31 now becomes more complicated. So we focus on the input fault mask for a specific set of nibbles - 0,4,8 and 12 respectively:

- If the input fault mask for both nibble 0 and nibble 1 in round 30 is 0000, then the input fault mask for nibbles 0, 4, 8 and 12 in round 31 is definitely 0000.
- If the input fault mask for nibble 0 and nibble 1 in round 30 are 0000 and 0001, respectively (equivalently 0001 and 0000, respectively), the input fault mask for nibbles 0, 4, 8 and 12 in round 31 is 0001 (equivalently 0010).
- Finally, if the input fault mask for both nibble 0 and nibble 1 in round 30 is 0001, then the input fault mask for nibbles 0, 4, 8 and 12 in round 31 is definitely 0011.

The input fault mask for the other sets of nibbles may be similarly characterized. Thus, once again, the input fault mask for each of the nibbles in round 30 has Hamming Weight at most 2, and takes at most $2^2 = 4$ values. Thus is again in accordance with Theorem 3.1.

4.3 Key Recovery via SCADFA on PRESENT

Having validated that Theorem 3.1 indeed holds with respect to PRESENT, we now explicitly illustrate the key-recovery procedure. Assuming a non-zero output fault mask

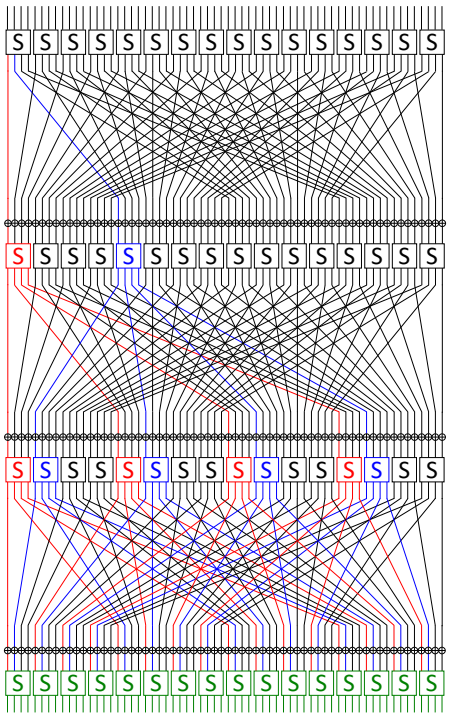

Fig. 4: Fault propagation in PRESENT for the output fault mask 0011

with Hamming weight x for the target nibble in round 28, there are $2^x - 1$ possible values of the input fault mask β in round 31 (applying Theorem 3.1 with $n = m = 4$), which in turn gives rise to $2^x - 1$ possible differential relations as described above. Hence, any value of x in the set $\{1, 2, 3\}$ reduces the entropy of the key nibble K by a factor of approximately 2^{4-x} on an average, and is hence expected to allow recovering K uniquely after $4/(4 - x)$ fault injections (under the assumption that all faults help recover unique key bits). For $x = 1, 2$ and 3, the expected number of fault injections are thus 1.33, 2 and 4 respectively. In practice, the required number of fault injections would be slightly higher since the affected key bits would likely overlap; however, key-recovery would still be feasible. On the other hand, for $x = 4$, all 15 possible values of β could potentially occur during the attack; hence, *faults with output mask of Hamming weight 4 are not useful for our attack*.

It is also worth observing that we do not require separate fault injection instances for recovering each key nibble. On the contrary, each fault injection instance in round 28 is expected to yield multiple faulty nibbles in round 31, and each of these nibbles may be analyzed independently and in parallel for key recovery. This inherent parallelism makes the attack very efficient and reduces the overall number of fault injections necessary to recover 64 bits of the last round key. The attack efficiency is also supported by experimental results in Section 6.

5 Application of SCADFA to the GIFT Block Cipher

GIFT is another recently proposed and promising lightweight BPOD block cipher construction. GIFT has two variants: GIFT-64 ($d = 64$ bit block cipher) and GIFT-128 ($d = 128$ bit). GIFT-64 has similar parameters $n = 4$ and $m = 4$ (same as PRESENT) and it consists of $r = 28$ rounds. GIFT-128 has parameters $n = 4$, $m = 8$ and it consists of $r = 40$ rounds. Both versions of GIFT support keys of length 128 bits. Here we will mainly focus on GIFT-128 as the attack on GIFT-64 is nearly identical to the attack shown on PRESENT (requiring 9-18 faults on average) in Section 4.

For GIFT-128, it is easy to verify that the output of the S-Box group $\{4.i, 4.i + 1, 4.i + 2, 4.i + 3\}$ in round r entirely constitutes the input for the S-Box group $\{i, i + 8, i + 16, i + 24\}$ in round $r + 1$, where $i \in \{0, 1, \ldots, 7\}$.

5.1 Fault Model, Fault Location and Fault Propagation for SCADFA on GIFT-128

As $n = 4$, we will still consider random nibble fault model without any specific requirements on the nature of the corresponding output fault mask. The fault is injected at the input of round 37 during a GIFT-128 encryption operation. The timing of fault injection is again based on the fact that GIFT is bit-permutation-based with optimal diffusion. For GIFT-128, any fault injected in round $(r - 3)$ spreads to half of the cipher state by round r. To cover all nibbles, the attacker only needs to inject two faults in round $(r - 3)$. This is again useful from an attack efficiency point of view, since it ensures that the attacker can recover the maximum possible key nibbles with only few fault injections.

We present a concrete fault injection instance using a fault mask of Hamming weight 1 to illustrate the nature of the fault propagation for GIFT-128 (refer Fig. 5). Using this instance, we validate that Theorem 3.1 also holds with respect to the bit permutation of GIFT-128.

Suppose the output fault mask of the target nibble in round 37 of GIFT-128 is 0001, which in turn implies that in round 38, nibble 0 has an input fault mask of 0001. Again, the corresponding fault mask is non-deterministic: all one can infer is that each of the nibbles 0, 8, 16 and 24 in round 39 have input fault mask of 0000 (implying no fault propagation), or have individual input fault masks of 0010, 0100, 1000, and 0001, respectively. The output fault masks of round 39 are again non-deterministic; however, one can easily make the following observations:

- If the input fault mask for nibble 0 in round 39 is 0010, then the input fault mask for nibble 0 (respectively, 8,16 and 24) in round 40 is either 0000 (implying no fault propagation) or 0010 (respectively, 0100, 1000 and 0001). On the other hand, if the input fault mask for nibble 0 in round 39 is 0000, then the input fault mask for nibbles 0, 8, 16 and 24 in round 40 is definitely 0000.
- If the input fault mask for nibble 8 in round 39 is 0100, then the input fault mask for nibble 2 (respectively, 10,18 and 26) in round 40 is either 0000 (implying no fault propagation) or 0010 (respectively, 0100, 1000 and 0001). On the other hand, if the input fault mask for nibble 8 in round 39 is 0000, then the input fault mask for nibbles 2, 10, 18 and 26 in round 40 is definitely 0000.
- If the input fault mask for nibble 16 in round 39 is 100, then the input fault mask for nibble 4 (re-

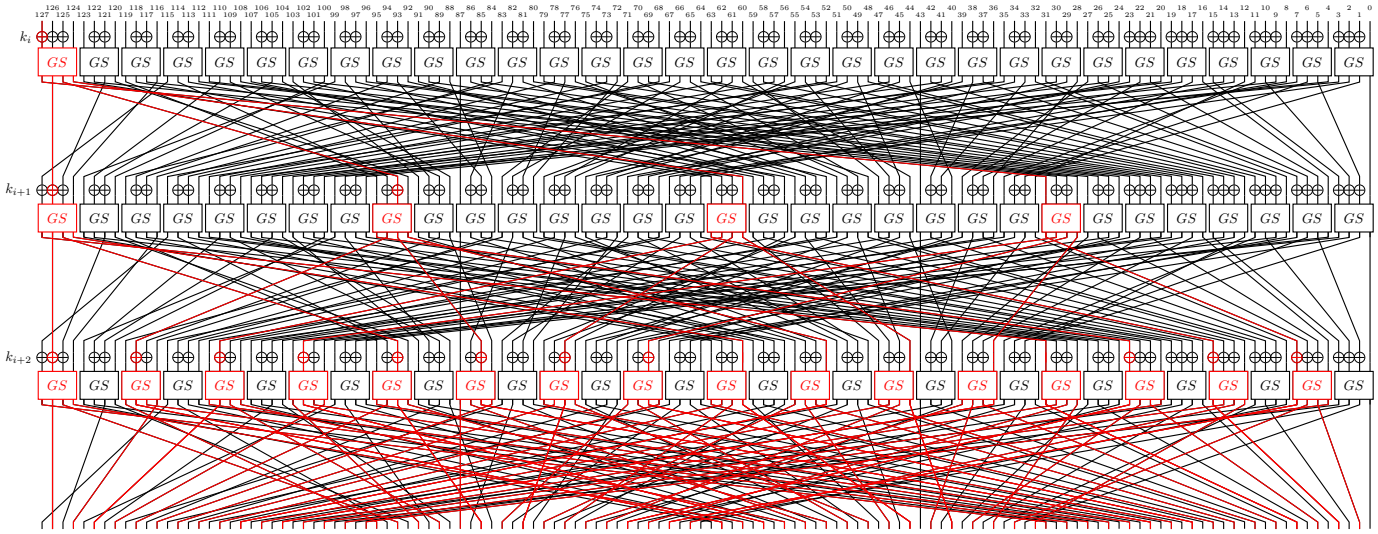

Fig. 5: Fault propagation in last three rounds of GIFT if MSB is faulted in round $n-3$.

spectively, 12,20 and 28) in round 40 is either 0000 (implying no fault propagation) or 0010 (respectively, 0100, 1000 and 0001). On the other hand, if the input fault mask for nibble 16 in round 39 is 0000, then the input fault mask for nibbles 4, 12, 20 and 28 in round 40 is definitely 0000.

- Finally, if the input fault mask for nibble 24 in round 39 is 0001, then the input fault mask for nibble 6 (respectively, 14,22 and 30) in round 40 is either 0000 (implying no fault propagation) or 0010 (respectively, 0100, 1000 and 0001). On the other hand, if the input fault mask for nibble 24 in round 39 is 0000, then the input fault mask for nibbles 6, 14, 22 and 30 in round 40 is definitely 0000.

Thus for each of the affected nibbles in the final round, the input fault mask takes at most two values, each with Hamming weight at most 1, which is in accordance with Theorem 3.1. Finally, note that unlike in PRESENT, in GIFT-128, a single fault instance in round 37 cannot propagate the fault to all nibbles at the input of round 40. This can be easily addressed by making two fault injections in round 37 instead of one - the first in any of the nibbles $[0, 15]$, and the second in one of the nibbles $[16, 31]$. The first injection instance will potentially propagate the fault to all even nibbles at the input of round 40, while the second injection instance will potentially propagate the fault to all odd nibbles at the input of round 40.

5.2 Key Recovery via SCADFA on GIFT-128

Now, assuming a non-zero output fault mask with Hamming weight x for the target nibble in round 37, there are $2^x - 1$ possible values of the input fault mask β in round 40 (this is proven using theorem 3.1 with $n = 8, m = 4$), which in turn gives rise to $2^x - 1$ possible differential relations as described above. As $m = 2.n$, by virtue of observation 3.2 we know if the output fault is injected on any nibble between $0 - 15$ (or $16 - 31$) in round 37, it can affect only the even nibbles (odd nibbles respectively) of round 40. Now, any value of x in the set $\{1, 2, 3\}$ reduces the entropy of the key nibble K by a factor of approximately 2^{4-x} on an average, and is hence expected to allow recovering K uniquely after $4/(4-x)$ fault injections, and as before the expected number of fault injections for $x = 1, 2$ and 3 are 1.33, 2 and 4 respectively.

Again, each fault injection instance in round 37 is expected to yield multiple faulty nibbles in round 40, and each of these nibbles may be analyzed independently and in parallel for key recovery. This inherent parallelism makes the attack very efficient and reduces the overall number of fault injections necessary to recover 64 bits of the last round key. The attack efficiency is also supported by experimental results in Section 6.

6 Experimental Results

6.1 Combined SCADFA Set-up

Our experimental setup is depicted in Figure 6. We have used laser fault injection (LFI) technique to disturb the device under test (DUT) and a digital sampling oscilloscope to capture the power consumption after the injection. The LFI source was a diode pulse laser with 1064 nm wavelength and 8 W maximum output power. Spot size was adjusted to $15\times3.5\ \mu m$ with $20\times$ objective lens. As a DUT, we used ATmega328p microcontroller, mounted on Arduino/Genuino UNO board. DUT was optically accessible from the backside of the die, thanks to de-packaging and polishing of the backside substrate.

6.2 Determining the Fault Mask for PRESENT-80 using SCA

In the following, we will detail the process of estimating the fault mask in case of PRESENT-80, based on fault injection timing, and side-channel leakage.

There are two steps of determining the faulty nibble and the mask in round 28:

1) In order to get the information on which nibble has been faulted, we have to check the timing from the trigger. The S-Box computation on the DUT takes

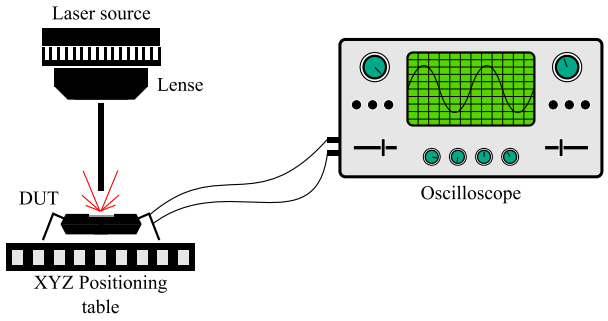

Fig. 6: Experimental setup for verifying SCADFA, including the laser fault injection device and oscilloscope for side-channel measurement.

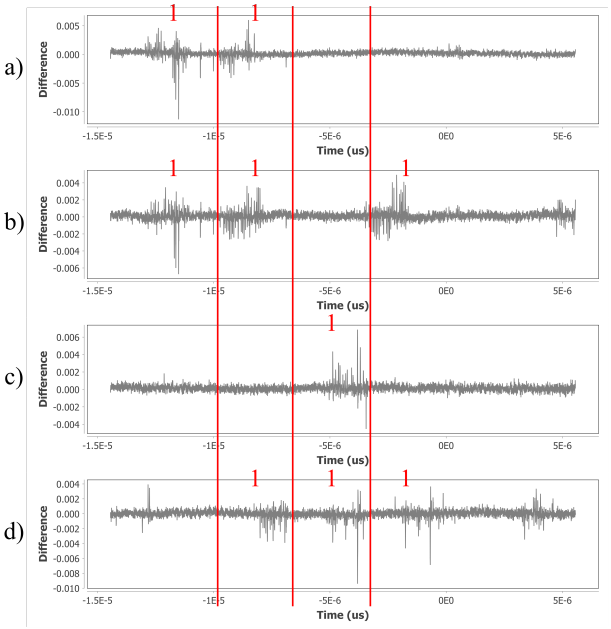

Fig. 7: Fault mask determination from SCA leakage.

$\approx 11\mu s$. During the profiling phase, the timing for processing each nibble can be estimated with 100% success rate. Nibbles are processed in a reversed order (15 to 0), therefore, for example, w.r.t defined trigger signal, nibble 14 starts with a timing offset of $2,331$ ns, while nibble 0 starts at $12,537$ ns.

2) For estimating the fault mask, we have to check the side-channel leakage in the round 29. This process is depicted in Figure 7 and the values corresponding to each power trace are detailed in Table 3. As can be easily seen, the difference trace shows the nibble position in round 29 which has a different value from the trace corresponding to non-faulty execution. The position of the changed byte can be determined by comparing traces corresponding to various differences in the initial profiling phase. Reverse engineering techniques can also be applied in this case to determine the round and nibble position. In Figure 7, the red guiding lines show the relative position of nibbles, while '1' indicates that there is a difference w.r.t. the original trace. Then, by applying a reverse bit-permutation, the output S-Box difference of round 28 can be determined.

TABLE 3: Analysis of the leakage difference patterns from Figure 7.

Trace	Offset (ns)	I/P Fault Mask:R29	O/P Fault Mask:R28
a)	4032	0000000000800080	00000000000C0000
b)	4914	0040000000400040	0000000000D00000
c)	7686	0000080000000000	0000000200000000
d)	9072	0200020002000000	0000070000000000

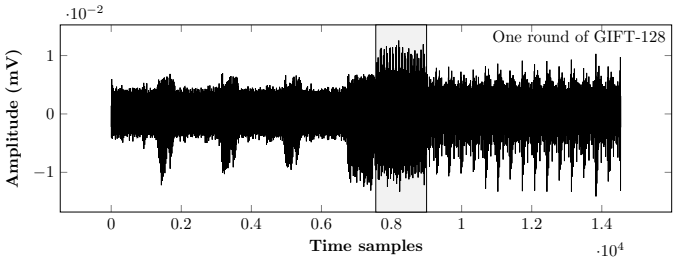

Fig. 8: Trace capturing one round of GIFT-128 with highlighted key addition part.

6.3 Determining the Fault Mask for GIFT-128 using SCA

Below, we show how to determine the mask for GIFT-128. First, a key addition part has to be identified from the trace. Figure 8 shows one round of GIFT-128, highlighting the key addition. Therefore, for the subsequent fault mask observation, we have to focus on this part in order to get the difference traces.

In Figure 9, the traces corresponding to key addition of GIFT-128 were measured. In this example, the fault will be injected at the first byte, thus affecting the first nibble of the SubCells operation at round 37 (R37). Due to the permutation layer of GIFT-128, the output bits of the first byte, i.e., {127,126,125,124}, will be mapped to bits {31,126,93,60} in the next key addition (Due to byte-wise implementation, these bits correspond to bytes 12, 0, 4, and 8 respectively). The whole propagation can be observed in Table 4. Hence, based on this permutation, the obtained mask will be rotated left by 1 bit. For example, `0xB` will be observed as `0x7`.

TABLE 4: Analysis of the leakage difference patterns from Figure 9.

	I/P Fault Mask: Key Addition R37	O/P Fault Mask: SubCells R37
a)	00000000200000010000000800000000	B0000000000000000000000000000000
b)	40000000000000001000000000000000	50000000000000000000000000000000
c)	40000000000000001000000080000000	D0000000000000000000000000000000
d)	00000000000000000000000080000000	80000000000000000000000000000000

In Figures 10 and 11, the results of the attacks for different Hamming weights are shown on both PRESENT-80 and GIFT-128, respectively. The figure denotes the number of key nibbles recovered for different fault injections (Note: for each nibble of GIFT-128, only 2 bits of the key could be recovered, and hence, by key nibble recovered, it means the faults will be affecting 2 nibbles). In this case, it can be observed that the attack works for both cases, and for GIFT-128, it actually requires fewer fault injections in general to recover a key nibble. Overall, PRESENT requires 9 -

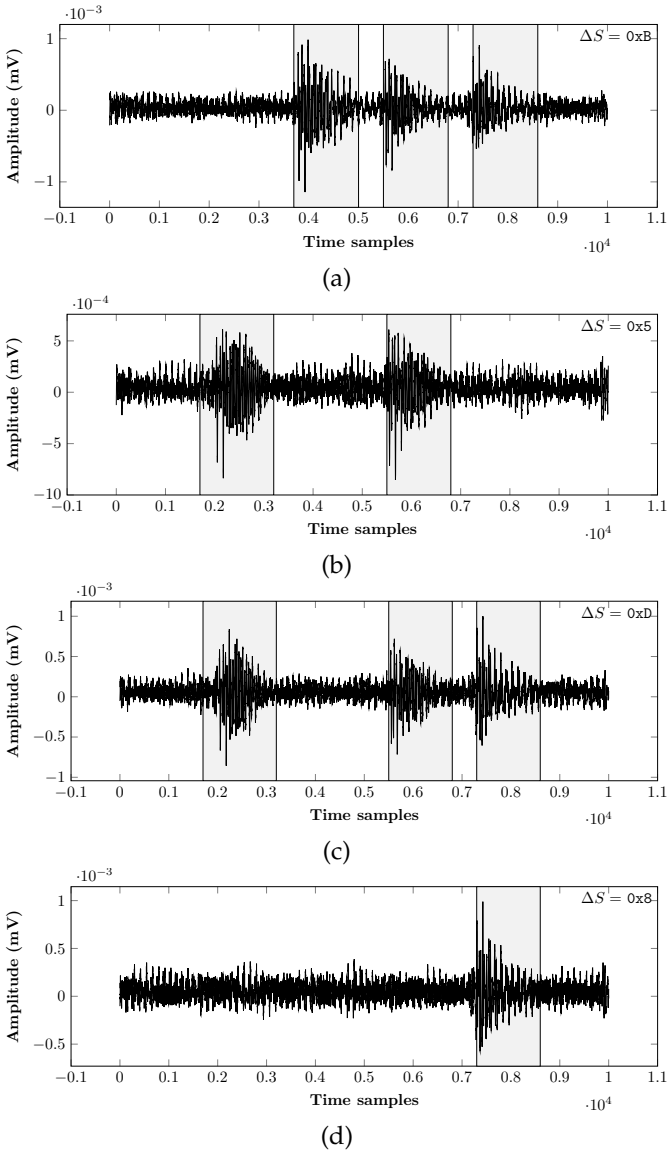

Fig. 9: Difference traces measured at the key addition of GIFT-128. Sbox input differences were (a) `0x1`, (b) `0x2`, (c) `0x3`, (d) `0x7`, resulting to output differences `0xB`, `0x5`, `0xD`, `0x8`, respectively. These were further rearranged by the permutation layer.

18 fault injections to recover the whole last round key, whereas GIFT-128 requires 6 - 9 fault injections for different Hamming weight values.

7 Further Discussions

7.1 SCADFA v/s Traditonal SCA/DFA

A pertinent question with respect to our proposed SCADFA is the following – why should an adversary resort to this attack approach if she can already launch traditional SCA or DFA attacks on unprotected implementations of the target cipher? We justify this by pointing out that SCADFA should not be considered as an improvement over SCA or DFA but as another attack in the arsenal. This is because SCADFA allows an attacker to target the internal rounds of a block cipher, where traditional SCA and DFA attacks are often too

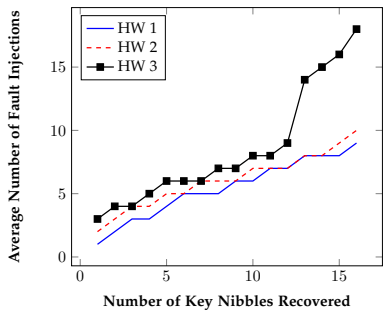

Fig. 10: Key recovery for PRESENT-80: average number of fault injections v/s number of key nibbles recovered.

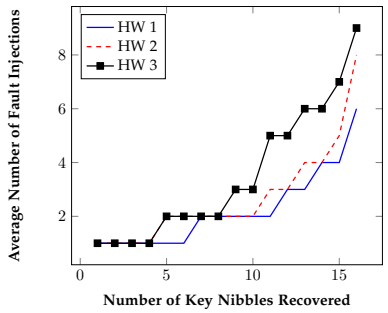

Fig. 11: Key recovery for GIFT-128: average number of fault injections v/s number of key nibbles recovered.

expensive or even practically infeasible. We expand more on this in the ensuing discussions.

Traditionally, SCA techniques target the first and last rounds of the target block cipher. This is because targeting the internal rounds significantly blows up the number of leakage traces/samples required as well as the cost of key-recovery. The increase in attack complexity may be attributed to the diffusion characteristics of the cipher, which ensures that as we trace back the encryption path from the ciphertext to the target internal round, more and more sub-keys (key-bytes or key-nibbles) get involved. This often requires solving a very complex system of equations, thereby blowing up the cost of the attack. A similar scenario is encountered when attempting to launch a DFA attack on an internal cipher round.

For the aforementioned reasons, designers often claim that in applications targeting resource-constrained devices, a reasonable security v/s efficiency trade-off is to only protect the first and last rounds of the target block cipher implementation against SCA and DFA attacks. However, our SCADFA attack bypasses such protection strategies by allowing the adversary to inject faults in rounds $r-3$ and $r-2$, and then recover the fault mask using very few side-channel traces. The overall attack also requires few fault injections. Rather than being adversely affected by the diffusion characteristics of the block cipher, our attack strategy actually exploits it to attacker's advantage. In this sense, SCADFA can provide even more effective attack strategy than either SCA or DFA in isolation, especially against existing implementations that are only partially fortified with countermeasure strategies in the first and last rounds.

7.2 Optimal v/s Non-Optimal Diffusion

We note that our combined SCA+DFA attack strategy depends on the nature of the diffusion in the following sense: if the diffusion is *optimal*, then a fault injected in round r is guaranteed to spread into every nibble of the cipher state by round $(r+3)$ (in case the state-size is 64 bits) or round $(r+4)$ (in case the state-size is 128 bits), assuming 4×4 S-Boxes. In general, for (n, m)-BPOD block cipher (as per the general characterization of bit permutation-based block ciphers in Section III.A), the fault injected in round r spreads to every nibble of the cipher state by round $(r + \lceil \log_n n^2.m \rceil)$.

We first enumerate the advantages of optimal diffusion from an attacker's point of view:

1) From an attack efficiency point of view, it is desirable that the fault spreads to as many state nibbles as possible, since this allows the attacker to recover the maximum possible key nibbles with only few fault injections. Optimal diffusion guarantees this.
2) Moreover, the aforementioned fault diffusion behavior allows the attacker to target *any* nibble when injecting the fault in the target round, since the fault is guaranteed to spread across the whole state within a pre-determined number of rounds. This makes the attack easy to mount on a variety of target devices.

However, we do not claim that the attack is impossible when the target block cipher has non-optimal diffusion characteristics. In such a case, depending on the fault location, the number of nibbles to which the fault diffuses in subsequent rounds may vary. This means that in order to recover all key nibbles, the attacker would need to inject a potentially larger number of faults.

In summary, our attack strategy is general enough to cover block ciphers with both optimal and non-optimal diffusion characteristics. However, all existing bit-permutation-based block ciphers have optimal diffusion characteristics, and our proposed attack is more efficient on such block ciphers.

7.3 Mounting SCADFA on 32-bit MCUs

In Section 6, we have demonstrated SCADFA on an 8-bit MCU, where a single nibble is rewired at a time. A pertinent question is to ask if the attack can be ported to other platforms, such as 32-bit MCUs. This scenario is possible, but the attack strategy needs to be adjusted accordingly. Please note that this strategy also requires that an observation of 1-bit change on a 32-bit word is possible.

We begin by noting that in a 32-bit MCU, as many as 8 nibbles can be rewired in parallel. This implies that precise identification of the fault mask can be extremely costly. More specifically, such precision would only be realistic with template attacks, where a separate template would have to be created for each of the 2^{32} possible fault masks. However creating 2^{32} templates requires non-trivial effort and would seem to go against the fundamental purpose of SCADFA, which is making an attack more efficient compared to SCA/DFA attack alone.

As it turns out, precise fault mask identification is not necessary when launching SCADFA on 32-bit MCUs. In particular, the properties of the underlying BPOD block ciphers, such as PRESENT-64 and GIFT-128, allow for alternative fault mask identification strategies. The strategies are different for PRESENT-80 and GIFT-128 because of their different state sizes.

In case of PRESENT-80, the cipher state fits into two 32-bit words. Therefore, from SCA observation, the attacker only gains the knowledge of 4 distinct cases: both words changed; none of the words changed; either the first or the second word changed [28]. The idea here is to repeat the fault until the fault mask affects either the first half or the second half of the nibble, i.e. we either get $1000, 0100, 1100$, or $0010, 0001, 0011$ as a fault mask. This can be easily determined by observing which of the two words changed in the following round. From here, the attack continues as described in the previous sections, with the difference that instead of considering a single fault mask value, the attacker always needs to consider three scenarios, since it cannot be determined which of the two bits (or both) were affected. We conducted some additional experiments which showed that the attack targeting implementations of PRESENT-80 on 32-bit MCUs requires 15-20 fault injections to recover the last round key.

In case of GIFT-128, the cipher state fits into four 32-bit words. Thanks to BPOD, each output bit of an Sbox goes to a different word. In practice, this means that by observing a change in one of the four words, the attacker can precisely determine the fault mask, allowing the exactly same attack procedure as on 8-bit architectures. This was confirmed by conducting some additional experiments, which showed that the attack targeting implementations of GIFT-128 on 32-bit MCUs requires 6-9 fault injections to recover the last round key, which is the same as on the 8-bit MCUs.

7.4 Difference Determination after Propagation

A natural question to arise would be whether it is possible to trace the difference in further rounds after it spreads. Figure 12 captures the last three rounds of PRESENT-80, after injecting the difference in round 28. In round 29, the difference can be clearly determined, with two peaks corresponding to ones. However, as the differential propagates further, it scrambles the entire cipher state, making it infeasible to distinguish the difference from a simple observation. While it might be possible with statistical side-channel analysis methods, the advantage of simplicity of such combined attack would be gone.

7.5 Applicability of SCADFA on Other Ciphers

It is interesting to see that the combined attack methodology, in its current for, would fail against ciphers such as AES, LED and Midori that use mix column layers based on MDS or Almost-MDS. The diffusion characteristics of an MDS or Almost-MDS layer break any correlation between the output fault mask of a prior round and the eventual input fault mask at a later round, by causing the fault to always diffuse to the maximum possible number of nibbles in each round. With the same argument, we claim that the attack as it is would not work against Feistel based structures as well.

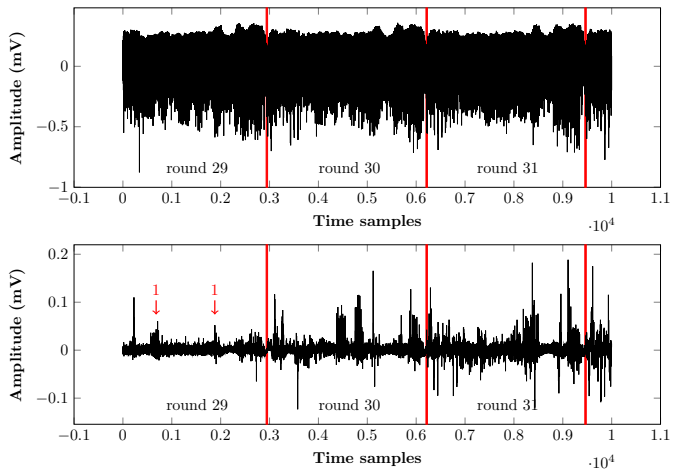

Fig. 12: Traces capturing the last three rounds of PRESENT-80 after the fault injection. Upper trace shows the power consumption traces, lower trace shows the difference. While at round 29 it is easy to spot the difference, later the propagation spreads to almost the whole cipher state.

However, knowledge of the fault mask might be useful in reduction of the time complexity for existing fault attacks on these ciphers or open up other avenues for analysis.

7.6 Possible Countermeasures

We discuss potential countermeasures and their effectiveness in resisting our combined attack methodology on bit-permutation based SPN block ciphers. Standard fault detection mechanisms such as spatial and temporal redundancy could potentially increase the number of fault injections required; but they can be bypassed using biased fault injection techniques [29] that allow injecting the same fault in both the original and redundant computations with high probability. The other way to prevent SCADFA is to hide the side-channel signature. Naturally, there are several side-channel countermeasures which can be used to prevent an attacker from deriving the precise fault mask from captured side-channel traces. The most commonly used side-channel countermeasures which can be applied against SCADFA as well are:

- **Noise generators:** These countermeasures work by adding artificial noise to the captured side-channel measurement in order to reduce the effective signal to noise ratio (SNR) of the side-channel trace. Use of clock jitter or parallel uncorrelated computation are the straightforward methods [30]. Shuffling, which involves randomly varying the sequence of internal operations, breaks the synchronization of measured data, resulting in reduced SNR [31]. The simple nature of these countermeasures also leads to low overheads.
- **Data hiding:** Data hiding [32] aims at balancing side-channel signature independent of the data. This is achieved by replacing each signal x with a pair $(x, \overline{x})$, where $\overline{x}$ is the logical inverse of x. Similarly, logical operation G is replaced by pair $(G, \overline{G})$. Since every activity in x and computation by G is compensated by $\overline{x}$ and $\overline{G}$ respectively, the overall side-channel activity is constant, which comes at about twice the overhead. Some process and circuit mismatches can still reveal little information but the SNR remains low.
- **Data masking:** Masking [33] on the other hand relies on randomness to hide side-channel signature. Signal x is mixed with a random r to derive masked x_r. For instance in Boolean masking $x_r = x \oplus r$. The computations are performed on x_r, where r is randomly chosen for every run of the computation, thus totally disassociating side-channel activity from signal x. Higher order attacks can still be applied. Masking also has at least $2\times$ overhead.

One or a combination of these countermeasures can be incorporated to increase the complexity of retrieving the fault mask. In some cases it can be done in a way that the retrieval might be practically impossible.

Another possibility is to implement the target cipher on a microcontroller platform with inherent protections against side-channel and fault analysis attacks. However, with respect to experimentation on inherently protected microcontroller platforms, we faced the following difficulties. First, no such platform was readily available to us for public research. Secondly, our attack uses techniques related to both SCA and DFA, and it is difficult to find a single platform with inherent resistance against both forms of attacks. We agree that this is an interesting direction for future exploration.

8 Conclusion

In this paper, we have presented a combined side-channel and fault attack methodology that can be used against bit permutation based block ciphers. Generalization of such methodology to bit permutations with optimal diffusion allowed us to design efficient attacks on PRESENT and GIFT. We have supported our results with simulations and experimental evaluation, showing the practicality of our work.

In the future, it would be interesting to look into combined attacks on different diffusion functions as well as on implementations with various SCA countermeasures.

References

[1] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: an ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, 2007, pp. 450–466.

[2] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede, "RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms," *SCIENCE CHINA Information Sciences*, vol. 58, no. 12, pp. 1–15, 2015.

[3] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: a small present towards reaching the limit of lightweight encryption," to appear in Cryptographic Hardware and Embedded Systems - CHES 2017, 19th International Conference, Taipei, Taiwan, September 25-28, 2017, 2017.

[4] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Proceedings of CRYPTO'99*, ser. LNCS, vol. 1666. Springer-Verlag, 1999, pp. 388–397.

[5] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, "A practical implementation of the timing attack," in *Smart Card Research and Applications*. Springer, 1998, pp. 167–182.

[6] E. De Mulder, P. Buysschaert, S. B. Örs, P. Delmotte, B. Preneel, G. Vandenbosch, and I. Verbauwhede, "Electromagnetic analysis attack on an fpga implementation of an elliptic curve cryptosystem," in *Computer as a Tool, 2005. EUROCON 2005. The International Conference on*, vol. 2. IEEE, 2005, pp. 1879–1882.

[7] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, 2002, pp. 29–45.

[8] J. Blömer and J. Seifert, "Fault based cryptanalysis of the advanced encryption standard," *IACR Cryptology ePrint Archive*, vol. 2002, p. 75, 2002. [Online]. Available: http://eprint.iacr.org/2002/075

[9] P. Dusart, G. Letourneux, and O. Vivolo, "Differential fault analysis on a.e.s." in *ACNS*, ser. Lecture Notes in Computer Science, J. Zhou, M. Yung, and Y. Han, Eds., vol. 2846. Springer, 2003, pp. 293–306. [Online]. Available: http://dblp.uni-trier.de/db/conf/acns/acns2003.html#DusartLV03

[10] G. Piret and J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and KHAZAD," in *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, 2003, pp. 77–88. [Online]. Available: https://doi.org/10.1007/978-3-540-45238-6_7

[11] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," in *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*, 2011, pp. 224–233. [Online]. Available: https://doi.org/10.1007/978-3-642-21040-2_15

[12] G. Wang and S. Wang, "Differential Fault Analysis on PRESENT Key Schedule," in *Computational Intelligence and Security (CIS), 2010 International Conference on*, Dec 2010, pp. 362–366.

[13] X. Zhao, S. Guo, T. Wang, F. Zhang, and Z. Shi, "Fault-propagate pattern based DFA on PRESENT and PRINT cipher," *Wuhan University Journal of Natural Sciences*, vol. 17, no. 6, pp. 485–493, 2012.

[14] N. Bagheri, R. Ebrahimpour, and N. Ghaedi, "New differential fault analysis on PRESENT," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, no. 1, pp. 1–10, 2013.

[15] J. Breier and W. He, "Multiple fault attack on present with a hardware trojan implementation in fpga," in *2015 International Workshop on Secure Internet of Things (SIoT)*, Sept 2015, pp. 58–64.

[16] F. De Santis, O. M. Guillen, E. Sakic, and G. Sigl, "Ciphertext-only fault attacks on present," in *Lightweight Cryptography for Security and Privacy: Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014*, T. Eisenbarth and E. Öztürk, Eds., pp. 85–108.

[17] N. F. Ghalaty, B. Yuce, and P. Schaumont, "Differential fault intensity analysis on present and led block ciphers," in *Constructive Side-Channel Analysis and Secure Design: 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015.*, pp. 174–188.

[18] B. Robisson and P. Manet, "Differential behavioral analysis," in *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, 2007, pp. 413–426.

[19] C. Clavier, B. Feix, G. Gagnerot, and M. Roussellet, "Passive and active combined attacks on aes combining fault attacks and side channel analysis," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2010 Workshop on*. IEEE, 2010, pp. 10–19.

[20] T. Roche, V. Lomné, and K. Khalfallah, "Combined fault and side-channel attack on protected implementations of aes," in *Smart Card Research and Advanced Applications*. Springer, 2011, pp. 65–83.

[21] F. Dassance and A. Venelli, "Combined fault and side-channel attacks on the aes key schedule," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*. IEEE, 2012, pp. 63–71.

[22] A. Moradi, O. Mischke, C. Paar, Y. Li, K. Ohta, and K. Sakiyama, "On the power of fault sensitivity analysis and collision side-channel attacks in a combined setting," *Cryptographic Hardware and Embedded Systems–CHES 2011*, p. 292.
Side-Channel Analysis and Secure Design. Springer, 2013, pp. 137–153.

[24] M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria, "When Clocks Fail: On Critical Paths and Clock Faults," *Smart Card Research and Advanced Application*, pp. 182–193, 2010.

[25] A. Barenghi, G. M. Bertoni, L. Breveglieri, and G. Pelosi, "A fault induction technique based on voltage underfeeding with application to attacks against aes and rsa," *Journal of Systems and Software*, vol. 86, no. 7, pp. 1864–1878, 2013.

[26] A. Dehbaoui, J.-M. Dutertre, B. Robisson, and A. Tria, "Electromagnetic transient faults injection on a hardware and a software implementations of aes," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*. IEEE, 2012, pp. 7–15.

[27] E. Trichina and R. Korkikyan, "Multi fault laser attacks on protected CRT-RSA," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2010 Workshop on*. IEEE, 2010, pp. 75–86.

[28] J. Breier, D. Jap, X. Hou, and S. Bhasin, "On side-channel vulnerabilities of bit permutations: Key recovery and reverse engineering," Cryptology ePrint Archive, Report 2018/219, 2018, https://eprint.iacr.org/2018/219.

[29] S. Patranabis, A. Chakraborty, P. H. Nguyen, and D. Mukhopadhyay, "A Biased Fault Attack on the Time Redundancy Countermeasure for AES," in *Constructive Side-Channel Analysis and Secure Design*. Springer, 2015, pp. 189–203.

[30] T. Güneysu and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 33–48.

[31] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F.-X. Standaert, "Shuffling against side-channel attacks: A comprehensive study with cautionary note," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2012, pp. 740–757.

[32] K. Tiri and I. Verbauwhede, "Secure logic synthesis," in *International Conference on Field Programmable Logic and Applications*. Springer, 2004, pp. 1052–1056.

[33] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 142–159.

[23] Y. Li, S. Endo, N. Debande, N. Homma, T. Aoki, T.-H. Le, J.-L. Danger, K. Ohta, and K. Sakiyama, "Exploring the relations between fault sensitivity and power consumption," in *Constructive*