Breaking Redundancy-Based Countermeasures with Random Faults and Power Side Channel

Sayandeep Saha*, Dirmanto Jap†, Jakub Breier†, Shivam Bhasin†, Debdeep Mukhopadhyay*, and Pallab Dasgupta*

*Department of Computer Science and Engineering, IIT Kharagpur, India
† Physical Analysis & Cryptographic Engineering (PACE) Labs, Nanyang Technological University, Singapore
{sahasayandeep, debdeep, pallab}cse.iitkgp.ernet.in, {sbhasin, djap, jbreier}@ntu.edu.sg

***Abstract*—Redundancy based countermeasures against fault attacks are a popular choice in security-critical commercial products, owing to its high fault coverage and applications to safety/reliability. In this paper, we propose a combined attack on such countermeasures. The attack assumes a random byte/nibble fault model with existence of side-channel leakage of the final comparison, and no knowledge of the faulty ciphertext. Unlike the previously proposed biased/multiple fault attack, we just need to corrupt one computation branch. Both analytical and experimental evaluation of this attack strategy is presented on software implementations of two state-of-the-art block ciphers, AES and PRESENT, on an ATmega328P microcontroller, via side-channel measurements and a laser-based fault injection. Moreover, this work establishes that even without the knowledge of the faulty ciphertexts, one can still perform differential fault analysis attacks, given the availability of side-channel information.**

***Index Terms*—Fault Attack, Side-Channel, Combined Attack, Redundancy-Based Countermeasure**

I. Introduction

Fault-based cryptanalysis is one of the potent practical threats to modern cryptographic primitives. It has been shown on several occasions that a certain number of properly placed faults can compromise the security of state-of-the-art cipher implementations. Among different classes of fault attacks, Differential Fault Analysis (DFA) [1] is the most prominent and generic one in the context of block ciphers. Two most remarkable features of DFA are; 1) its extremely relaxed random fault model, and 2) low fault complexity. However, it has been found that faults occurring at certain devices are often biased in nature. Such biased fault models have been successfully exploited to realize so-called Statistical Fault Attacks (SFA [2]), which are simpler than DFA but demand a large number of faults. Few SFA can work even with knowledge of faulty ciphertext only [3].

Countermeasures to fault attack generally use some form of redundant computation (time, information, or space redundancy) [4]–[6], to detect the presence of a fault. One of the most common and effective countermeasures involves double computation (in time or space) with a final comparison. If the results from redundant computations do not match, the output of the encryption function is suppressed/randomized. Under a single fault injection model, this countermeasure has 100% fault coverage. Likewise, the number of redundant computations can be increased to $n+1$ to detect up to n fault injections. It is widely used in commercial products because it also has applications in safety and reliability. Moreover, if the designer is using highly optimized encryption functions, the countermeasure only requires an extra call (time) or instantiations (space) without the need of major modification. Few previous works have shown practical attacks against this countermeasure. For example, [7] uses biased fault model to inject 2 faults in both computation branches of a time redundant countermeasure (for $n = 1$). A double laser injection was shown to break space redundant countermeasure [8]. Thus, both the attacks exploited 2 fault injection locations (for $n = 1$) to break a time/space-redundant countermeasure.

In this paper, we present a practical combined attack on redundancy countermeasure (either space or time). It works with a single fault location adversarial model, thus injecting faults in one of the redundant branches. Then, it uses a side-channel information to get the knowledge of the fault propagation. This attack does not require any faulty ciphertexts. Instead, we assume that byte/nibble-wise Hamming distance (HD) of the correct and the faulty ciphertext is known for each fault injection. Virtually every fault attack countermeasure either blocks or randomizes the output once a fault is detected. Hence the absence of faulty ciphertexts brings this attack to practical conditions. On the other hand, the byte/nibble-wise Hamming weight (HW) can be obtained quite accurately by means of power or electromagnetic side-channels from the block comparing the actual and a redundant computation. Based on these assumptions, we propose DFA attacks for two state-of-the-art block ciphers AES [9] and PRESENT [10]. In a nutshell, the major contributions of this work are as follows:

- We propose a first combined attack on redundancy based countermeasure. The attack assumes a *single fault location adversary* model assisted with supplementary side-channel information. This, to best of our knowledge, is the first work addressing inherent vulnerabilities of a countermeasure without bypassing it.
- The attack functions under random fault model, unlike, previous works which assume a biased or multiple fault injection.
- The proposed attack does not require the knowledge

of faulty ciphertexts. The side-channel is exploited to measure HD of correct and faulty ciphertext. Although the recently proposed ineffective fault attacks [11] work without faulty ciphertexts, they typically exploit a very special class of biased faults which may not take place in all class of devices.
- The proposed attack strategy is evaluated on two widely deployed block ciphers – AES and PRESENT.
- Validation of the attack is shown using practical laser fault injection on microcontroller target. The number of required injections are 2^{25} and 4 for AES and PRESENT respectively, which can be easily performed in a day.

The rest of the paper is organized as follows. In Section II, we present a brief description of the block ciphers under consideration, as well as a precise description of the redundancy-based countermeasure we target. A brief overview of the related work is also presented. Section III formally presents the proposed attack and describes two use-cases on AES and PRESENT. Practical realization of the attacks are discussed in Section IV. Finally we conclude in Section V.

II. Preliminaries

A. Overview of AES and PRESENT Block Ciphers

AES [9] is a Substitution-Permutation Network (SPN) cipher which is currently the worldwide standard for symmetric key encryption. The most widely utilized version, AES-128, is composed of 10 rounds and has a block size and a key size of 128 bits. Each encryption round, except the last one, consists of 4 bijective boolean functions to provide confusion and diffusion. The nonlinear *SubBytes* layer of AES, which provides confusion, consists of 16, 8×8 S-Boxes. Each S-Box performs an inversion operation on the finite-field $GF(2^8)$ followed by an affine transform. On the other hand, the diffusion layer of AES is a combination of a byte-permutation (*ShiftRows*) followed by a Maximum Distance Separable (MDS) matrix (*MixColumns*). Finally, a round key of 128 bits, generated from the 128-bit master key by means of an invertible key schedule, is XOR-ed with the state at the end of each round to ensure the confidentiality (*AddRoundKey*). The last round of AES does not include the *MixColumns* operation.

The block cipher PRESENT [10] is another member of the SPN class of ciphers, which is specifically engineered for lightweight applications. It consists of 31 rounds, a block size of 64 bits and supports master keys of lengths 80 and 128 bits. Each encryption round contains: an *addRoundKey* layer for XORing the round keys, a nonlinear S-Box layer having 16 4×4 S-Boxes (*sBoxLayer*), and a linear bit-permutation layer (*pLayer*) providing the diffusion. Additionally, a post-whitening round key is XOR-ed with the outcome of the last round. Similar to AES, PRESENT also consists of an invertible key schedule generating 64-bit round keys from the 80/128-bit master key.

B. Redundancy-based Fault Attack Countermeasures

Redundancy based countermeasures are probably the most widely used primitives to thwart DFA attacks. They can be classified in three broad classes, namely – information redundancy, time redundancy, and space redundancy. Information redundancy countermeasures utilize error-detection/correction codes as redundant computation [12]. Such countermeasure strategies, though typically lightweight in nature, may not provide 100% fault coverage guarantee.

Time and space redundancies [5], [6], on the other hand, are based on a relative simpler principle of recomputation. In the easiest case, the cipher can be recomputed on the same input twice in time or space and then the results can be checked for a match [5]. Such recomputation may happen at various levels of granularities, for example for each round or even for each sub-operation. A couple of other variants, like computing inverse or redundant computation over permuted operands, has also been proposed. In fact, some of such implementations are commercially deployed and also shown vulnerable against practical fault attacks [13].

C. Related Work

The proposed attack in this work uses Side-Channel Analysis (SCA) to aid the fault analysis attack. As shown before, fault injection countermeasures can lower the implementation resistance to SCA [14]. Several works have demonstrated SCA assisted biased fault attacks on AES [15]–[17]. The first practical combination of DFA with SCA was proposed in [18]. Authors used SCA to determine the exact value of injected random fault mask which simplified the analysis. Practical attacks were shown on an unprotected implementation of PRESENT-80 running on an 8-bit microcontroller. The attack strategy proposed by us is somewhat motivated by [18].

Attacks on redundancy based countermeasures have also been explored previously. In [19] and [20], authors practically show that exploitable fault injections are possible for most of the cases with high probability in information redundant countermeasures. A strong practical example in this context is [13], where it was shown that the commercially available processors for automotive domain, having an ASIL-D level of safety and fault tolerance guarantees can be easily bypassed with faults. Another relevant work in this context is [7], where the bias in the fault distribution was exploited to inject same faults in both branches of a time redundancy countermeasure for AES. The work in [8] extended the same attack philosophy on infection based countermeasures using multiple laser-based fault injection. Recently, a fault-assisted side-channel attack was proposed [21], which faulted the mask loading in a masked implementation, resulting to a zero mask and effectively changing the implementation to unprotected one. Then they could mount a classical side-channel attack on the cipher.

However, most of these attacks tried to bypass the countermeasure mechanism, whereas we try to exploit the countermeasure itself to leak the information.

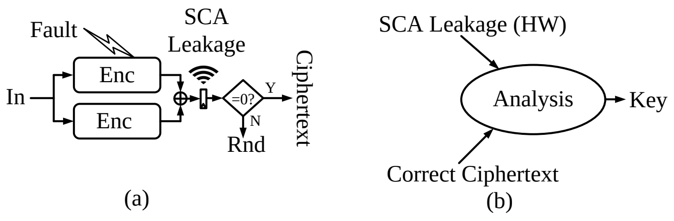


Fig. 1: Fault propagation pattern in AES with the fault injected at the beginning of the 8th round. The distinguishing property is shown in grey.

III. Attacking Redundancy-based Countermeasures with Random Faults

In this section, we describe the proposed attack strategy on redundancy based countermeasures with random byte/nibble faults. Our work can be distinguished from the previous proposals in two aspects. The random fault injection attacks described previously in [19], [20] try to figure out faults which can evade the countermeasure. In contrary, the proposed attack here works even if the fault is detected. On the other hand, the biased fault attacks in [7] and [8] assume relatively restricted fault model and injection of two faults in actual and redundant computation, whereas our proposal assumes a single fault injection.

A. Adversary Model

In this work, we target time/space redundancy countermeasures which perform encryption operation on the same data at least twice and then compare the ciphertexts. Due to its simplicity, this form of fault tolerance can be adapted quickly in most of the practical scenarios. Almost every redundancy-based countermeasure uses a comparison operation to detect the existence of faults in the computation. Such comparison blocks are supposed to report an error if the ciphertexts are unequal. The adversary measures the side-channel activity of the comparison.

The main idea of the proposed attack is to extract HW of this XOR output. The main idea is depicted in Figure 1. Without the loss of generality, we assume that there is one actual, and one redundant branch of computation in the countermeasure implementation under consideration. It is fairly reasonable to assume that the output of the XOR comparison is stored in registers, which in turn causes the side-channel leakage. A random byte/nibble fault with unknown fault value and known location is injected at one of these computation branches. The HW obtained basically represents the HD between the correct and the faulty ciphertext. For obvious reasons, we assume the correct ciphertext to be known. On the other hand, the attacker will have no exact knowledge about the faulty ciphertexts as they are supposed to be blocked or randomized after the fault is detected. It should be noted that the assumption of random fault injection is still relevant, even with the existence of sophisticated biased fault attacks on redundancy-based countermeasures [7]. In practice, one cannot expect the existence of a favorable biased fault model for all possible class of devices. Further, there exist biased fault aware synthesis techniques for hardware [22], which one may also adopt for certain general purpose processors. All these facts leave random fault models as the only mean for realizing fault injection attacks, at least for certain scenarios.

The experimental validation is later done on a 8-bit micro-controller, thus we assume assumption the knowledge of byte-wise HW from SCA.

B. A General View for DFA Attack Complexity

DFA attacks are known for their extremely relaxed fault models. However, the complexity of the attacks become extremely significant while realizing them practically. From a general point of view, the complexity of a fault attack depends on three factors. The first among them is the number of faults injected which we denote as $|\mathcal{F}|$. Fault injection results in a distinguisher (or a set of distinguishers) which is used as a filter to rule out wrong key guesses. Each key guess must be evaluated by the distinguisher exhaustively (in a divide-and-conquer fashion). So there is a cost of exhaustive evaluation which we denote by $|\mathcal{E}|$. Finally, there is a reduced key-space after distinguisher evaluation, to be exhausted via brute-force search. Let us denote the size of the remaining keyspace as $|\mathcal{R}|$. The complexity of the attack is denoted as:

$$\mathcal{C} = \langle|\mathcal{E}|, |\mathcal{F}|, |\mathcal{R}|\rangle \tag{1}$$

One should note that in order to be practically realizable, all of the three complexity measures must remain below some feasible computational limits. We consider the feasible limit of computation to be 2^{50}, which is reasonable with modern computers. However, in the case of $|\mathcal{F}|$ and $|\mathcal{R}|$, one must also ensure that they still remain lower than the brute-force search complexity of the key bits associated. Otherwise, no gain will be obtained from the fault attack.

C. Attacking Block Ciphers without Faulty Ciphertexts

Being a special form of SCA attacks, fault attacks also exploits information leakage to reduce the entropy of the secret key. Although the source of the leakage in all classes of fault attacks is the injected fault, the observation point of the leakage may be different. For example, in classical DFA attacks, the faulty ciphertexts manifest the leakage. On the other hand, in typical biased fault attacks e.g. Differential Fault Intensity Analysis (DFIA) [2], information is leaked both via the faulty ciphertexts and the biased fault model. In certain cases, the very existence of fault also leaks information. Most prominent examples of such fault attacks are Fault Sensitivity Analysis (FSA) [23] and the recently proposed Ineffective fault-based attacks [11]. In fact, with a rather restricted fault model, attacks can be performed without the knowledge of the plaintext or the ciphertexts, as shown in [24].

While most of the alternative leakage sources are manifested in the context of biased faults, in this work we show that such leakage also exists for DFA. The byte-wise HD of the correct and faulty ciphertexts are the main source of leakage in the

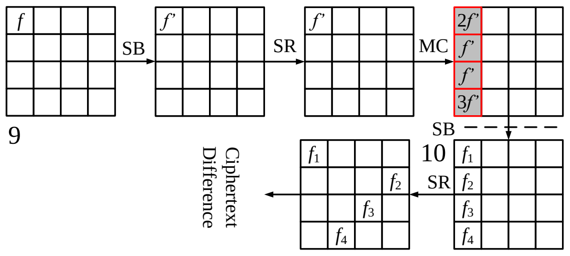


Fig. 2: Fault propagation pattern in AES with the fault injected at the beginning of the (a) 9th round, and (b) 8th round. Distinguishing properties of interest are shown in grey.

proposed attack strategy. In the following, we explain the root cause of key entropy reduction due to this HD information.

Let us denote a byte of the correct ciphertext as C and that of a faulty ciphertext as C^*. The XOR difference between these two is denoted as δ, and the SCA provides us with $HW(\delta) = HW(C \oplus C^*)$. Given $HW(\delta) = w$ ($w \in \{1,2,3,4,5,6,7,8\}$) and C as known, we have $\binom{8}{w}$ choices for C^*, whereas without the knowledge of $HW(\delta)$, the number of choices would become 2^8. One of these $\binom{8}{w}$ choices must be the actual faulty ciphertext. A key observation is that, for typical values of $HW(\delta)$, $\binom{8}{w}$ can be significantly low, and in the best case with $w = 8$, $\binom{8}{w}$ becomes 1. Hence, it can be concluded that, *even without having the exact knowledge of the faulty ciphertexts, for certain choices of $HW(\delta)$ the number of choices for C^* will be sufficiently low to be tried exhaustively.* This is the main motivation of the attacks we are going to propose next.

D. Case Study I: Attack on AES

Let us consider the AES block cipher with a byte fault injected at the beginning of the 9th round. The propagation pattern of the fault is presented in Figure 2. The fault corrupts only 32 bits of the state and consequently only 32 bits of the keys associated can be extracted simultaneously [1]. In order to extract the last round keys in 32-bit chunks, one has to solve a system of difference equations, as shown in the following.

$$\begin{aligned} 2f_1 &= S^{-1}(C_1 \oplus k_1) \oplus S^{-1}(C_1 \oplus \delta_1 \oplus k_1) \\ f_1 &= S^{-1}(C_{14} \oplus k_{14}) \oplus S^{-1}(C_{14} \oplus \delta_{14} \oplus k_{14}) \\ f_1 &= S^{-1}(C_{11} \oplus k_{11}) \oplus S^{-1}(C_{11} \oplus \delta_{11} \oplus k_{11}) \\ 3f_1 &= S^{-1}(C_9 \oplus k_9) \oplus S^{-1}(C_9 \oplus \delta_9 \oplus k_9) \end{aligned} \quad (2)$$

Here C_i, k_i and δ_i denote a byte of the correct ciphertext, the key and XOR difference of the correct and the faulty ciphertext, respectively ($i \in \{1,9,11,14\}$). Note that, exact knowledge of δ enables exact determination of the faulty ciphertext C^*. S^{-1} here denotes the inverse S-Box operation. In classical DFA attack, both the correct and faulty ciphertexts are known and as a result, each δ_i is also known uniquely. However, the present context, where only the HW of each δ_i is known, leaves us with multiple options for a δ_i, and as a result, δ_i must be treated as an unknown in the difference equations.

[1]It may seem that an injection at 8-th round, corrupting the complete state is better. However, we may not be able to exploit this advantage in our present scenario, as it will be shown later in this paper

From Eq. (2), it is evident that one should enumerate the candidate solutions for each pair (k_i, δ_i), one of which will provide the correct key as well as the original faulty ciphertext byte. The total number of possible values assumed by each δ_i depends upon its HW. Assuming the width of δ_i to be 8, there could be 8 possible HW's excluding the 0. The number of candidate δ_is corresponding to each HW is given by $\binom{8}{W(\delta_i)}$, with $W()$ denoting the HW. As a result, the number of values assumed by each (k_i, δ_i) pair becomes $2^8 \times \binom{8}{W(\delta_i)}$.

For any fixed δ_i and f_i each equation from Eq. (1) gives one solution for k_i on average. Using this fact, we can estimate the number of solutions for Eq. (1) having form $\langle (k_1,\delta_1),(k_9,\delta_9),(k_{11},\delta_{11}),(k_{14},\delta_{14})\rangle$. The total number of possible solutions is $\left(2^8\right)^4 \times \prod_{i\in I}\binom{8}{W(\delta_i)} \times 2^{-24} = 2^8 \times \prod_{i\in I}\binom{8}{W(\delta_i)}$, where $I = \{1,9,11,14\}$. The factor 2^{-24} is the probability of occurrence of the distinguishing pattern $\langle 2f_1, f_1, f_1, 3f_1\rangle$. As per the notations in this paper, we have $|\mathcal{E}| = 2^{32} \times \prod_{i\in I}\binom{8}{W(\delta_i)}$, $|\mathcal{R}| = 2^8 \times \prod_{i\in I}\binom{8}{W(\delta_i)}$, for $|\mathcal{F}| = 1$.

1) Selection of Proper Faults: It is evident that in our attack setup under consideration, keys are extracted in 4-byte chunks. So, in order to obtain an attack, which is better than brute-force we have to settle down to a complexity figures of $|\mathcal{R}| \le 2^{32}$ and $|\mathcal{F}| \le 2^{32}$. Also we require $|\mathcal{E}|$ to be a practically enumerable figure. Let us first analyze the worst case complexity figures for this attack. The worst case situation happens when $\binom{8}{W(\delta_i)} = \binom{8}{4} = 70$. In this case, we have $|\mathcal{E}| = 2^{32} \times \binom{8}{4}^4 \approx 2^{32} \times (2^6)^4 = 2^{56}$, and $|\mathcal{R}| = 2^{32}$ for $|\mathcal{F}| = 1$. Apparently, there is no reduction in the key space. In contrary, the best case situation happens while $\binom{8}{W(\delta_i)} = \binom{8}{8} = 1$. In this case, we can uniquely determine the faulty ciphertext and the $|\mathcal{R}|$ becomes 2^8.

One crucial observation at this point is that we can identify the best and worst cases quite easily as we already have the HW information in our possession for each fault injection. One can thus wait for a favorable fault case to happen and then launch the attack. This will, however, increase the $|\mathcal{F}|$. Unfortunately, the probability of occurrence of a favorable even is often low. For example, the best case event has the probability $Pr[\wedge_{i\in I} W(\delta_i) = 8] = \frac{\prod_{i\in I}\#(W(\delta_i)=8)}{2^{32}} = (\frac{1}{2^8})^4 = 2^{-32}$. That is to say, we have to inject 2^{32} faults in order to get the best case event. Also, note that unique identification of the key is still not possible with one favorable case and we need one more such case. So, in total, we require 2^{33} injections to uniquely figure out 32 secret key bits. This is, in fact, worse than exhaustive search once again.

It is apparent that we have to strike a trade-off between the number of fault injections and the size of the remaining key space. In order to achieve this, we set a simple strategy. Let $S = \{1,2,3,4,5,6,7,8\}$ be the set of all possible HWs for a byte. The idea is to consider cases from the set $S' = S - \{3,4,5\}$. To be precise, we want $|\mathcal{F}| = \frac{2^{32}}{\prod_{i\in I}\#(W(\delta_i)=w_i)}$, where $w_i \in S'$. In the worst case, $|\mathcal{F}| = \frac{2^{32}}{(28)^4} \approx \frac{2^{32}}{(2^5)^4} \approx 2^{12}$,

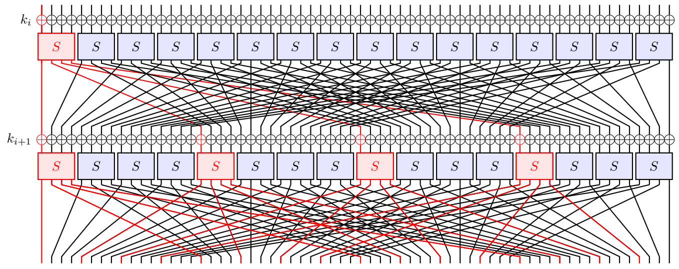


Fig. 3: Fault propagation in PRESENT with the fault injected at the beginning of the 30^{th} round (modified from [26]).

which is a fairly reasonable quantity. On the other hand, the worst case value of $|\mathcal{R}|$ is $|\mathcal{R}| = 2^8 \times \prod_{i \in I} \binom{8}{2} = 2^8 \times (28)^4 \approx 2^8 \times (2^5)^4 = 2^{28} \leq 2^{32}$. For a random selection of $w_1 = 1, w_9 = 2, w_{11} = 8, w_{14} = 7$, we get $|\mathcal{F}| = 2^{32}/(8 \times 28 \times 1 \times 8) \approx 2^{22}$ and $|\mathcal{R}| = 2^8 \times (8 \times 28 \times 1 \times 8) \approx 2^{18}$. This means that with 2^{22} fault injection the key space reduces to 2^{18}. So, with 2×2^{22} injections, the 32 bit key can be determined uniquely. The complete 128-bit round key can be extracted with total $2^{23} \times 4 = 2^{25}$ fault injections in this case.

2) Fault Injection at the 8th Round: Injection of a fault at the beginning of the $8th$ round results in complete diffusion of the fault to all the bytes. As it has been shown in [25], propagation of the fault gives rise to 4 independent system of difference equations returning each of the 32-bit key chunks. In the case of classical DFA, where the ciphertext bytes are known, the complete 128-bit key can be extracted only by means of a single injection in 8th round [25]. However, the present context is slightly different as here we can only use those fault instances which result in specific HW patterns in the ciphertext differences. Obtaining favorable HW patterns for more than 4 bytes of δ_is will have extremely low probability in practice, which will necessitate injection of a formidable number of faults. As a result, we cannot directly take the advantage of 8th round fault injection in the present scenario. However, one may carefully store incidentally occurring favorable cases corresponding to each independent key chunk, while running a fault campaign. This strategy should provide quite a significant reduction in the number of fault injections.

3) Practicality of the Proposed Attack: Although the complexity figures of the proposed attack are feasible both theoretically and practically, the number of fault injections required is still significantly high. However, we argue that this is probably the best figure we can achieve without the knowledge of the faulty ciphertexts in a random fault scenario. Also, with the availability of sophisticated and fast commercial fault injection setups, such figures are quite possible to achieve practically. As shown later, our setup can easily reach over 2^{26} correct and faulty ciphertext pairs per day. It is largely limited by the time for each encryption.

E. Case Study II: Fault Attack on PRESENT

It is apparent from the last example that, attacking AES is indeed possible even without exact knowledge of the faulty ciphertexts. However, the fault complexity is relatively high. In this second use case, we show that for certain lightweight block ciphers such attacks can be realized with very reasonable fault and computational complexities. As a concrete realization, we describe an attack on the PRESENT cipher with nibble faults injected at the beginning of the 30th round.

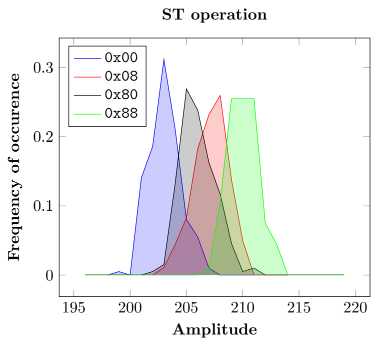


Fig. 4: Extracting nibble-wise HW values from byte-wise HWs. SCA traces for 4 possible situations. The distributions are derived from practical measurements on a micro-controller.

The encryption operation in PRESENT involves 31 round keys and a post-whitening key at the end of the 31st round. We aim to extract this post-whitening with our proposed attack. With the knowledge of per-nibble HD of correct and faulty ciphertexts, we solve equations having the following form for each nibble.

$$f_i = S^{-1}(C_i \oplus K_i) \oplus S^{-1}(C_i \oplus \delta_i \oplus K_i), \text{for } i \in [0, 15] \quad (3)$$

1) Choice of the Injection Round: Most of the previous DFA attacks on PRESENT choose the 28th or 29th round of the cipher. Such choices allow fault propagation to most of the ciphertext nibbles with a very high probability, and also reduce the required number of injections. However, in the current context, we choose the 30th round as the fault injection point. The reason behind this choice is entirely practical. Since our target architecture is an 8 bit microcontroller we can only expect byte-wise HW values by means of SCA. However, PRESENT utilizes nibble and bit-level boolean operations and it is not straightforward to obtain nibble level information from byte-level SCA. Although template building can be a general solution to this problem, detailed evaluation of template building is out of the scope of this paper. For our specific case, however, we can provide an alternative and simple solution. The idea is to restrict the number of values each δ_i may assume. Note that δ_is are a 4-bit variables.

With a fault injection at 30th round, we can easily achieve the aforementioned goal. We first illustrate this by means of an example. The fault propagation pattern, with a 30th round nibble fault injection, is depicted in Figure. 3, where the nibble 0 of the state has been corrupted. We call a nibble/byte as *active*, if the differential of the correct and faulty values assumed by it is nonzero. Similarly, an S-Box is *active*, if it has a non-zero input differential. Referring to Figure. 3, one can observe that for any value of the injected fault at round

30, at least 1, and at most, 4 S-Boxes at round 31 may become active. Moreover, the input differential for each of these active S-Boxes assumes the value 1000, thanks to the bit-permutation layer of PRESENT. After the completion of the 31st round, at least 4 and at most 16 nibbles get activated. Once again due to the bit-permutation operation, all the corrupted nibbles assume the value 1000.

The above-mentioned observation can generalized for any given nibble location and rounds as follows:

Observation 1. *For $n, d, l \in \{0, 1, 2, 3\}$, if the $(4n + d)$th S-Box at round r is active, then the lth bit of the output differential of this active S-Box becomes the dth bit of the input differential of the S-Box $(n + 4l)$ in round $r + 1$. As a consequence, for any nibble fault injection at round r, the input differentials of the active S-Boxes at round $r+1$ becomes* 1000 *or* 0100 *or* 0010 *or* 0001*, depending on the nibble where the fault was injected. The same input differential values can be observed for the active S-Boxes at round $r + 2$. Further, the number of active S-Boxes are at most* 4 *for round $r + 1$ and at most* 16 *for round $r + 2$.*

It is now trivial to observe that after a fault injection at 30th round, the nibble-wise HWs for each ciphertext differential ($W(\delta_i)$) becomes either 1 (for active nibbles) or 0 (otherwise). An example of a typical nibble-wise HW pattern in the ciphertext is $[0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1]$. *In terms of bytes, one can only observe* 3 *possible HW values, namely* 0, 1 *and* 2. While it is fairly straightforward to identify the corresponding nibble-wise HWs for byte-wise HW values 0 and 2, an ambiguity takes place if a byte-wise HW of 1 is obtained. There are two possible patterns of nibble-wise HWs ($[1, 0]$ and $[0, 1]$) corresponding to a byte-wise HW of 1. Fortunately, we found that the SCA signatures corresponding to these two patterns are clearly distinguishable. Figure 4 depicts the SCA traces corresponding to 4 byte-wise HW cases which shows that nibble-wise HW values can be identified unambiguously.

2) Nibble-wise Key Recovery with HW Information: As already pointed out in the last subsection, the δ_is can only assume values 1000, 0100, 0010, and 0001, while active, and a value 0 otherwise. Further, for a fixed fault injection location, only a single value can be assumed by an active δ_i. For example, as per the fault propagation pattern shown in Figure. 3, an active δ_i can only assume a value 1000. Hence, for a known fault injection location, there is a one-to-one correspondence between the δ_i and $W(\delta_i)$ values. As a consequence of the aforementioned one-to-one correspondence, we can directly figure out the δ_i values from the observed $W(\delta_i)$ values and the faulty ciphertexts can be obtained directly.

With the faulty ciphertexts in possession, the post-whitening key can be recovered nibble-wise in a parallel manner, by solving Eq. (3) independently for each nibble. One should note that the exact value of f_i is known here having a probability of occurrence $\frac{1}{2^4}$. As a result, the quantity $|\mathcal{E}|$ becomes 2^4, whereas $|\mathcal{R}|= 2^4 \times 2^{-4} = 1$ for $|\mathcal{F}|= 1$. In nutshell, each key nibble can be uniquely determined by means of a single fault injection. Further, due to the incomplete diffusion of the fault up to the beginning of the 31st round (see Figure. 3), at most 4 key nibbles can be recovered at once for a single nibble fault injection. In order to recover the complete 64 bit key, thus the adversary should corrupt 3 other nibble locations, one at a time. The total number of fault injections thus becomes $|\mathcal{F}|= 4$, for a complete round key. Also, due to the probabilistic nature of the fault propagation in PRESENT, not all the nibble gets active with a single fault injection at the 30th round. The calculation $|\mathcal{F}|= 4$ is, therefore, a best-case estimate of the total number of injections required. In practice, $|\mathcal{F}|$ will increase to some extent. At the worst case, recovery of each key nibble will require one injection resulting in total 16 injections. All these complexity figures are fairly reasonable.

IV. Experiments

To validate our attack, we implemented the target design on an 8-bit microcontroller with a 2-stage pipeline. The choice of micro-controller was motivated by reasonable signal to noise ratio (SNR) for side-channel measurement and unexpected fault effects in the pipeline.

A. Experimental Setup

For the purpose of fault injection, we use laser as the injection technique. The laser setup is made of a near-infrared (1064 nm) pulse laser diode with a power of 20 W, reduced to 8 W by using 20× objective lens. The spot size after 20x magnification is 15×3.5 μm. The device under test (DUT) was mounted on the X-Y-Z positioning table with a step precision of 0.05 μm. The DUT, operating at a core frequency of 16 MHz, has a total chip area of 3 mm × 3 mm. For a DUT operating at 16 MHz, the laser system can target every other instruction with its pulse repetition rate of 10 MHz. To enable laser injection, the chip had to be opened from the backside by mechanical means to expose the substrate. The substrate of the chip is further polished for making the laser injection effective. To precisely localize the points of injection we perform a profiling phase on the microcontroller instruction set. We identified a favorable zone on the chip with an area of approximately 75×100 μm large (≈0.083% of the chip area). The laser power was capped at 4.5% as it allowed 100% repeatability of faults and higher power might have destroyed the chip. After the initial profiling phase, the cipher was implemented and flashed on the micro-controller. We target the S-Box look-up operation in both AES and PRESENT ciphers. This primarily involves targeting a `LD` (load) operation during a table look-up. As both the cipher implementations are constant time, once the timing of target `LD` operation is determined, we are able to inject faults with 100% repeatability in both AES and PRESENT. When analyzed, the resultant fault model is instruction-skip/change, but when ported to S-Box look-up, it is equivalent to random byte fault, which is compatible with the required fault models.

B. Practical Evaluation and Results

The redundancy countermeasure was first implemented in 'C' language, by comparing outputs of two encryption calls with same inputs. However, the resulting comparison was not constant time with several `branch` instructions. To fix the timing problem, the comparison was implemented by series of `EOR` instructions (XOR). We identified two prominent instructions to measure which carried the information related to the comparison of redundant two ciphertexts i.e. `EOR` and `ST` (store). As the data width of the micro-controller is 8-bits, the comparison is implemented byte-wise.

The side-channel measurement on the comparison of redundant encryption is done using a Lecroy WaveRunner 610zi oscilloscope. We preferred electromagnetic (EM) measurement over power to achieve high SNR. RF-U 5-2 H-field probe from Langer was used for the measurement. The measurement was further boosted by 30 dB amplification and a low-pass filter (0-48 MHz). However, with the micro-controller under a microscope for laser injection, it was hard to measure the EM signal directly on the chip. We looked for points of interest along the chip to measure using visual profiling of S-Box operation. By trial and error approach, we were able to find relevant measurement points on the package, while still allowing laser injection on the chip. Finally, the EM measurement is disturbed by laser injection itself, which can be avoided by allowing few cycles between laser injection and side-channel measurement. As the fault is done during the encryption and side-channel measurement in final comparison, this condition was easy to achieve.

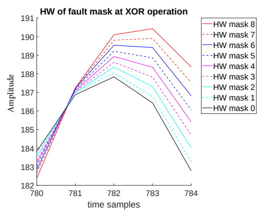
The final measurements are done for AES-128 and PRESENT-80. The byte-wise comparison is ideal in case of AES, but for PRESENT-80 it will be ideal to measure the comparison nibble-wise. However, as already explained in Section. III-E1, measuring PRESENT-80 comparison byte-wise will also give nibble-wise information (with some added noise). The details of implementation are given in Table I. As we need one correct and one faulty ciphertext for each plaintext, twice the number of encryptions must be performed. Thus the total number of correct faulty ciphertext pair collected in a day is also provided in Table I. The measurement results and the distribution along different HWs is shown in Figure 5 (for AES-128) and Figure 6 (for PRESENT-80). As shown, the `ST` operation allows a clear identification of the ciphertext difference. Even for `EOR` operation the identification is possible. This especially works with our fault model where only corner HW are of prime interest. Extending to other targets like 32-bit micro-controller or other targets would need a proper update of the measurement setup.

C. Discussion

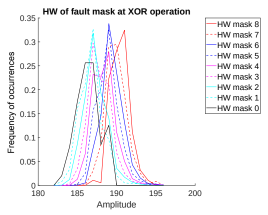
It is worth noting that, apart from being the first instance of DFA having no explicit information about the faulty ciphertexts, the proposed attack strategy also points out that the comparison step of correct and faulty ciphertext must be protected properly. The technique has been implemented on an 8-bit microcontroller. Further adaption of SCA measurement setup might be required for the extension to other platforms, which could involve, template building, localized measured, pre-processing and filtering etc.

TABLE I: Summary of the attacks. T_{ENC} denotes delay of single redundant encryption in milliseconds. N_{EXP} denotes maximum number of laser injection per day. The last column presents the estimated attack complexity to extract a complete round key (refer Eq. (1)). Note that the fault complexity ($|\mathcal{F}|$) for PRESENT will be slightly more due to its probabilistic fault propagation

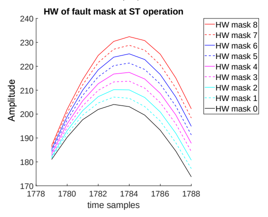
Cipher	Code Size (bytes)	T_{ENC}	N_{EXP}	($\lvert\mathcal{E}\rvert$,$\lvert\mathcal{F}\rvert$,$\lvert\mathcal{R}\rvert$)
AES-128	7570	0.326	$2^{26.98}$	(2^{43},2^{25},1)
PRESENT-80	7110	4.01	$2^{23.36}$	(2^4,4,1)



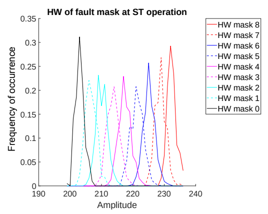
(a)



(b)



(c)



(d)

Fig. 5: The side-channel observations when comparing the correct and faulty ciphertext on byte level. (a) The HW difference when observing the XOR directly (`EOR` operation) and (c) the storing of comparison (`ST` operation). The distribution of the HW values for each case are presented in (b) and (d).

To protect against such attacks, the comparison function must be implemented in an SCA secure manner. Techniques like masking and shuffling can be readily adapted to secure this comparison. However, it will increase the overheads to some extent. One should also note that this attack is not directly applicable to a relatively costly implementation of redundancy based countermeasure, where the comparison operation is performed at the end of each round. The reason is that the correct ciphertext is unavailable in such scenarios. However, the HW information can still be obtained. An interesting future extension could be to figure out an attack for these situations.

V. Conclusion

Evaluating the security of fault attack countermeasures is a problem of great theoretical and practical value. The problem has become even more relevant in the recent context with the inclusion of side-channel security as an essential evaluation criterion in the NIST call for lightweight ciphers [27]. In this paper, we address this problem for a class of time/space redundancy countermeasures and show that they can be attacked by means of localized random faults. Perhaps the most important contribution here is to show that the countermeasures are

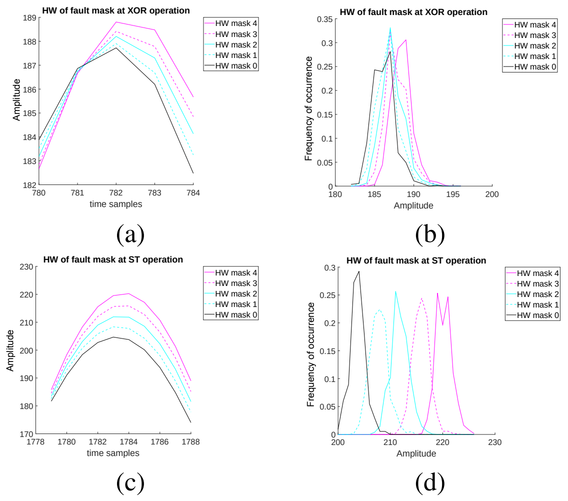


Fig. 6: The side-channel observations when comparing the correct and faulty ciphertext on byte level with a single nibble active. (a) The HW difference when observing the XOR directly (`EOR` operation) and (c) the storing of comparison (`ST` operation). The distribution of the HW values for each case are presented in (b) and (d) respectively.

inherently leaky and an attack may take place with the aid of side-channel measurements. Although the attack presented on AES have a high fault complexity, it is still below practical limits of computation, which can be addressed with modern high-speed fault injection mechanisms. Most notably, the attack on PRESENT was fairly easy to launch. Such a disparity in computational complexity also points out at weaknesses of bit-permutation based diffusion layers with respect to MDS based diffusion layers in the context of physical attacks. Further extension can explore the adaption of proposed attack to round based redundancy with techniques similar to to [28]. Adaptation of the attack for complex processor architectures is another potential direction of research.

References

[1] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," *Advances in CryptologyCRYPTO'97*, pp. 513–525, 1997.

[2] N. F. Ghalaty, B. Yuce, M. Taha, and P. Schaumont, "Differential fault intensity analysis," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*. IEEE, 2014, pp. 49–58.

[3] T. Fuhr, E. Jaulmes, V. Lomné, and A. Thillard, "Fault attacks on aes with faulty ciphertexts only," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*. IEEE, 2013, pp. 108–118.

[4] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," *IEEE Transactions on Computers*, vol. 59, no. 5, pp. 608–622, 2010.

[5] T. G. Malkin, F.-X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Fault Diagnosis and Tolerance in Cryptography*. Springer, 2006, pp. 159–172.

[6] X. Guo and R. Karri, "Recomputing with permuted operands: A concurrent error detection approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 10, pp. 1595–1608, 2013.

[7] S. Patranabis, A. Chakraborty, P. H. Nguyen, and D. Mukhopadhyay, "A biased fault attack on the time redundancy countermeasure for AES," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2015, pp. 189–203.

[8] B. Selmke, J. Heyszl, and G. Sigl, "Attack on a dfa protected aes by simultaneous laser fault injections," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2016 Workshop on*. IEEE, 2016, pp. 36–46.

[9] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pp. 19–22, 2001.

[10] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 450–466.

[11] C. Dobraunig, M. Eichlseder, T. Korak, S. Mangard, F. Mendel, and R. Primas, "Exploiting ineffective fault inductions on symmetric cryptography," Tech. Rep.

[12] M. Karpovsky, K. J. Kulikowski, and A. Taubin, "Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard," in *Dependable Systems and Networks, 2004 International Conference on*. IEEE, 2004, pp. 93–101.

[13] N. Wiersma and R. Pareja, "Safety!= Security: On the Resilience of ASIL-D Certified Microcontrollers against Fault Injection Attacks," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2017 Workshop on*. IEEE, 2017, pp. 9–16.

[14] F. Regazzoni, L. Breveglieri, P. Ienne, and I. Koren, "Interaction between fault attack countermeasures and the resistance against power analysis attacks," in *Fault Analysis in Cryptography*. Springer, 2012, pp. 257–272.

[15] B. Robisson and P. Manet, "Differential behavioral analysis," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 413–426.

[16] 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.

[17] 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," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 292–311.

[18] S. Patranabis, D. Mukhopadhyay, J. Breier, and S. Bhasin, "One Plus One is More than Two: A Practical Combination of Power and Fault Analysis Attacks on PRESENT and PRESENT-like Block Ciphers," in *FDTC*, 2017.

[19] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *Journal of Cryptographic Engineering*, vol. 5, no. 3, pp. 153–169, 2015.

[20] J. Breier, D. Jap, and S. Bhasin, "A study on analyzing side-channel resistant encoding schemes with respect to fault attacks," *Journal of Cryptographic Engineering*, vol. 7, no. 4, pp. 311–320, 2017.

[21] Y. Yao, M. Yang, C. Patrick, B. Yuce, and P. Schaumont, "Fault-assisted side-channel analysis of masked implementations," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2018, pp. 57–64.

[22] H. Eldib, M. Wu, and C. Wang, "Synthesis of fault-attack countermeasures for cryptographic circuits," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 343–363.

[23] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis," in *CHES'10*. Springer, 2010, pp. 320–334.

[24] R. Korkikian, S. Pelissier, and D. Naccache, "Blind fault attack against spn ciphers," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*. IEEE, 2014, pp. 94–103.

[25] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," in *IFIP International Workshop on Information Security Theory and Practices*. Springer, 2011, pp. 224–233.

[26] J. Jean, "TikZ for Cryptographers," https://www.iacr.org/authors/tikz/, 2016.

[27] NIST, "Submission requirements and evaluation criteria for the lightweight cryptography standardization process," https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/Draft-LWC-Submission-Requirements-April2018.pdf, 2018.

[28] J. Breier, D. Jap, and S. Bhasin, "SCADPA: Side-channel assisted differential-plaintext attack on bit permutation based ciphers," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*. IEEE, 2018, pp. 1129–1134.