One Plus One is More than Two: A Practical Combination of Power and Fault Analysis Attacks on PRESENT and PRESENT-like Block Ciphers

Sikhar Patranabis, Debdeep Mukhopadhyay
Department of CSE, IIT Kharagpur, India
sikhar.patranabis@iitkgp.ac.in, debdeep@cse.iitkgp.ernet.in

Jakub Breier, Shivam Bhasin
Temasek Labs, NTU Singapore
jbreier@ntu.edu.sg, sbhasin@ntu.edu.sg

***Abstract*—We present the first practically realizable side-channel assisted fault attack on PRESENT, that can retrieve the last round key efficiently using single 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 an attack model via side-channel analysis experiments on top of a laser-based fault injection setup, targeting a PRESENT-80 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 key nibbles independently and in parallel. Both analytically and through experimental evidence, we show that the combined attack can recover the last round key of PRESENT with 4 random nibble fault injections in the best case, and around 7-8 nibble fault injections in the average case. Our attack sheds light on a hitherto unexplored vulnerability of PRESENT and PRESENT-like block ciphers that use bit-permutations instead of maximum distance separable (MDS) layers for diffusion.**

***Keywords*-DFA, DPA, PRESENT, combined attacks, fault attacks, side-channel analysis, bit-permutation**

I. INTRODUCTION

The advent of pervasive devices for communication and information systems such as the Internet of Things (IoT) has led to an increased demand for embedded security solutions with a good balance of performance and security. In particular, IoT applications have motivated a whole branch of cryptographic design, denoted as *lightweight cryptography*. The aim of lightweight cryptography is to design security primitives, such as block ciphers, that have low area footprint, reasonable throughput, low energy consumption, and provide adequate security guarantees against classical cryptanalytic attacks. While design and analysis of block ciphers requires careful consideration, real-world security also mandates secure implementations for the same. In particular, such implementations must resist side-channel attacks (SCA) as well as fault attacks (FA).

One of the most popular lightweight block ciphers is PRESENT [1]. The design of PRESENT has inspired a whole new family of 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. Several new proposals for lightweight block ciphers such as Rectangle [2] and GIFT [3] have opted for bit-permutations for efficiency in hardware. In this paper, we propose a novel side-channel assisted fault analysis of PRESENT and PRESENT-like ciphers that exposes an interesting vulnerability of bit-permutations. In particular, the attack is based on the principle that for bit-permutation based ciphers, the knowledge of the output fault mask in an earlier round could be exploited to significantly reduce the entropy of the input fault mask at a later round.

The main contributions of this paper are as follows:

1) The paper proposes the first practically realizable combination of side-channel analysis (SCA) and differential fault analysis (DFA) on PRESENT. The attack uses a relaxed fault model corresponding to a given target round, and assumes that the adversary uses side-channel leakage to precisely determine the resulting fault mask. Prior attacks on PRESENT have mostly been limited to either SCA or FA, but have not exploited the combined potential of both. 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 corroborate the theoretical analysis with a real-world demonstration of the combined attack on an ATmega328P microcontroller-based implementation of PRESENT-80 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 only 4 fault injections in the best case, and 7-8 fault injections in the average case. This makes our attack one of the most efficient to be proposed on PRESENT and PRESENT-like block ciphers.

II. PRELIMINARIES

A. Overview of the PRESENT Block Cipher

PRESENT is based on a substitution-permutation network (SPN). It consists of 31 rounds, block length is 64 bits and

Table I: 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

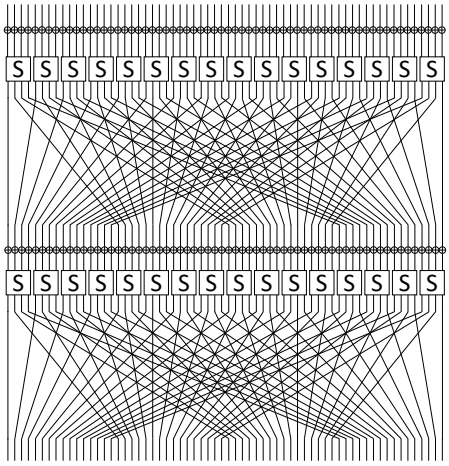

Figure 1: Structure of two rounds of PRESENT

it supports keys with lengths of 80 and 128 bits. In this paper, we focus on the 80 bit key length version, which we denote as PRESENT-80, however the attack applies on 128-bit as well. Each round consists of three operation layers: an XOR-layer with the round key, a substitution layer using 16 identical 4×4 S-Box (Table I) and a bit-permutation layer (Figure 1). 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-80 comprises of a rotation, S-Box look-up and round counter addition, thus making it invertible.

B. Related Work

The first DFA on PRESENT, published by G. Wang and S. Wang [10] required 64 pairs of correct and faulty ciphertext on average, with a computational complexity of 2^{29}. Later, Zhao et al. [11] utilized a fault-propagation pattern-based DFA, targeting PRESENT and PRINT. The attack on PRESENT-80/128 required 8/16 ciphertext pairs on average, reducing the key search space to $2^{14.7}/2^{21.1}$ in average. Bagheri et al. [12] showed attacks utilizing single bit-flip and single nibble fault models. The first attack obtains the last subkey with 48 ciphertexts pairs, while the second attack reveals the key with 18 ciphertext pairs on average. Breier and He [13] proposed a multiple fault attack, targeting four nibbles at once, being able to recover the secret key in 2 encryptions. DeSantis et al. [14] presented a ciphertext-only attack, which requires only two ciphertext pairs in the best case. Ghalaty et al. [15] 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.

Table II: Summary of Combined SCA and FA on Block Ciphers

Reference	Setting	Target	Combination
Robisson et al [4]	Simulated	Non-masked AES	DPA + Stuck-At
Clavier et al. [5]	Simulated	Masked AES	CPA + Instruction-skip
Roche et al. [6]	Simulated	Key Schedule of Masked AES	CPA + Stuck-At or Byte
Dassance et al. [7]	Simulated	Key Schedule of Masked AES	CPA + Byte
Moradi et al. [8]	Practical	Unprotected + Protected AES	CCA + FSA
Li et al. [9]	Practical	Unprotected AES	DPA + FSA

A summary of state-of-the art combined SCA and FA attacks on block ciphers is presented in Table II. Differential behavioral analysis (DBA [4]) is a combined SCA with safe-error attacks. A combined SCA and FA targeting first AES key addition was proposed in [5]. Roche et al. proposed a DFA on AES key schedule in [6] by injecting faults in pen-ultimate round key computation, further improved in [7]. All these attack were demonstrated in simulated settings. Combinations of fault sensitivity analysis (FSA) with collision correlation attack (CCA) [8] and CPA were also proposed. [9]. To the best of ours knowledge, no previous work demonstrates a practical demonstration of combining DFA with SCA on PRESENT.

III. The Proposed Combined SCA and DFA of PRESENT

In this section, we propose a combined SCA and DFA of PRESENT. We now present a detailed description of the fault attack.

A. Properties of the PRESENT Block Cipher

We begin with a description of the properties of PRESENT that are exploited in our attack. We would like to point out that these properties are due to the bit-permutation layer of PRESENT, and are also observed in other PRESENT-like block ciphers that use bit-permutations as opposed to MDS layers for diffusion.

- The input of an S-Box in round r comprises output bits from four different S-Boxes in round $r-1$.
- The output of an S-Box in round r is distributed across the inputs of four different S-Boxes in round $r+1$.
- The output of the S-Box group $[4n, 4n+3]$ in round r entirely constitutes the input for the S-Box group $\{n, n+4, n+8, n+12\}$ in round $r+1$, where $n \in \{0,1,2,3\}$. More precisely, for $n,d,l \in \{0,1,2,3\}$, the l^{th} bit in the output of S-Box $4n+d$ in any round is precisely the d^{th} bit in the input of S-Box $n+4l$ in the next round, albeit after XOR-ing with the corresponding round key bit. As stated before, visualization of this propagation is depicted in Figure 1.

B. Fault Model and Fault Location

Our attack assumes random nibble fault model. The fault is injected at the input of round 28 during a PRESENT

encryption operation. Nibble faults have been demonstrated to be practically achievable using traditional fault injection techniques such as clock and voltage glitches [16], [17] as well as more advanced injections methods such as electromagnetic (EM) pulse or laser pulse injection [18], [19]. We use the term *output fault mask* to denote the differential Δ_{out} of the correct and faulty nibble at output of S-Box operation in round 28. We would like to point out that recent attacks on PRESENT, such as that presented by Breier et al. in [13], 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.

C. The Role of Side-Channel Analysis in Our Attack

The role of the side-channel leakage in our analysis is to deterministically obtain the output fault mask Δ_{out} corresponding to round 28. Note that since our attack assumes a random fault in a single nibble, the output fault mask Δ_{out} is not known. Instead, we use side-channel analysis to determine Δ_{out}. The basic principle is as follows: assuming the fault is injected in a particular nibble during round r (round 28 in our attack on PRESENT), we collect the leakage traces corresponding to the fault-free and faulty computations in round $r+1$ (round 29 in our attack). By computing the difference in side-channel measurements of correct and faulty execution, the bit-permutation pattern makes it possible to determine the exact value of Δ_{out}. We practically demonstrate the recovery of Δ_{out}, when injecting faults using a laser fault injection a 8-bit micro-controller platform. The fault was injected during the S-Box lookup corresponding to the target nibble in round 28 of PRESENT encryption. The fault injection timings were profiled with respect to each nibble operation, allowing a 100% repeatability in corrupting a target nibble. This was followed by a differential analysis between the leakage traces corresponding to the fault-free and faulty computations to retrieve the output fault mask. Detailed experimental results corresponding to the SCA are presented in Section IV.

D. The Fault Propagation Characteristics

We now present the fault propagation characteristics corresponding to our attack in details. As already mentioned, the attack targets a single nibble in round 28 of PRESENT. We begin by stating the following theorems.

Theorem 1: Suppose the Hamming Weight of the output fault mask of the target nibble in round 28 of PRESENT is x, where $x \in \{0,1,\cdots,4\}$. Then, the Hamming Weight of the input fault mask of any nibble in round 31 is at most x.

Theorem 2: Suppose the Hamming Weight of the output fault mask of the target nibble in round 28 of PRESENT is x, where $x \in \{0,1,\cdots,4\}$. Then, the input fault mask of any nibble in round 31 takes at most 2^x values.

We first present a concrete example to support the above theorems, and then prove them for generalized instances.

*1) **A Concrete Example**:* 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). Once again, 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. Figure 2 illustrates the fault propagation characteristics for the above example.

*2) **The Generalized Proof**:* The above example provide an intuitive explanation for Theorems 1 and 2. We now present a generalized and formal proof for the same. We present the fault mask characteristics in each of rounds 28, 29, 30 and 31 separately.

- **Round 28:** Suppose the adversary injects a fault in nibble $4n+d$, where $n,d \in \{0,1,2,3\}$, and suppose the output fault mask has Hamming weight $x \in \{0,1,\cdots,4\}$. In particular, let $l_1,\cdots,l_x \in \{0,1,2,3\}$ be the bits in the output fault mask that are set to 1.
- **Round 29:** Consider the effects of the bit $l_1,\cdots,l_x$ in the output fault mask corresponding to nibble $4n+d$ in round 28. As per the generic properties of the diffusion layer of PRESENT discussed in Section III-A, these faulty bits will propagate to the nibbles $n+4l_1,\cdots,n+4l_x$ respectively, creating an input fault mask of Hamming Weight 1.

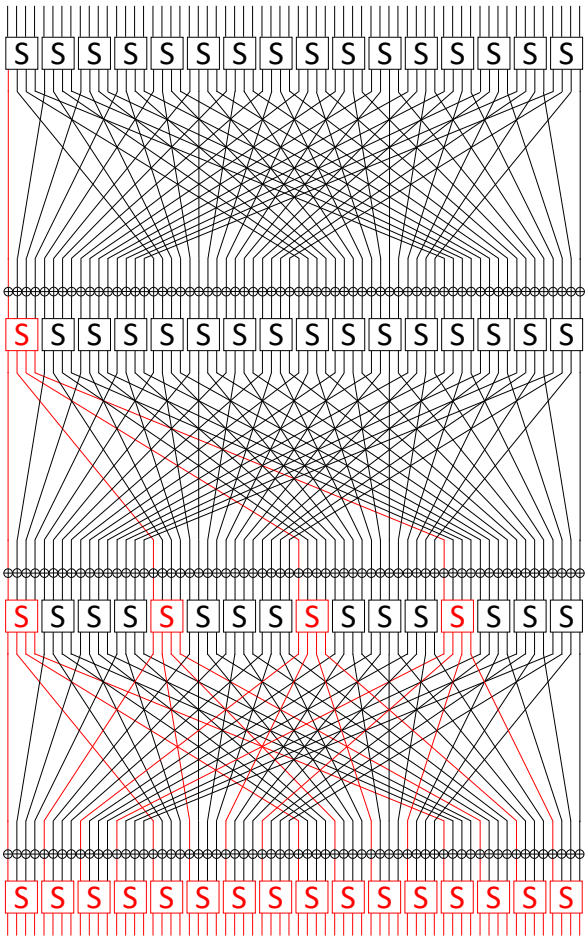

Figure 2: Fault propagation for the output fault mask 0001

- **Round 30:** We now focus on a specific faulty nibble, say nibble $n+4l_1$, in round 29. Once again, as per the generic properties of the diffusion layer of PRESENT discussed in Section III-A, the output of this faulty nibble will potentially propagate to the n^{th} input bit of the nibbles l_1, l_1+4, l_1+8 and l_1+12 in round 30. Similarly, the output of the faulty nibble $n+l_x$ in round 29 will potentially affect the nibbles l_x, l_x+4, l_x+8 and l_x+12. *It is important to observe that for each of $l_1, \cdots, l_x$, the set of potentially faulty nibbles in round 30 is disjoint.*
- **Round 31:** Finally, we examine the fault propagation to the input of round 31. Consider the faulty quartet of nibbles l_1, l_1+4, l_1+8 and l_1+12 in round 30. The nibble l_1 will potentially spread the fault to the l_1^{th} input bit of the nibbles $0, 4, 8$ and 12 in round 31. Similarly, the nibble l_1+4 will potentially spread the fault to the l_1^{th} input bit of the nibbles $1, 5, 9$ and 13 in round 31. In general, the faulty nibble $l_{k_1}+4k_2$, where $k_1 \in \{1, \cdots, x\}$ and $k_2 \in \{0, 1, 2, 3\}$, will potentially spread the fault to the $l_{k_1}^{\text{th}}$ input bit of the nibbles k_2, k_2+4, k_2+8 and k_2+12 in round 31. This, in turn, implies each nibble in round 31 can receive a faulty input bit from at most x faulty nibbles in round 30.

Thus, each nibble in round 31 has an input fault mask of Hamming Weight at most x. Additionally, since exactly x bits of each input fault mask are 1, and the faulty bits are determined by the values of $l_1, \cdots, l_x$, each input fault mask in round 31 can take 2^x values. This completes the proof of Theorems 1 and 2.

E. The Key Recovery Process

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 PRESENT, where we ignore the bit-permutation operation of round 31. 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:

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

where S^{-1} denotes the inverse S-Box operation. From the differential uniformity of the PRESENT S-Box, the expected number of values of K satisfying the above equation is one. Now, assuming a non-zero output fault mask 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 (this is proven using Theorems 1 and 2), 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

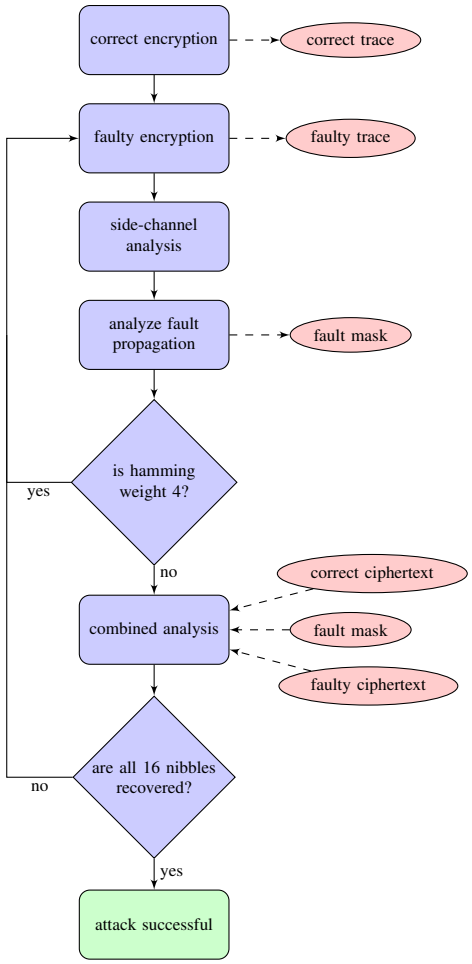

Figure 3: Execution steps for Proposed Combined Attack

key. The attack efficiency is also supported by experimental results in Section IV.

The overall attack flow is summarized in Figure 3. Note that the attack can also be repeated in round 27 to additionally recover 64 bits of the penultimate round key, along with 64 bits of the last round key. The combination of two recovered keys can then be used to recover the entire last round key of PRESENT.

IV. Experimental Results

A. The Combined SCA+FA Setup

The setup for our combined SCA and DFA-based attack is depicted in Figure 4. The core of the setup consists of near-infrared diode pulse laser (1064 nm wavelength) with the maximum output power of 20 W. This power is further reduced to $\approx$ 8 W with usage of 20$\times$ objective lens that scales the effective spot size to $15 \times 3.5 \mu m$. The laser activation length was set to 150 ns and the laser power was set to 3%, resulting to $\approx$ 0.24 W.

As the device under test (DUT), we used ATmega328P microcontroller, decapsulated from the back-side and mounted on Arduino UNO development board. The area

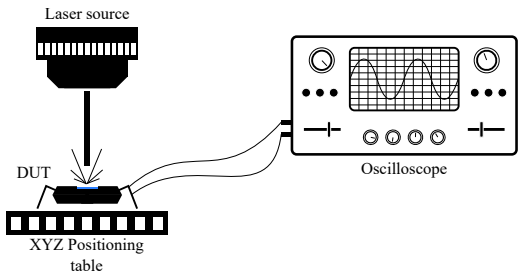

Figure 4: Experimental setup for the SCA-aided DFA procedure.

Table III: Analysis of the leakage difference patterns from Figure 5.

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

of the chip is 3×3 mm^2, while the sensitive area covers $\approx$ 0.05% of the whole chip size. The implementation of PRESENT we used computes the addRoundKey byte-wise and S-Box nibble-wise, therefore, we had to take this into account during the fault model estimation. The attack has to be likewise adjusted if the S-Box is computed byte-wise.

To control the impact location, we used XYZ positioning table with the spatial precision of 0.05 μm. Timing precision is achieved by inserting the trigger at the start of round 28.

For the side-channel leakage measurement, we used a digital oscilloscope, capturing the time frame of one round after the fault was injected. In order to distinguish the fault mask, we first profiled the standard power consumption by calculating an average of 100 encryptions. As the repeatability of the fault was 100%, another 100 experiments were done while injecting the fault at the same position and same timing and again, averaging these traces. Afterwards, we calculated a difference of these two traces and it gave us the knowledge of the injected fault mask in the previous round.

We would like to point out that both the triggering for fault injection and the averaging of the power traces are optional. By observing the side-channel trace in our case, it was straightforward to pin-point beginning and ending of each round and each operation (S-BoxLayer, pLayer, addRoundKey) had a unique side-channel signature (this can be seen in the upper part of Figure 8). Similarly, the fault mask was recoverable from single trace. Indeed, if the repeatability of the fault is low, averaging will be difficult.

B. Determination of Fault Mask

In the following, we will detail the process of estimating the fault mask, based on laser fault injection parameters, and side-channel leakage.

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

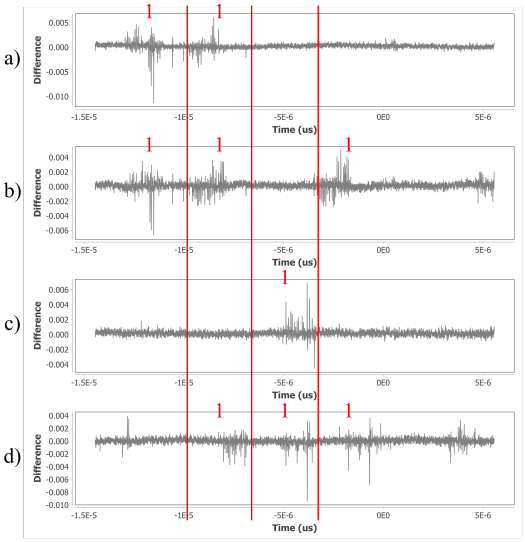

Figure 5: Fault mask determination from SCA leakage.

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 $\approx 11\mu s$. 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 5 and the values corresponding to each power trace are detailed in Table III. 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. This position can be easily determined from the profiling phase and comparing different traces. Reverse engineering techniques can also be applied in this case to determine the round and nibble position. In Figure 5, 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.

C. Key Recovery: Performance and Efficiency

We now present results illustrating the number of necessary fault injections to recover the last round key of PRESENT using the faults. Our SCA experiments demonstrated that nearly all the output fault masks obtained upon fault injection had Hamming weights of 1, 2 and 3, which is precisely the useful class of faults for our attack. Figure 6 presents a comparison of the estimated number of fault

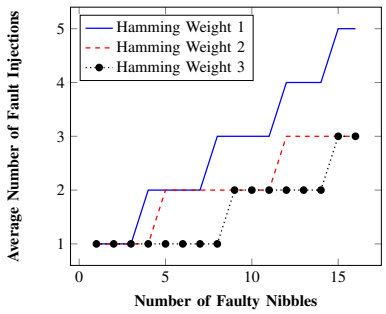

Figure 6: Fault Propagation: Average Number of Fault Injections v/s Number of Faulty Nibbles

injections required under each kind of fault mask for fault propagation to a given number of nibbles in the ciphertext. Recall that the DFA can recover a given key nibble only if the fault propagates to that nibble in the last round. Hence, the efficiency of the attack depends on the average number of fault injections required for the fault to propagate to each nibble. Quite intuitively, greater the Hamming weight of the output fault mask in the target round, the faster the fault propagates to a larger number of nibbles in the subsequent rounds. This is also reflected in our experimental results. However, a higher Hamming weight of the fault mask also reduces exploitation potential due to enhanced fault propagation in rounds 29 and 30, as illustrated next.

We now present the average number of fault injections required to recover a given number of nibbles of the last round key for each category of fault mask in Figure 7. Quite evidently, key recovery for fault masks with Hamming weights 1 and 2 requires a significantly lesser number of fault injections than with Hamming weight 3. This is in accordance with the theoretical estimate for the required number of fault injections in Section III-E. On average, a combination of fault masks with Hamming Weights 1 and 2 recovers 64 bits of the key in **7-8 fault injections**, which is at par with the best known fault attack on PRESENT, while the best case scenario allows recovering 64 bits of the key with 4 **fault injections**. The best case scenario is typically encountered when using fault masks of Hamming weight 1, and the fault diffuses to all nibbles of the final round in each injection. The worst case scenario, on the other hand, is encountered with fault masks of Hamming weight 3, when 19 fault injections are found to be necessary for key recovery.

D. Applicability to Hardware Implementations

A final point to note is that although we experimentally demonstrate our attack on a software implementation of PRESENT, the attack is also applicable on hardware (HW) implementations. Low-cost nibble HW implementations will have the same vulnerability. Parallel implementations will be harder to exploit, nevertheless, localized measurements can

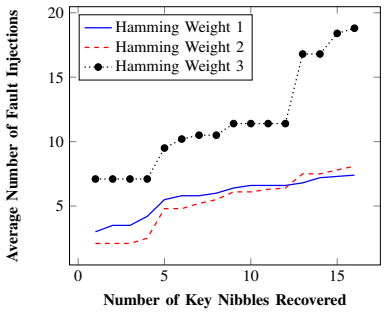

Figure 7: Key Recovery: Average Number of Fault Injections v/s Number of Key Nibbles Recovered

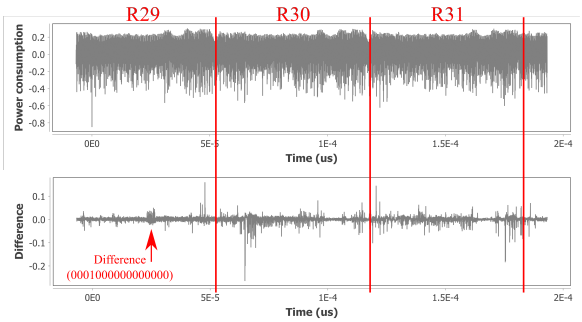

Figure 8: Power trace for the last three rounds with a corresponding differential trace.

be used to recover the fault mask and perform the attack.

V. Discussion

A. Extension to Other Rounds

While it is relatively easy to determine the faulty nibbles in round 29, this process becomes harder once the propagation of the fault produces collisions. For example, let us assume that after several rounds, the difference reaches nibbles 8 and 11, producing the S-Box output masks of value '8' for both nibbles. These two bits will propagate to nibble 15 in the next round, producing the S-Box input mask 'C'. Such a difference cannot be easily seen from the power trace by just observing the differential peaks, and therefore, we can only assume that some of the nibbles between 8-11 in the previous round were faulted. This behavior would require creation of SCA templates for each nibble and each fault mask, resulting in total of 256 different templates.

Similar scenario can be seen in Figure 8, depicting the last three rounds of encryption – upper part is the power trace of the non-faulty encryption process, so that it makes it possible to estimate particular rounds. Lower part is the differential trace which shows that while it is trivial to determine the fault mask for round 28 based on difference in round 29, following rounds are hard to analyze.

B. Extension to Other PRESENT-like Block Ciphers

The combined SCA and FA based attack on PRESENT can also be extended to other PRESENT-like block ciphers that use bit-permutations for diffusion. One such recently proposed block cipher is GIFT [3]. GIFT has two versions - GIFT-64 with a 64 bit plaintext block and GIFT-128 with a 128 bit plaintext block. The bit-permutation layer for GIFT-64 differs from PRESENT, which lends it some additional advantages against cryptanalytic attacks and better efficiency in software. However, similar to PRESENT, the bit-permutation layer of GIFT-64 also ensures that each of the four sets of nibbles, namely [0-4],[5-8], [6-11] and [12-15] affect exactly four non-overlapping sets of nibbles $\{0,4,8,12\}$, $\{1,5,9,13\}$, $\{2,6,10,14\}$ and $\{3,7,11,15\}$, respectively. Since it is this property of the bit-permutation that our combined attack exploits, the attack is also applicable to GIFT-64 with the same efficiency as PRESENT. An extension of the attack is also applicable in case of GIFT-128, although a detailed description of the same is beyond the scope of the current paper.

It is, however, interesting to see that the combined attack methodology would fail against ciphers such as AES or LED that use MDS layers. The diffusion characteristics of an MDS layer would 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.

C. Possible Countermeasures

We briefly 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 [20] that allow injecting the same fault in both the original and redundant computations with high probability. Standard side-channel countermeasures such as masking could be incorporated to make the attack more difficult in the sense that one might require higher order analysis over a larger number of power traces to retrieve the desired fault mask value in such a scenario. Other less costly alternatives to masking such as shuffling and hiding might also increase the complexity of retrieving the fault mask.

VI. Conclusion

In this paper, we have practically demonstrated the strength of combining SCA and FA in attacking PRESENT-like block ciphers that use bit-permutations instead of MDS layers for diffusion. We used a laser fault injection based setup to inject nibble faults in the 28^{th} round of PRESENT, and perform a simplified variant of differential power analysis on the correct and faulty execution trace to fully infer the resulting fault mask. We subsequently demonstrated a

generalized DFA expanding across the last four rounds of PRESENT that allows reducing the entropy of the input fault masks for the last round, and recovering multiple key nibbles in parallel. We have practically corroborated our theoretical observations via actual fault injection and key recovery attacks on an ATmega328P microcontroller-based implementation of PRESENT-80. Our attack exposes an interesting and unexplored vulnerability of PRESENT-like block ciphers that use bit-permutations as opposed to MDS layers for diffusion. Further research can look into vulnerability of MDS matrix-based linear layer for similar attacks. Extension of these attacks on protected targets can also be an interesting line of work.

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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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 Side-Channel Analysis and Secure Design*. Springer, 2013, pp. 137–153.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.