Glitch-Resistant Masking Schemes as Countermeasure Against Fault Sensitivity Analysis

Victor Arribas
KU Leuven, imec-COSIC, Belgium
victor.arribasabril@esat.kuleuven.be

Thomas De Cnudde
KU Leuven, imec-COSIC, Belgium
thomas.decnudde@esat.kuleuven.be

Danilo Šijačić
KU Leuven, imec-COSIC, Belgium
danilo.sijacic@esat.kuleuven.be

***Abstract*—Fault Sensitivity Analysis is an attack on cryptographic implementations that exploits dependencies between the sensitive data and the intensity of an injected fault. Masking, an established Side-Channel Analysis countermeasure, was originally believed to resist Fault Sensitivity Analysis, until Moradi et al. presented a successful attack on several masked AES ASIC cores by leveraging Fault Sensitivity Analysis. However, the attacked masked implementations are known to be vulnerable to power analysis through glitches occurring from non-ideal gates in CMOS. This means that glitch-resistant masking schemes specifically have not been assessed against Fault Sensitivity Analysis. In this work we give a response to this matter and show that implementations protected with these glitch-resistant masking schemes provide Fault Sensitivity Analysis resistance by design. We argue our claims through a theoretical elaboration and provide further evidence through simulations for both ASIC and FPGA platforms. In our setup we give the attackers numerous, often unrealistic, advantages, only to see the attacks fail against glitch-resistant masking schemes.**

***Index Terms*—FSA, Glitch, Masking, Countermeasure, Non-Completeness.**

I. Introduction

When cryptographic algorithms are naively implemented on embedded systems, many problems surrounding the actual security of the cryptographic device may arise. A well known issue in embedded cryptosystems is the leakage of sensitive information through so called side channels. These unintentional information channels manifest themselves in the dependency between sensitive data and some observable characteristics of the implementation, e.g. the time it takes to finish a computation [1], the power consumed during an encryption [2], or the electromagnetic waves radiated [3], [4] from a device during cryptographic operations. Side-Channel Analysis (SCA) exploits information leaked through these side channels to attempt to break cryptosystems.

To harden devices against Side-Channel Analysis, masking [5], [6] is a well-studied countermeasure due to its general applicability and provable security [7]. Masking reduces the information leaked through side channels by randomizing intermediate values of the computation. The security offered by classic masking schemes was shown to deteriorate when they are implemented in hardware with non-ideal gates, i.e. gates that evaluate more than once per clock cycle, or *glitch* [8], [9]. As a result, glitch-resistant masking schemes were proposed, of which Threshold Implementations (TI) [10]–[13], Consolidated Masking Schemes (CMS) [14], Domain Oriented Masking (DOM) [15], and the Roche and Prouff masking scheme [16], [17] are some examples.

After Side-Channel Analysis, another well-known path to obtain sensitive information from cryptographic implementations is through the deliberate injection of faults during operations. Such attacks that rely on the malicious interaction of an adversary are known as Fault Attacks (FAs) [18]. A very powerful FA subtype against symmetric cryptosystems is Differential Fault Analysis (DFA) [19]. In contrast to side-channel attacks, which are related to the actual implementation of the cryptographic algorithm, DFA exploits knowledge of faulty ciphertexts and the type of injected fault in order to facilitate cryptanalysis on the target cipher.

At CHES 2010, a new type of side-channel attack was presented as Fault Sensitivity Analysis (FSA) [20]. Unlike the completely passive SCA, FSA is an actively triggered passive attack. It requires an attacker to inject faults of increasing intensity for a given input, up until a fault is detected. An advantage compared to DFA is that an FSA attacker only needs to observe the occurrence of a fault, as opposed to the actual faulty output value. This attack has already been used in practice to break devices with conventional FA countermeasures [21]–[23], even when they do not reveal faulty outputs.

Initially masking was thought to prevent FSA, until Moradi et al. [21], [24] successfully broke several masked AES implementations. One of the successfully attacked cores, named the AES_TI core, hints at securing its AES with Threshold Implementations. Despite its name this AES core did not consider the *uniformity* and *non-completeness* properties, both essential for side-channel security. Thereby, this core can not be considered secure in the presence of glitches.

Actual glitch-resistant masking schemes have thus not yet been subjected to FSA. In this work, we show both theoretically and practically that glitch-resistant masking schemes, such as TI, provide protection against both SCA and FSA. We ground this observation in several experiments that involve both ASIC and FPGA implementation. We simulate FSA attacks on different unprotected and protected S-Boxes to show their resilience against FSA. Attacks on the glitch-resistant masked implementations fail even in a scenario that is beyond worst-case in practice: an attacker with advantageous noiseless, high-precision and simulated observations does not succeed in retrieving the key.

The remainder of this paper starts with a review of FSA, a summary of the characteristics of TI, and a review of related work in Section II. We give theoretical reasons on why this masking scheme works as countermeasure against FSA in Section III. We present experiments to show that our claims hold in practice in Section IV, and the corresponding results in Section V. Finally, we conclude in Section VII.

II. Preliminaries

A. Fault Sensitivity Analysis

FSA is an actively triggered side-channel attack that can retrieve secret keys from the correlation of processed data values with their fault sensitivity (FS) [20]. The fault sensitivity is the intensity of an injected fault at which the device shows a detectable characteristic, e.g. supply voltage and/or clock frequency where the device starts outputting incorrect results.

An FSA attack is mounted in two phases: a collection or profiling phase, and a key retrieval phase. In the profiling phase, fault sensitivities of input values are collected. The FS of a value is retrieved by repeatedly encrypting a fixed plaintext. During each encryption, the intensity of the applied fault is gradually increased. The first fault intensity at which the output becomes faulty is noted as the fault sensitivity for that input. In the case of clock glitching, this corresponds to shortening the clock period momentarily to a value lower than the propagation delay caused by the input transition. This process of retrieving the FS is repeated for several different plaintexts. In the subsequent key retrieval phase, the FS information is used to find the highest correlation between the predicted FS from a key guess and the measured FS. The highest correlation then leads to the correct key [20].

Two factors that codetermine the success of an FSA attack are the resolution of the fault injection the attacker can apply, and whether or not a model for the FS and the processed data can be extracted. A clock glitch generator has shown to provide enough resolution for successfully mounting FSA attacks on both IC and FPGA implementations [25], [26]. The Hamming Weight model has been used with success [20], but was shown to not work on all S-Boxes [27]. An alternative model based on the zero-value attack was shown successful in the latter cases [22].

B. Threshold Implementations

Masking is considered secure at a certain order. If the attacker observes the first-order statistical moment of the power consumption, the implementation can be secured using first-order masking. The attacker can, however, observe the variance, or second-order statistical moment, to break the first-order masking scheme. Generally, an attacker observing the d^{th}-order statistical moment can be mitigated by a d^{th}-order masking scheme. It is known that attacking becomes exponentially harder with the masking order.

The Threshold Implementation (TI) masking scheme [10]–[13] can protect hardware implementations against SCA in the presence of glitches at any order d. Due to its efficiency in both area and latency, it has been implemented extensively.

Secret or sensitive values are shared among several players using boolean masking, and a multi-party computation protocol is employed for computations on the shares. TI is a (n, n)-threshold scheme, i.e. the secret value is distributed among n players and n of them are required to reconstruct the secret input.

A sensitive variable $Z \in \mathbb{F}_{2^m}$ is shared among n players in the form $\mathbf{Z}$ by drawing $n-1$ random coefficients S_i from a uniform distribution and constructing the final share as:

$$S_n = Z + \sum_{i=1}^{n-1} S_i \ . \qquad (1)$$

To achieve SCA security, three properties have to hold, namely:

1) *Correctness:* a shared function $\mathbf{f}$ such that $f_i(\mathbf{Z}) = Y_i$, where $i = 1 \dots n$, is correctly shared if $\sum Y_i = Y = f(Z)$,
2) *Uniformity:* the inputs of a shared function have to conform to a uniform distribution,
3) *Non-completeness:* a shared function f is d^{th}-order non-complete if any combination of up to d intermediate values of the shared function is independent of at least one input share.

Whereas correctness and uniformity are properties shared by all masking schemes, non-completeness was specifically introduced in TI to provide resistance against glitches and early propagation originating from non-ideal gates.

C. Related Work

In the original work of Li, it was argued that masking schemes could provide resistance against FSA [20]. After several masking schemes were broken using FSA [21], [22], [24], dedicated countermeasures against FSA have been proposed. These can be categorized into gate-level countermeasures and RTL-level countermeasures.

The first gate-level countermeasure against FSA was proposed by Ghalaty et al. [28]. Their proposed method is based on internally changing the circuit in order to balance out the sources of FSA leakage with low area overhead.

Another approach to counteracting FSA attacks is through RTL-level countermeasures [29]. By only allowing the results of the combinational logic to the inputs of the capturing register after its outputs have been stabilized using an enable signal, the fault sensitivity is made constant and equal to the arrival of the enable signal. This method was shown possible by Endo et al. by choosing the enable signal to be greater than the largest critical path delay of the circuit during post-manufacturing reconfiguration [30], [31].

While these countermeasures have a reasonably small impact on the area and throughput, the resulting circuits are still vulnerable to passive SCA. It is reasonable to expect an integrated circuit to require both protection against SCA and FAs and therefore, combinations of threshold implementations and FA countermeasures have been investigated in more recent works: ParTI [32] and Private Circuits-II enhanced TI [33]. Since the latter two approaches build on the properties of TI, we believe the claims of FSA resistance we make in this work can directly be applied to them.

III. The power of SCA glitch-resistance against FSA

In this section we provide theoretical insights on why glitch-resistant schemes against SCA are also a valid protection against Fault Sensitivity Analysis. We begin by giving the assumptions we make on the attacker, and proceed by explaining the concepts that lead to FSA resistance.

A. Assumptions

In order to facilitate our explanation, we make the following assumptions:

1) we assume that the attacker actually measures the timing properties of the circuit, i.e. the propagation delay through his FSA attack,
2) we assume the players to leak independently and that the requirements for the masking schemes are satisfied.

The validity of the first assumption can be explained by looking at the physical effect of the FSA attack. The attacker applies a fault and gradually increases its intensity, each time noting whether the output was correct or incorrect. After m injections, the attacker holds a list of m responses $[(\Delta_1, FI_1), ..., (\Delta_m, FI_m)]$ [27], symbolizing whether or not the fault injection led to correct or incorrect outputs under fault intensity FI_i. If $\Delta_i = 1$, we consider the computation to have finished correctly under fault intensity FI_i and $\Delta_i = 0$ otherwise. The fault intensity $\overline{FI}$, for which $FI_i = \overline{FI}$ leads to $\Delta_i = 1$, and F_{i+1} leads to $\Delta_i = 0$, is defined as the fault sensitivity for input x. This point can be considered as the last point in time for which not all output bits were valid yet, i.e. the critical propagation delay.

The independent leakage requirement of the second assumption means that, from a side channel point of view, the power consumption of the n different players of the masking scheme are independent from each other. In case the players operate in a parallel way (such as e.g. in TI), this means the power consumption of the players can be linearly decomposed into independent instantaneous power consumption traces. When the players operate in a serial way (such as e.g. in the implementations of the Roche and Prouff masking scheme [34], [35]), the instantaneous power consumption of the n players should not overlap in time. The assumption that the requirements of the masking scheme are satisfied is straightforward, as otherwise even the side-channel security can not be guaranteed.

B. On Non-Completeness

As pointed out in [20], signal transitions in a device are data-dependent, and thus the fault sensitivities as well. Following the same principle as the previous work uses to prove the data-dependency of the signal timing delay, we prove that masking schemes fulfilling the non-completeness property are inherently secure against FSA.

To do this, we use the following example of a TI shared AND gate. Given the shares of the two inputs x and y as (x_1, x_2, x_3) and (y_1, y_2, y_3), then the shares (z_1, z_2, z_3) of the output z are computed as:

$$\begin{aligned} z_1 &= f_1(x_1, x_2, y_1, y_2) = x_1y_1 + x_1y_2 + x_2y_1 \, , \\ z_2 &= f_2(x_2, x_3, y_2, y_3) = x_2y_2 + x_2y_3 + x_3y_2 \, , \\ z_3 &= f_3(x_1, x_3, y_1, y_3) = x_3y_3 + x_3y_1 + x_1y_3 \, . \end{aligned} \tag{2}$$

Fig. 1 presents a generic scheme of a three shares threshold implementation of a non-linear operation. Any TI implementation follows a similar structure, which means the claims presented below can be generalized.

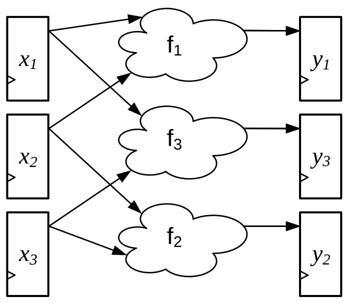

Fig. 1: General structure of a threshold implementation

All three combinational circuits are computed in parallel and independently, as stated in assumption 2, where every one of them has a different propagation delay. Thus, the fault sensitivity targets solely the circuit with the longest (critical) propagation delay. Let us assume this holds for the first combinational circuit:

$$FS_1(x_{1,2}, y_1) \geq FS_2(x_{2,3}, y_2) \geq FS_3(x_{1,3}, y_1) \tag{3}$$

We analyze in detail the data-dependency of the transitions of the first share from the previous example, looking at the timing delay of the signals. This is depicted in Fig. 2, where T_c represents the timing delay of a component c, either wire or gate, and, in case of a fanout wire, T^i refers to the i-th branch.

As shown in [20], the propagation delay of a transition in an XOR gate is determined by the longest propagation delay at its inputs, independently of the value of the inputs. Authors also show that the propagation delay of an AND gate is predisposed by the values on its inputs before and after each transition. For example, arrival of the first logic *zero* at the inputs determines the output value, hence the propagation delay, regardless of the other input bits. On the contrary, setting one of the inputs to a logic *one* can not determine the propagation delay through the AND gate until the rest of the inputs have been updated. Therefore, regardless of the initial (reset) value of the AND gate the more logic zeros the new input contains, the shorter the propagation delay. In other words, for each input transition, the propagation delay is directly correlated with the Hamming Weight (HW) of the inputs. Reversely, OR gates have the logic

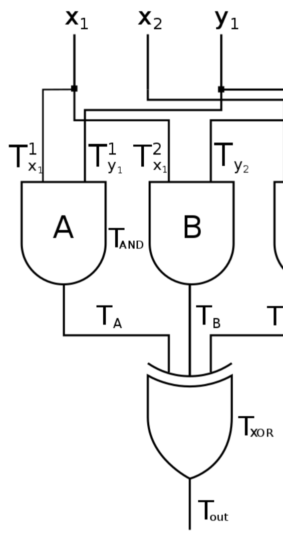

Fig. 2: First share combinational circuit from the 3-share AND gate en Eqn. 2

one as the dominant signal and leads to the reverse but dual correlation between the two.

We assume the delay of any share of the variable x to be greater than the ones from y. Therefore, we can express the timing delays of the outputs of the AND gates from Fig. 2 as follows.

$$T_A = \begin{cases} T^1_{y_1} + T_{AND} \text{ (if } y_1 = \text{'0')} \\ T^1_{x_1} + T_{AND} \text{ (if } y_1 = \text{'1')} \end{cases}$$
$$T_B = \begin{cases} T_{y_2} + T_{AND} \text{ (if } y_2 = \text{'0')} \\ T^2_{x_1} + T_{AND} \text{ (if } y_2 = \text{'1')} \end{cases} \quad (4)$$
$$T_C = \begin{cases} T^2_{y_1} + T_{AND} \text{ (if } y_1 = \text{'0')} \\ T_{x_2} + T_{AND} \text{ (if } y_1 = \text{'1')} \end{cases}$$

It is straightforward to see that the side-channel information leaked from attacking the first combinational logic cloud only reveals information on two out of the three shares of the secret (x_1, x_2 and y_1, y_2). FSA targets exclusively the longest propagation delay, which means that it is not possible to get additional information from a different combinational cloud and combine it with previous information to retrieve the secret.

Hence, we can conclude that any threshold implementation fulfilling the *non-completeness* property is secure against FSA. Both the CMS and DOM sharing schemes inherit non-completeness to provide security in the presence of glitches, which means that these schemes are also secure against FSA.

C. *On Uniformity*

It is important to note that the example above does not have a uniform output sharing. This property is needed to provide first-order SCA security when the output is subsequently used in a nonlinear computation. We emphasize that, as demonstrated above, it is not needed to provide FSA security. For a scheme to provide security against FSA it suffices that the *non-completeness* property is satisfied. It is certainly advantageous to use a uniform sharing for larger systems so that the scheme provides security against both FSA and first-order SCA.

IV. Experiments

To illustrate the theoretical insights of our claims, we simulate the actual FSA attack using two representative TI implementations of S-Boxes. We discuss the details of our experimental setup and the two stages of the FSA attack in the remainder of this section.

A. *Setup*

We rely on post place-and-route simulations, in both ASIC and FPGA. This way we are able to obtain perfectly aligned noiseless observations with a timing precision of 1ps. As we extract the data-dependent propagation delays from the simulated data, we do not have to take into account any jitter introduced by the glitching apparatus. Furthermore, we observe target circuits in isolation from any other components by directly calculating the propagation delay of this circuit for a given input. Therefore we are arming our adversary with unrealistically clean measurements that are likely unattainable in practice.

1) ASIC: We synthesize designs using a 45nm open-source standard cell library from NanGate. It relies on state of the art Composite Current Source models which provide 1ps precision for timing simulations. We use Synopsys Design Compiler for logic synthesis; Cadence Innovus for placement, routing and RC extraction; and MentorGraphics QuestaSim for timing simulation. Design flow automation and data processing is done using the CASCADE [36] framework.

2) FPGA: We run the experiments with the Xilinx tools, using the Isim environment to perform the simulations. We choose a Spartan-6 XCLX75-2CG484 as the target platform for our experiments.

B. *Profiling phase*

In the seminal paper Li et al. [20] use the value zero as the reset value between two fault injections. We leverage the small size of our target circuits, and the high speed and high precision of our simulated approach, to show that this is not necessarily the best value to start the attack from. Instead of only collecting fault sensitivities for up to 2^N input values from an N-bit circuit, we simulate all possible $2^{2\cdot N}$ transitions as well. From there we differentiate 2^N profiles, using all 2^N reset values. Consequently, each profile has 2^N traces. The double exponential complexity of this approach makes it infeasible for circuits with larger number of inputs. From this approach we obtain profiles with the maximal correlation between fault sensitivities and input data, i.e. profiles which are optimal for the attacker. We then proceed to key recovery using the most effective profile.

C. Key Recovery

We attack a dummy cipher round consisting of a single S-Box and key addition, as depicted in Fig. 3. As in the profiling phase, the small input space of our targets allows us to collect all possible ciphertexts. Note that since the delay of XOR gates is not data dependent, as pointed out in [20], the XOR circuit need not be included in the profiling.

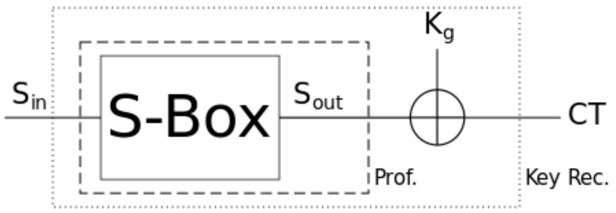

Fig. 3: Circuit under attack, unprotected.

The profiling is made over the inputs of the S-Box, so we attack the inputs of the circuit. Eqn. 5 presents the function used to get the predicted fault sensitivities (FS_g), using the ciphertext (CT) and the key guess (K_g):

$$FS_g = HW(SBox^{-1}(CT \oplus K_g)) \ . \qquad (5)$$

The guessed fault sensitivities are subsequently correlated with the observed ones, where the maximum correlation should correspond with to the correct key.

For the protected implementations, the circuit under attack has similar structure, in this case shared over three shares, as shown in Fig. 4. We allow the attacker to only observe unshared plaintext-ciphertext pairs, i.e. the attacker sees the dotted domain as a black box. On the contrary, we allow the attacker to see the exact sharing to do the profile, i.e. what is outside of the dashed box. This means that the attacker is able to get the exact shared input that corresponds to the profile with the highest correlation. The aim of these experiments is to

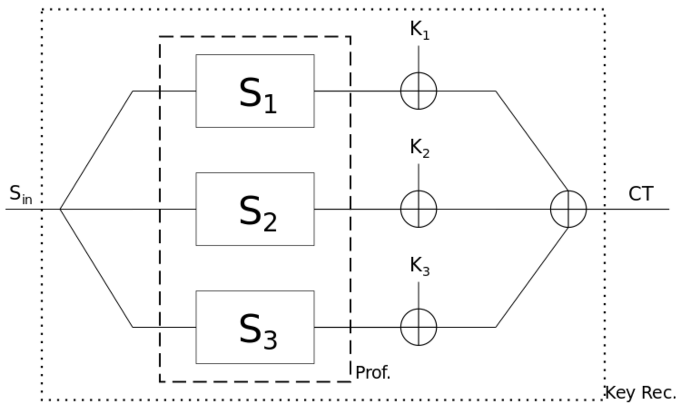

Fig. 4: Circuit under attack, protected.

immerse our S-Boxes into the worst case scenario, to test how the proposed countermeasure responds. In reality, attackers are very unlikely to attain such detailed profiling nor have such a stripped down target circuit. Moreover, the attackers would not be able to cover the whole state of inputs in a real design, which means that the attacks are limited to the amount of measurements they can procure.

V. Results

In this section we present the results of our experiments. We perform our experiments in both ASIC and FPGA to get a deeper and broader view of the effects of fault sensitivity analysis. As expected, the experiments show that the attacks fail against the TI implementations, even given the highly favorable conditions for the attack. We present results for two target circuits that illustrate different scenarios: FSA resistance with a conventional TI implementation, and FSA resistance with a non-uniform implementation.

A. Present

Our first target is the 4-bit S-Box of the lightweight ISO standard PRESENTblockcipher. For the protected design we use the TI approach from Poschmann et al. [37]. This design is decomposed into two quadratic S-boxes F and G forming a pipelined implementation. As already mentioned before, FSA targets the longest propagation delay. To comply with this, we target just one of the two stages to gather the profiles, choosing the first stage F for our experiments. The unprotected implementation has a total of 2^4 inputs and $2^{2 \cdot 4}$ possible transitions that can occur on the inputs. The protected implementation is split into three shares per variable, in accordance with the TI principles. The total number of inputs (resp. outputs) is thus 12 (4 sensitive bits masked with 3 shares), resulting in the total of $2^{2 \cdot 12}$ transitions.

Fig. 5 shows correlation values between the HW of the input and the FS for profiles with different reset values. Prominent correlation peaks appear in the case of the unprotected implementation. In case of ASIC as target, the correlation peaks at 0.75 for reset value 0b1011, whereas in the case of FPGA as target, peak value is as high as 0.55 corresponding to reset value 0b0000. It is interesting to see that this peak is closely followed by the 0.51 value for reset value 0b1111. We will discuss this in the later Sec. VI.

In case of the protected implementation, the attacker is allowed to profile the S-Box for all 2^{12} shared reset values to maximize the attacking precision. Still, correlation peaks do not surpass 0.41 for ASIC with reset value 0x798, nor 0.39 for FPGA with reset value 0x821. In comparison, the correlation values corresponding to the all-zero reset vector are 0.08 for ASIC and 0.24 for FPGA. This demonstrates that the all-zero reset (or initial state) vector may be an effective one, but it is not necessarily the best one.

At first glance it would stand to reason to maximize the number of collected traces by looking at the correlation between the HW and the FS across all transitions. Nevertheless, doing so results in consistently lower correlation values across all of our experiments. Fig. 6 provides a graphical explanation of this.

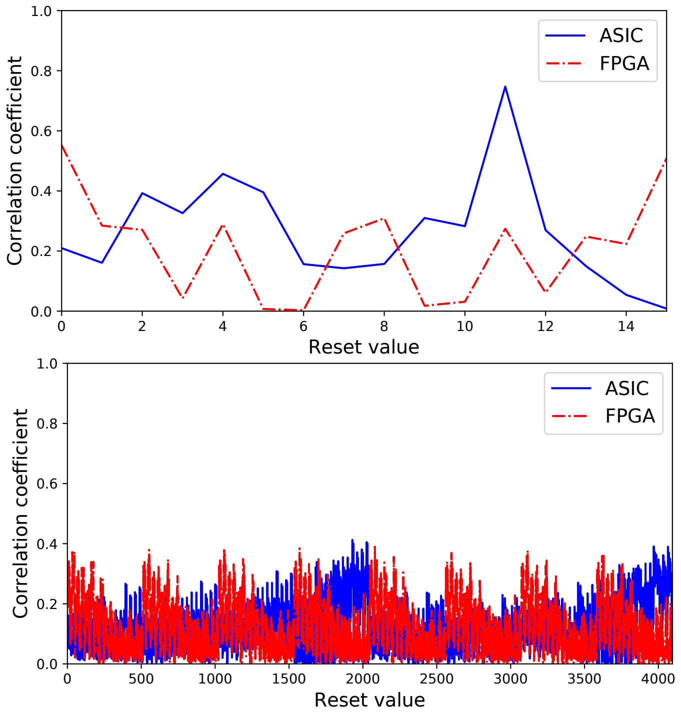

Fig. 5: ASIC vs FPGA correlations for all profiles of the unprotected (top) and protected (bottom) PRESENT S-Box.

Similarly, we tried using the correlation of Hamming Distance (HD) and FS in both scenarios, only to obtain significantly lower correlation values. We attribute this to the nature of the side channel we attempt to exploit compared to the classical SCA. Let us observe a 2-input AND gate, transitioning first 0b00→0b11, and then back 0b11→0b00. The power consumption of the two transitions will normally be comparable as it is proportional to the amount of current needed to charge and discharge the circuit's capacitances. For FSA however, the differences in the input-dependent propagation delays matter. Hence a transition ending in 0b00 will on average be faster in an AND gate [20]. As a consequence of this discrepancy, the HW of the input value at the end of a transition is the most relevant observation as opposed to the HD.

Fig.7 illustrates the key recovery attacks on the unprotected and protected implementations. We can see that the guessed value (×) coincides with the target value (■) of the key for the unprotected implementation, representing a successful attack. This is not the case for the TI implementation, where even though the attacker is given the best reset value and unrealistically accurate profiling, correlations for each key guess remain dauntingly small.

B. KECCAK

Our second target is the 5-bit KECCAK χ permutation. We use the TI implementation from [38] for our experiments. It has three 5-bit shares along with a 4-bit input for randomness. By fixing the random bits to 0b0000 we disable the uniformity at the output of this implementation. We thereby test the claim

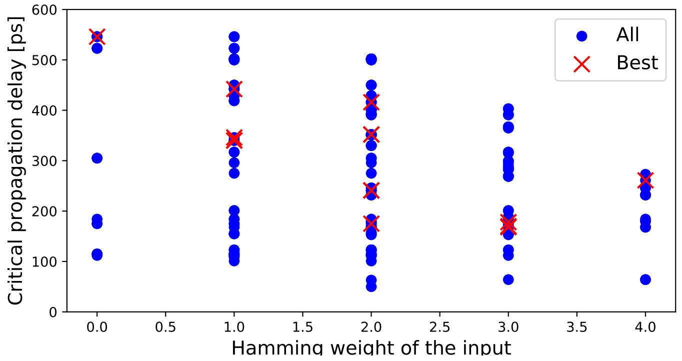

Fig. 6: ASIC FS of the unprotected PRESENT S-Box for the highest correlation profile.

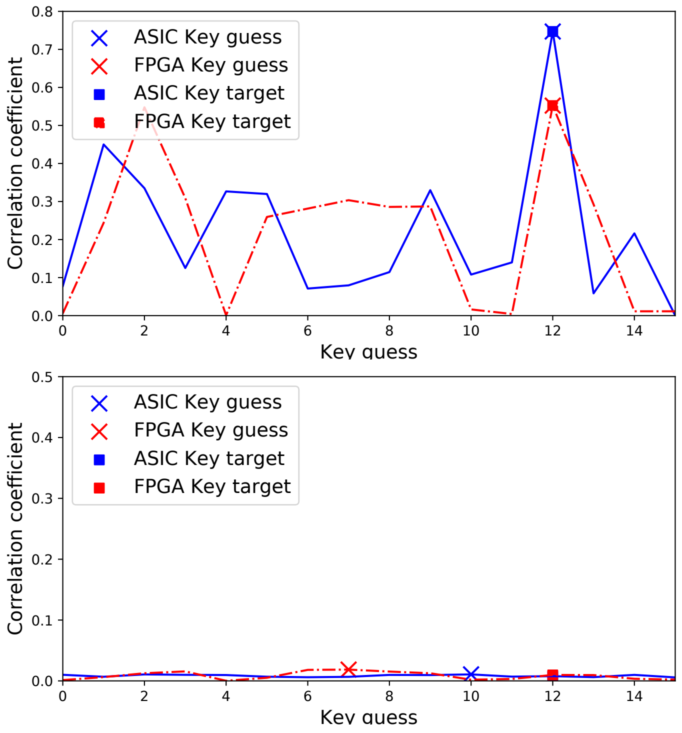

Fig. 7: FSA attack on the unprotected (top), and TI (bottom) implementation of the PRESENT S-Box.

that the non-completeness is sufficient to provide security against FSA attacks, as opposed to both non-completeness and a uniformly shared output. A qualitative difference compared to the previous experiments is that we do not exhaust all of the $2^{2 \cdot 3 \cdot 5}$ input transitions. Instead we pick $2^{2 \cdot 12}$ random transitions to match the previous experiment in volume. Note that the results we obtain are consistent over different pools of random inputs.

The results of the FSA attack are shown in Fig. 8, which shows that despite the lack of uniformity at the output of the TI, the non-completeness is sufficient to provide protection against FSA.

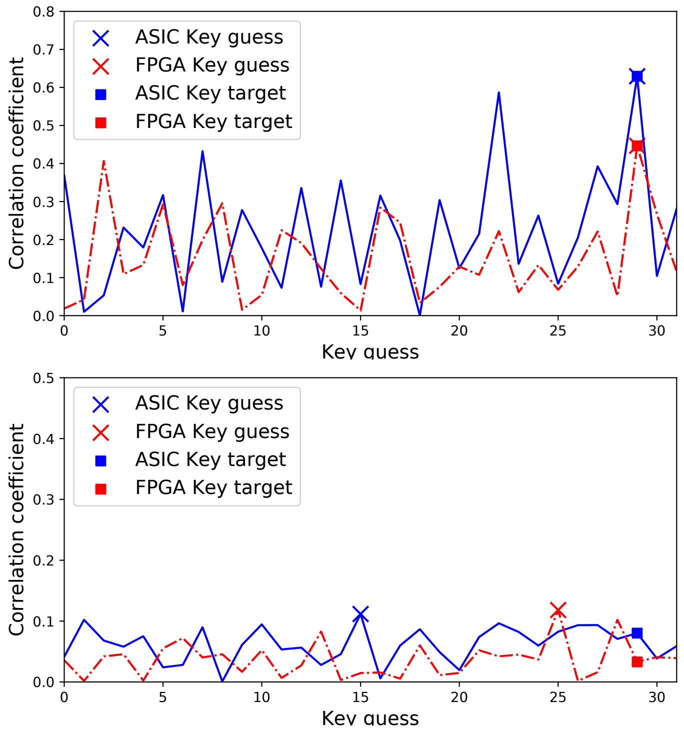

Fig. 8: FSA attack on the unprotected (top), and TI (bottom) implementation of the KECCAK S-Box.

VI. Discussions

A. The Roche and Prouff Scheme

The Roche and Prouff masking scheme [16], [17] offers an alternative approach at masking hardware implementations at any order d in the presence of glitches. Shamir's secret sharing scheme [39] is used to share the sensitive variables, and computations on the resulting shares are performed using the BGW secure multi-party computation protocol [40].

The protocol for a multiplication (or AND gate) is split into two stages to achieve glitch-resistance [34]. In neither of the two stages is there an intermediate variable that depends on any unmasked inputs, which is exactly the definition of non-completeness. Thus, we argue that this sharing scheme is also resistant against FSA given that non-completeness is fulfilled. By extension, any scheme that is based on a glitch-resistant masking scheme benefits from FSA resistance, including the schemes by Reparaz et al. [41] and Seker et al. [42].

B. ASIC vs FPGA

We simulated attacks on two substantially different platforms, namely a standard-cell library for ASIC development, and an FPGA board. Albeit their difference, all obtained results are consistent across the target circuits. The first difference is that the propagation delays in FPGA are much larger. For the ASIC example, Fig. 6 shows that the propagation delays of the unprotected PRESENT S-Box are spread between 100–600ps. The same circuit on the FPGA leads to propagation delays that are spread between 8200–9200ps. An older technology and a higher complexity of the configurable LUTs that realize logic on an FPGA lead to this absolute difference.

A second difference stems from the 5to1 (5-input, 1-output) LUTs of the target FPGA, which allow merging multiple logic gates, leading to a lower number of LUTs than standard cells needed for the same circuit. The fine-grained standard cells of ASICs lead to a higher spread in propagation delays. Hence, a finer distinction can be made in the latter between different input values, resulting in higher correlation peaks.

Furthermore, as depicted in Fig. 5, the choice of the best profile, i.e. the best reset value, depends on the target platform. In our experiments, where the target circuit is small enough to fit into one FPGA LUT, choosing an all-zero, or all-one vector as the reset value consistently leads to the best profile. For ASIC circuits, which rely on less flexible atomic cells to implement combinational logic, the best profile greatly differs from design to design. So much that even for the same RTL code synthesized under a different set of constraints, the best reset value would change.

C. Reset Value

In the original work, Li et al. [20] use an all-zero vector as the reset value. Although this is a natural choice, and coincidentally often fruitful for the FPGA implementations, we have shown that it is not necessarily the best one. In our experiments we have seen that even when attacking the unprotected implementations, the key can not be recovered for all reset values. From the perspective of a practical attacker, choosing an arbitrary reset value is not trivial and would at best require the application of a well chosen input to achieve the desired initial state of the circuit. On top of that, when attacking a real life design it is not possible to exhaust all input transitions in search of the best reset value. Therefore, by choosing the best reset value, we are yet again giving advantage to the attackers, only to see them fail.

VII. Conclusion

In this paper we investigated the resistance of masking schemes against Fault Sensitivity Analysis attacks. We proved both theoretically and experimentally that glitch-resistant masking schemes featuring the non-completeness property are secure against FSA, even under extremely favorable conditions for an attacker. We strengthened our arguments through simulations in both ASIC and FPGA environments. Our elaboration showed that resistance against FSA comes directly from the glitch-resistant nature of the schemes and explained why Moradi et al. [21], [24] succeeded in breaking the masked AES implementations, which were actually insecure in the presence of non-ideal gates.

Acknowledgments

This project has received partial funding from the European Unions Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 643161, and NIST Research Grant 60NANB15D346; as well as partial funding from Intel. Thomas De Cnudde is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

[1] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 1109. Springer, 1996, pp. 104–113.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 1666. Springer, 1999, pp. 388–397.

[3] J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): measures and counter-measures for smart cards," in *E-smart*, ser. Lecture Notes in Computer Science, vol. 2140. Springer, 2001, pp. 200–210.

[4] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *CHES*, ser. Lecture Notes in Computer Science, vol. 2162, no. Generators. Springer, 2001, pp. 251–261.

[5] L. Goubin and J. Patarin, "DES and differential power analysis (the "duplication" method)," in *CHES*, ser. Lecture Notes in Computer Science, vol. 1717. Springer, 1999, pp. 158–172.

[6] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 1666. Springer, 1999, pp. 398–412.

[7] E. Prouff and M. Rivain, "Masking against Side-Channel Attacks: A Formal Security Proof," in *Advances in Cryptology EUROCRYPT 2013*, ser. Lecture Notes in Computer Science, T. Johansson and P. Q. Nguyen, Eds. Springer Berlin Heidelberg, 2013, vol. 7881, pp. 142–159.

[8] S. Mangard, T. Popp, and B. M. Gammel, "Side-channel leakage of masked CMOS gates," in *CT-RSA*, ser. Lecture Notes in Computer Science, vol. 3376. Springer, 2005, pp. 351–365.

[9] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *CHES*, ser. Lecture Notes in Computer Science, vol. 3659. Springer, 2005, pp. 157–171.

[10] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *ICICS*, ser. Lecture Notes in Computer Science, vol. 4307. Springer, 2006, pp. 529–545.

[11] S. Nikova, V. Rijmen, and M. Schläffer, "Secure hardware implementation of non-linear functions in the presence of glitches," in *ICISC*, ser. Lecture Notes in Computer Science, vol. 5461. Springer, 2008, pp. 218–234.

[12] ——, "Secure hardware implementation of nonlinear functions in the presence of glitches," *J. Cryptology*, vol. 24, no. 2, pp. 292–321, 2011.

[13] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Higher-order threshold implementations," in *ASIACRYPT (2)*, ser. Lecture Notes in Computer Science, vol. 8874. Springer, 2014, pp. 326–343.

[14] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *CRYPTO (1)*, ser. Lecture Notes in Computer Science, vol. 9215. Springer, 2015, pp. 764–783.

[15] H. Gross, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," *IACR Cryptology ePrint Archive*, vol. 2016, p. 486, 2016.

[16] E. Prouff and T. Roche, "Higher-order glitches free implementation of the AES using secure multi-party computation protocols," in *CHES*, ser. Lecture Notes in Computer Science, vol. 6917. Springer, 2011, pp. 63–78.

[17] T. Roche and E. Prouff, "Higher-order glitch free implementation of the AES using secure multi-party computation protocols - extended version," *J. Cryptographic Engineering*, vol. 2, no. 2, pp. 111–127, 2012.

[18] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults (extended abstract)," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 1233. Springer, 1997, pp. 37–51.

[19] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 1294. Springer, 1997, pp. 513–525.

[20] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis," in *CHES*, ser. Lecture Notes in Computer Science, vol. 6225. Springer, 2010, pp. 320–334.

[21] 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 *CHES*, ser. Lecture Notes in Computer Science, vol. 6917. Springer, 2011, pp. 292–311.

[22] O. Mischke, A. Moradi, and T. Güneysu, "Fault sensitivity analysis meets zero-value attack," in *FDTC*. IEEE Computer Society, 2014, pp. 59–67.

[23] F. Schellenberg, M. Finkeldey, N. Gerhardt, M. Hofmann, A. Moradi, and C. Paar, "Large laser spots and fault sensitivity analysis," in *HOST*. IEEE Computer Society, 2016, pp. 203–208.

[24] A. Moradi, O. Mischke, and C. Paar, "One attack to rule them all: Collision timing attack versus 42 AES ASIC cores," *IEEE Trans. Computers*, vol. 62, no. 9, pp. 1786–1798, 2013.

[25] S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "An on-chip glitchy-clock generator for testing fault injection attacks," *J. Cryptographic Engineering*, vol. 1, no. 4, pp. 265–270, 2011.

[26] ——, "A configurable on-chip glitchy-clock generator for fault injection experiments," *IEICE Transactions*, vol. 95-A, no. 1, pp. 263–266, 2012.

[27] F. Melzani and A. Palomba, "Enhancing fault sensitivity analysis through templates," in *HOST*. IEEE Computer Society, 2013, pp. 25–28.

[28] N. F. Ghalaty, A. Aysu, and P. Schaumont, "Analyzing and eliminating the causes of fault sensitivity analysis," in *DATE*. European Design and Automation Association, 2014, pp. 1–6.

[29] Y. Li, K. Ohta, and K. Sakiyama, "Toward effective countermeasures against an improved fault sensitivity analysis," *IEICE Transactions*, vol. 95-A, no. 1, pp. 234–241, 2012.

[30] S. Endo, Y. Li, N. Homma, K. Sakiyama, K. Ohta, and T. Aoki, "An efficient countermeasure against fault sensitivity analysis using configurable delay blocks," in *FDTC*. IEEE Computer Society, 2012, pp. 95–102.

[31] S. Endo, Y. Li, N. Homma, K. Sakiyama, K. Ohta, D. Fujimoto, M. Nagata, T. Katashita, J. Danger, and T. Aoki, "A silicon-level countermeasure against fault sensitivity analysis and its evaluation," *IEEE Trans. VLSI Syst.*, vol. 23, no. 8, pp. 1429–1438, 2015.

[32] T. Schneider, A. Moradi, and T. Güneysu, "Parti - towards combined hardware countermeasures against side-channel and fault-injection attacks," in *CRYPTO (2)*, ser. Lecture Notes in Computer Science, vol. 9815. Springer, 2016, pp. 302–332.

[33] T. D. Cnudde and S. Nikova, "More efficient private circuits ii through threshold implementations," in *FDTC*. IEEE Computer Society, 2016, pp. –.

[34] A. Moradi and O. Mischke, "On the simplicity of converting leakages from multivariate to univariate - (case study of a glitch-resistant masking scheme)," in *CHES*, ser. Lecture Notes in Computer Science, vol. 8086. Springer, 2013, pp. 1–20.

[35] T. D. Cnudde, B. Bilgin, O. Reparaz, and S. Nikova, "Higher-order glitch resistant implementation of the PRESENT s-box," in *BalkanCryptSec*, ser. Lecture Notes in Computer Science, vol. 9024. Springer, 2014, pp. 75–93.

[36] D. Šijačić, "Computer aided side channel analysis design environment (cascade)." [Online]. Available: https://github.com/dsijacic/CASCADE

[37] A. Poschmann, A. Moradi, K. Khoo, C. Lim, H. Wang, and S. Ling, "Side-channel resistant crypto for less than 2, 300 GE," *J. Cryptology*, vol. 24, no. 2, pp. 322–345, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00145-010-9086-6

[38] J. Daemen, "Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing," in *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, ser. Lecture Notes in Computer Science, W. Fischer and N. Homma, Eds., vol. 10529. Springer, 2017, pp. 137–153.

[39] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[40] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)," in *STOC*. ACM, 1988, pp. 1–10.

[41] O. Reparaz, L. De Meyer, B. Bilgin, V. Arribas, S. Nikova, V. Nikov, and N. P. Smart, "CAPA: the spirit of beaver against physical attacks," *IACR Cryptology ePrint Archive*, vol. 2017, p. 1195, 2017.

[42] O. Seker, T. Eisenbarth, and R. Steinwandt, "Extending glitch-free multiparty protocols to resist fault injection attacks," *IACR Cryptology ePrint Archive*, vol. 2017, p. 269, 2017.