Leakage Power Analysis in Different S-Box Masking Protection Schemes

Javad Bahrami*, Mohammad Ebrahimabadi*, Jean-Luc Danger†, Sylvain Guilley‡, Naghmeh Karimi*
*University of Maryland Baltimore County, United States
†LTCI, Télécom Paris, Institut polytechnique de Paris, France
‡Secure-IC, France

***Abstract*—Internet-of-Things (IoT) devices are natural targets for side-channel attacks. Still, side-channel leakage can be complex: its modeling can be assisted by statistical tools. Projection of the leakage into an orthonormal basis allows to understand its structure, typically linear (1st-order leakage) or non-linear (sometimes referred to as glitches). In order to ensure cryptosystems protection, several masking methods have been published. Unfortunately, they follow different strategies; thus it is hard to compare them. Namely, ISW is constructive, GLUT is systematic, RSM is a low-entropy version of GLUT, RSM-ROM is a further optimization aiming at balancing the leakage further, and TI aims at avoiding, by design, the leakage arising from the glitches. In practice, no study has compared these styles on an equal basis. Accordingly, in this paper, we present a consistent methodology relying on a Walsh-Hadamard transform in this respect. We consider different masked implementations of substitution boxes of PRESENT algorithm, as this function is the most leaking in symmetric cryptography. We show that ISW is the most secure among the considered masking implementations. For sure, it takes strong advantage of the knowledge of the PRESENT substitution box equation. Tabulated masking schemes appear as providing a lesser amount of security compared to unprotected counterparts. The leakage is assessed over time, i.e., considering device aging which contributes to mitigate the leakage differently according to the masking style.**

I. Introduction

Symmetric key cryptography is widely deployed to encrypt and decrypt large amounts of data. AES [1] and PRESENT [2] are two examples of such algorithms. They must be protected against the stealthy extraction of their secret key.

In practice, side-channel attacks are prominent threats against the implementation of such algorithms. As a response, random masking schemes have been put forward. They consist in making sure that each and every net in the netlist is adequately randomized by an unpredictability mask. Therefore, adversaries with the capability to exploit only one measurement point are doomed to failure in their attack attempt. However, it has been shown that even such stringent protection strategy is not devoid of pitfalls. Indeed, races in the combinational logic are responsible for "out-of-order" gates evaluation. This can result in "glitches", i.e., spurious transitions that occur on masked data and which can be responsible for transient unmasking. Specialized attacks have even been set up in this respect, for instance [3, 4]. For this reason, specific gate-level masking schemes that are aware of the presence of glitches have been promoted [5]. The so-called "threshold implementation" paradigm (TI [6, 7, 8]) for masking scheme aims at addressing the problem at large.

This topic is currently being addressed at the normative level, as attested for instance by the NIST "Masked Circuits for Block-ciphers" initiative [9].

The abovementioned spurious glitches do occur naturally in hardware implementations. As a non-functional activity, they are responsible for avoidable power loss. However, from a security standpoint, they can also be responsible for surreptitious information leakage.

Two strategies have been explored in literature:

- Conservative, by assuming that all glitches are potentially disclosing sensitive information. The approach is therefore to eliminate glitches, as for instance in [10].
- Resilient, by tolerating glitches, but making sure that they are harmless from an information leakage standpoint.

Facing such a challenge and diversity of approaches (glitch-aware vs. non-glitch-aware), it is paramount to get some guidance regarding which implementation is truly resistant. *We seek a quantitative answer based on a leakage spectral analysis.*

II. Preliminaries

A. Power Analysis Attack

Cryptographic primitives have been shown to be highly vulnerable to power Side-Channel Analysis (SCA) attacks [11], among which Correlation Power Analysis (CPA) [12] has received the lion's share of attention where the adversary retrieves the secret keys via evaluating the Pearson's correlation between the extracted power traces and the result of a hypothetical power function depending on the guessed key. Thereby, deploying an appropriate leakage model is of utmost importance to successfully retrieve the correct key.

The countermeasure to tackle the power SCA attacks can be categorized into the two main groups of *hiding* and *masking*. The former aims at reducing the dependency between the power consumption and processed data, while the latter tries to randomize the intermediate values. Hiding countermeasures seem less secure than the masking counterparts. For example, dual-rail logic suffers from process variations that result in mismatches between rails. Also, WDDL (a known hiding scheme) has the "early propagations" issue. On the other hand, masking schemes, and in particular Boolean masking, are more popular. However, most of the masking schemes ignore glitches occurring due to race conditions in combinational logics; thus may not always provide the desired level of security when implemented in hardware.

In practice, in a dth-order Boolean masking scheme, any secret-dependent intermediate variable x is presented by randomly selected $d+1$ shares x_0, x_1, ..., x_d , where $x_0 \oplus x_1 \oplus ... \oplus x_d = x$. On the other hand, an attack in which the adversary can get information about d points in her measurements is called a dth-order attack. Note that the implementations which are protected against dth-order attacks can be still vulnerable to

978-3-9819263-6-1/DATE22/©2022 EDAA

higher-order attacks ($d+1$-order). So, even implementations that are protected might be vulnerable if the attacker launches a stronger attack. Accordingly, considering the variability of masking schemes in the literature, this paper opts to compare a number of known masking schemes regarding their leakage to the first or higher-order attacks.

B. Device Aging

Device Aging can cause serious performance degradation and even device failures in extreme cases. Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI) [13] are the two most prominent aging mechanisms. NBTI (one class of BTI) [14] affects PMOS transistors, while PBTI (another class of BTI) and HCI affects NMOS devices.

BTI Aging: A PMOS (NMOS) transistor goes under two phases of NBTI (PBTI) depending on its operating condition [15]. The first phase, i.e., $stress$, occurs when the related transistor is "ON". Here, charges are trapped at the Si-SiO$_2$ interface and lead to an increase of the threshold voltage. The second phase, $recovery$, occurs when the transistor is off. In this phase, the charges trapped in the stress phase are partially removed, and thus the threshold voltage (V_{th}) drift that occurred during the stress phase partially recovers. The impact of BTI depends on the supply voltage, temperature, physical parameters of the transistor under stress, and stress time. Fig. 1 depicts the V_{th} drift of a PMOS transistor when it is continuously under stress for 6 months versus the case that it experiences stress and recovery phases every other month.

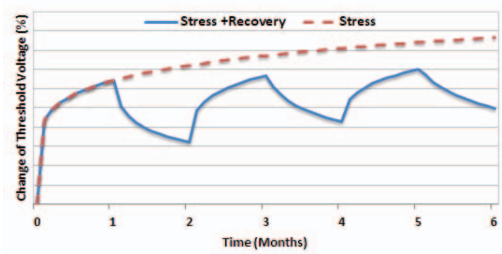

Fig. 1: NBTI-induced V_{th} drift of a PMOS transistor. Values on the Y-axis are not shown to make the figure generic and technology independent.

HCI Aging: HCI happens in an NMOS when hot carriers are injected into the gate dielectric during transistor switching and remain there. HCI is a function of switching activity; degrading the circuit by shifting the threshold voltage and drain current of stressed transistors. The threshold voltage drift induced by HCI depends on the activity factor of the transistor under stress, its temperature, clock frequency, and usage duration [13].

C. Masked S-Box Leakage

In theory, masking styles do not leak. However, in practice, security hypotheses can fail, and therefore some leakage may appear as a resurgence [16]. In the scope of this paper, we study logical effects of leakage, in that our SPICE simulations are at netlist level but do not involve any parasitics extraction whatsoever. As a matter of fact, we will show that all known masked implementations do leak. *The question we tackle is "how much do they leak"?*

III. Walsh-Hadamard Based Leakage Analysis

A. Measurement of Dataset

In order to analyse the leakage methodically, we extract traces collected from this simple experimental setup:

- Initial value is random encoding of a fixed constant value; here $(0000)_2 \in \mathbb{F}_2^4$ where we consider the nibble-oriented substitution box (S-Box) of PRESENT cipher;
- Final value is a random encoding of a text t ($t \in \mathbb{F}_2^4$).

This way of collecting traces ensures that both leakages in *values* (e.g., Hamming weight) and in *transitions* (e.g., Hamming distance) are captured. Indeed, the tranistion to a value x from initial value $(0000)_2$ is $x \oplus (0000)$, i.e., x itself.

B. Theoretical Tool for Spectral Analysis

Glitches are usually modeled informally. In practice, it is indeed hard, from an analytical standpoint, to state exactly when a glitch occurs and what sensitive information it leaks. However, some signal processing tools can assist such analysis, by decomposing the power signals in the unmasked input dependent basis. Spectral techniques are required in this respect, as for instance put forward in [17]. In this paper, the Fourier basis is leveraged. This technique allows to separate contributions from (first-order unmasked text t) leakages. The basis is denoted as ψ_u, for all vector u of n-bits (for PRESENT, $n=4$). The role of u is reciprocal compared to t in the Fourier transform. By definition, ψ_u function is $t \in \mathbb{F}_2^n \mapsto \psi_u(t) = \frac{1}{2^{n/2}}(-1)^{u \cdot t}$, where the operation $u \cdot t$ denotes the canonical scalar product, that is $u \cdot t = \bigoplus_{i=1}^n u_i \wedge t_i$. This basis is orthonormal, in that $\langle \psi_i \mid \psi_j \rangle = 0$ if $i \neq j$ and 1 otherwise. There is a simple interpretation of the Fourier basis: vector ψ_u represents the interactions between bits of the input corresponding to the 1 inputs in u (the so-called support of u).

Obviously, we ignore the zero-th component, which is the waveform average shape. But we exploit all the nonzero components.

Let us denote by $f(t)$ the leakage corresponding to (unmasked) S-Box input $t \in \mathbb{F}_2^n$. We present the glitch analysis on scalar leakage, but clearly the presentation naturally extends to multi-variate context (analyses are conducted in parallel).

The decomposition of the Fourier basis is as follows:

Lemma 1. *The leakage function $f : t \in \mathbb{F}_2^n \mapsto f(t) \in \mathbb{R}$ can be decomposed in the Fourier basis as: $f(t) = \sum_u \langle f \mid \psi_u \rangle \psi_u(t) = \frac{1}{2^{n/2}} \sum_u a_u (-1)^{t \cdot u}$, where $a_u = \frac{1}{2^{n/2}} \sum_t f(t)(-1)^{t \cdot u}$ is the Walsh-Hadamard transform of f.*

We recall the Parseval's identity: $\sum_t f^2(t) = \sum_u a_u^2$. Therefore, we get the following result:

$\sum_{u \neq 0} a_u^2 = \sum_t f^2(t) - \left(\sum_t f(t) \right)^2$, i.e., the variance of f.

C. Analysis of the Dataset

The projection in the Fourier basis requires an estimation of the a_u spectral components. Figure 2 depicts the power traces related to different classes, i.e. the unmasked input values, in the ISW circuit. Figure 3 shows that the convergence rate to accurate values of the coefficients is rapid: after only 1024 power measurements, the coefficients are already estimated with a high confidence. In the sequel, we will therefore present basis decomposition based on estimation from 1024 traces. The spectral coefficients are extracted using the average power values shown in Figure 2, by multiplying them by the 16×16 Walsh-Hadamard matrix.

The $(a_u)_{u \in (\mathbb{F}_2^n)^*}$ parameters allow to characterize the leakage. Typically, it is interesting to study the waveform of the

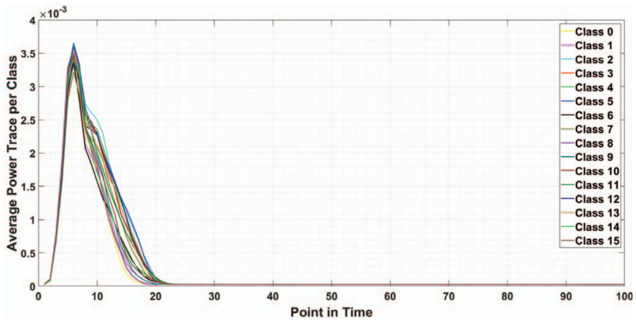

Fig. 2: Average power of ISW classified according to the 16 values of the unmasked plaintext, for 100 samples (2 ns trace, sampled at 50 Gsample/s).

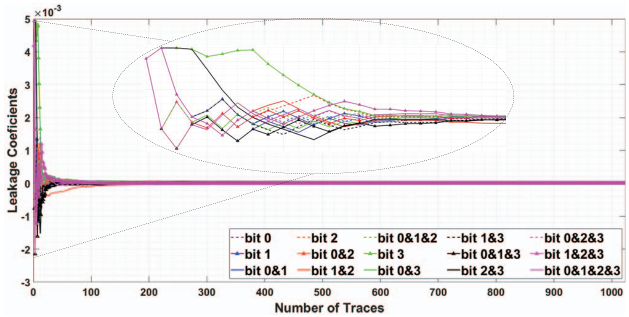

Fig. 3: Leakage coefficients of the ISW extracted using average power with respect to the number of traces.

coefficients. Recall that in absence of first-order leakage, the leakage should be null. Conversely, a high positive or negative value indicates a deviation from the baseline value (zero). The curve which features such deviation(s) reveals the nature of the leakage. Typically, it can be due to a single bit, when $w_H(u) = 1$, or otherwise to multiple bits leaking together. The former case is reasonably attributed to an unfortunate "demasking", while the latter case arises upon complex conditions on the unmasked inputs (connected to "glitch events). The reason is that the typical condition of appearance of a glitch is "when some configuration c is met, then the value of this net a changes values". The leakage value is therefore the logical value $c \wedge a$, where a and c are unmasked values. An interesting such leakage is illustrated in Fig. 4, where a strong leakage occurs as the conjunction between bits 1 and 2 (i.e., the projection of the leakage on vector: $(-1)^{t_1+t_2}$ is large). Interestingly, TI implementation features glitches, which nonetheless **do not** leak information, since those glitches mix only at most $d = 3$ shares of each variable randomly split in $d + 1 = 4$ shares. Still, another leakage can be exploited, namely *Hamming weight/distance parity*. Indeed, power is an additive quantity, hence the leakage of each individual gate (or net) sums up. Thus an overlooked first-order leakage:

Theorem 1. *A random Boolean splitting of any order is leaking the least significant bit (LSB) in the Hamming weight model.*

Proof. Let a Boolean sensitive variable $x \in \mathbb{F}_2$, that is randomly split for a given masking scheme into $x_0, x_1, \ldots, x_d$. Assume the leakage function is the Hamming weight (w_H). Then we get that: $LSB(w_H(x_0, x_1, \ldots, x_d)) = LSB(\sum_{i=0}^{d} x_i) = \bigoplus_{i=0}^{d} x_i = x$. $\square$

Therefore, we pinpoint an intrinsic leakage of masked logic: the parity of the leakage function discloses the value of the unmasked sensitive variable x. Interpreting the origin of leakage is out of the scope of this paper. We rather focus on comparing the amount of leakage of various implementations.

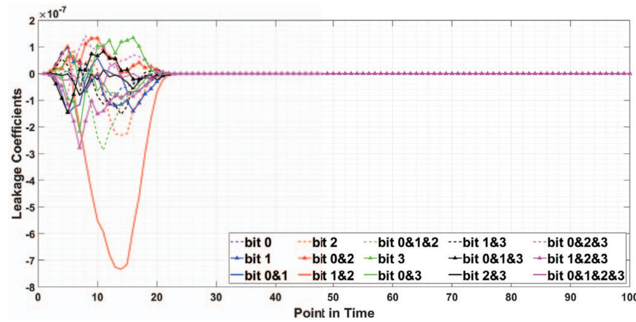

Fig. 4: Leakage coefficients of the ISW extracted with respect to the number of samples per unit of measurement.

IV. PRESENT S-Box Implementations

In this paper, we target the hardware implementations of PRESENT cipher, one of the lightweight cryptographic modules that fit the resource constrained IoT devices. In particular, we investigate the leakage and vulnerability of 2 unprotected implementations of PRESENT and 5 masking-protected counterparts, the details of which can be found below.

PRESENT Cipher: PRESENT is a lightweight symmetric 64-bit block cipher based on the common substitution-permutation network. A bitwise XOR operation, a nonlinear substitution layer (S-Box), and a linear permutation layer are included in every encryption cycle.

Being non-linear and possessing a contrasted confusion coefficient [18], the S-Box module is a highly appealing target for adversaries to leak the secret key by compromising it. Accordingly, in this study, we focus on PRESENT S-Box implementations (unprotected and masked-protected circuitries).

A. Unprotected Implementations

Lookup Table (LUT): LUT-based implementation is the most straightforward approach. It can simply be implemented using a 4-bit lookup table with 16 different addressable locations. We implemented this architecture using combinational logic and use it as the baseline for other protected variants.

Optimized Implementation (OPT): This implementation minimizes the number of non-linear (AND/OR) gates. The final netlist is generated by trying different implementations using Boolean Satisfiability (SAT) solvers [19]. This circuit has a longer critical path compared to the baseline implementation. Note that although based on Table I in both circuits 8 gates are included in the critical path, OPT critical path includes more XOR gates, while LUT has more AND/OR gates; thus OPT has a longer propagation delay.

B. Protected Implementations

Global Lookup Table (GLUT): GLUT masking is a function $\mathbb{F}_2^4 \times \mathbb{F}_2^4 \times \mathbb{F}_2^4 = \mathbb{F}_2^{12} \rightarrow \mathbb{F}_2^4$ satisfying

$$Y = GLUT(A, MI, MO)$$

such that:

$$Y \oplus MO = S\text{-}Box(A \oplus MI)$$

where A and Y are the masked input and output, respectively and MI and MO refer to the input and output masks.

Rotating S-Box Masking (RSM): Under the assumption that the underlying logic does not leak, the RSM is proved to be first-order secure (and even second- and third-order [20]). The RSM implementation aims at minimizing the required randomness in two ways, so, it has the following properties:

1) the masks set is a subset of the full mask set (the masking scheme is referred to as "low-entropy" [21], and
2) the masks used at the output of the S-Box are deduced deterministically from that at the input, namely by using the next one (in a circular manner) within the mask set.

In our implementation, since there are only 16 masks in the 4-bit PRESENT S-Box, we refrain from using a strict subset of the masks. However, we generated output masks based on the available input randomness [22], thus the output mask MO is generated as below:

$$MO = (MI + 1) \bmod 16$$

where MI and MO are integers in $\{0, 1, \dots, 15\}$, computed via $M \in \mathbb{F}_2^4 \mapsto \sum_{i=0}^{3} M_i 2^i \in \{0, 1, \dots, 15\}$ mapping. So, the relationship between RSM and GLUT can be written as:

$$RSM(A, MI) = GLUT(A, MI, (MI + 1) \bmod 16).$$

From an area perspective, RSM has a more compact architecture compared to the GLUT. Note that the genuine version of RSM does not have S-Box input on 8-bit, but solely on 4-bit [23] (wherein randomness comes from S-Box shuffling).

ROM-based RSM (RSM-ROM): A stronger implementation of RSM can be realized using a Read-Only Memory (ROM). In our implementation (so-called RSM-ROM), we target logic designs built only from the instantiation of gates in a Boolean library [24]. Here, initially, the datapath is synchronized for any input configuration which makes input-related deviations of leakage small. Then, the structure is designed with a *one-hot* strategy, i.e., only the required logic is activated, which further contributes to reducing the side-channel footprints.

Gate-Level Masking via Random Sharing (ISW): Proposed by Ishai, Sahai, and Wagner (ISW [25]), this secure implementation starts from the *OPT* netlist, and gradually replaces the gates with their gadgets, in order to deal with the non-linear gates. In this architecture, the gadget for the AND gate requires 1-bit of randomness, i.e., R). Given a random sharing (A_0, A_1) of bit A (where $A = A_0 \oplus A_1$), and a similar sharing for bit B, the AND of A and B denoted as Y is computed as below:

$$\begin{cases} Y_0 &= ((A_1 \wedge B_1) \oplus R) \oplus (A_0 \wedge B_0) \\ Y_1 &= ((A_0 \wedge B_1) \oplus R) \oplus (A_1 \wedge B_0) \end{cases}$$

In the above equations, the order of operations should be followed, thus the implementation must preserve the order of gates in the final netlist. In our implementation, we implemented OR via benefiting from De Morgan's law $\mathrm{OR}(a, b) = \neg\mathrm{AND}(\neg a, \neg b)$. Since combinational gates evaluate their outputs whenever their inputs change, preserving the order can be challenging in ISW due to the race condition. This can lead to a first-order leakage [26].

Threshold Implementation (TI): TI is an algorithmic countermeasure against power SCA, which benefits from multi-party computation and secret sharing [27]. TI, alike ISW, divides input bits into $d+1$ shares. Meanwhile, in TI the underlying logic can be aggressively optimized as it does not need to preserve gate ordering. Moreover, regarding its non-completeness property, in TI any output share only depends on d shares of each input. Thus, glitches cannot lead to secret information disclosure in TI. In our netlist, terms of order 3 (3-input AND gates) are required, hence 4 shares are needed; we managed to synthesize a TI-compliant *fully combinational* netlist of PRESENT S-Box. Notice that we are not aware of a similar netlist being published in the literature (the only paper tackling this problem resorts to an implementation with registers in the middle of the netlist [28]).

Synthesis: Table I compares the investigated S-Box implementations regarding the number of gates (with $2-4$ inputs), equivalent gates (#gates normalized by the number of equivalent 2-input NAND gates), random bits, and propagation delay (in terms of #gates in each path).

TABLE I: Gate-level specification of the targeted S-Box implementations

	LUT	LUT-OPT	GLUT	RSM	RSM-ROM	ISW	TI
# AND	18	2	580	134	0	16	800
# OR	7	2	180	74	0	0	0
# XOR	0	9	0	0	0	34	647
# INV	7	1	12	20	510	7	0
# BUF	0	0	0	0	0	0	1
# NAND	0	0	0	0	16	0	0
# NOR	0	0	0	0	716	0	0
# XNOR	0	0	0	0	0	0	2
Total Gates	32	14	772	228	1242	57	1450
Total Equ. Gates	41	29	1183	373.5	1121	112.5	2423.5
Delay	8	8	15	11	120	17	9
# Random Bits	0	0	8	4	4	4	12

V. Experimental Results and Discussions

A. Experimental Setup

We implemented the add-round-key and S-Box operations in the first round of the PRESENT cipher with 80-bit keys in the transistor level for the 7 types of S-Box using a 45-nm technology extracted from the open-source NANGATE library [29]. We used Synopsys HSpice for the transistor-level simulations, and the HSpice built-in MOSRA Level 3 model [30] to extract NBTI, PBTI, and HCI aging [14] effects. The effect of aging was evaluated for 4 years of device operation in time steps of 2 months. The simulations were conducted for temperature of 85°C, $V_{dd} = 1.2\ V$, and the operating frequency of 500 MHz. This clock period is selected based on the longest critical path among all the implementations.

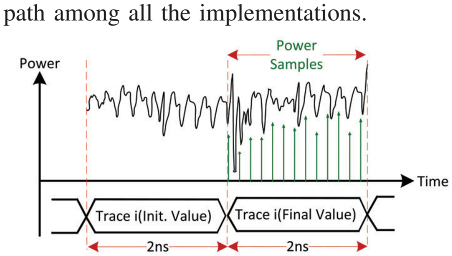

Fig. 5: Protocol for sampling power traces (recall that "initial value" is a random sharing of $(0000)_2$).

The simulated traces contain two parts: the results of key addition and S-Box outputs for each initial n-bit value as well as its following n-bit value. For the attack, we considered only the power samples of the second part, when the cryptographic circuit transitions from initial to final value. Fig. 5 shows the power sampling where 100 samples are captured for each trace after feeding the related final value over the course of 2 ns.

To have a non-biased evaluation, we generated the final values randomly, yet such that we have equal number of traces from each class, i.e., equal number of unmasked inputs (recall Fig. 2). The initial values for masks themselves as well as masked inputs are generated such that the initial values belong to class '0' (e.g., A_initial $\oplus$ MI_initial = 0 in GLUT). This results in a fair comparison among the implementations.

We generated 1024 input traces such that we have 64 traces in each class. We categorize the power traces into 16 classes based on the value of the unmasked inputs in their final stage. We then use the mean trace of each class in our leakage assessments.

B. Experimental Results and Discussion

1) Leakage power and total leakage power in different architectures: The first set of results compares the targeted implementations regarding their resiliency against power SCA by assessing the leakage power in each circuitry in each of the 100 sampling points. In particular, we calculate the leakage power at each point of time as below:

$$Leakage\_Power(T) = \sum_{u=1}^{15} a_u^2(T)$$

where a_u relates to the Walsh-Hadamard coefficients discussed in Proposition 1. Here, T is the $Point\ in\ Time$ in which the power traces are sampled, and u denotes a leakage source. As mentioned in Section III (and shown in Fig. 4), for the 4-bit PRESENT S-Box, we have 15 distinct leakage sources including 4 single-bit (related to one of bit0, bit1, bit2, and bit3) and 11 multiple-bit leakages (related to the pair, tuple, and quadrable combinations of those 4 bits).

Figure 6 shows the leakage power for all S-Box variants for the first 20 sampled points. These curves can be exploited to identify the "Points of Interest", i.e. when the leakage shows up. Using such information can increase the probability of the attack success by targeting the traces on those specific points of time. As Fig. 6 depicts, the average leakage power (computed using the above equation) is more prominent in the unprotected circuits followed by TI. Interestingly TI leaks more than other masked architectures although supposed to be more secure. This is due to its implementation complexity regarding its larger size compared to the other implementations. As shown in Table I, TI includes 21× equivalent gates compared to ISW and 2.1× more gates then RSM-ROM. The other masked implementations behave similarly in terms of average leakage power shown in Fig. 6.

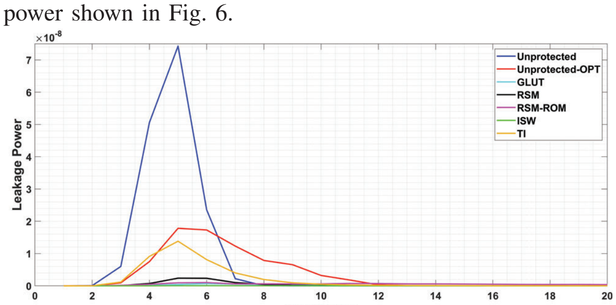

Fig. 6: Leakage power in different unprotected and masked-protected implementations of PRESENT S-Box in different points in time.

To analyze the resiliency of the implemented masked architectures in more detail, we assessed the total leakage power related to the 100 sampling points altogether:

$$Total\_Leakage\_Power = \sum_{T=1}^{100} \sum_{u=1}^{15} a_u^2(T)$$

The extracted Total_Leakage_Power values are shown in Fig. 7 for both, the fresh (not aged) and aged devices. In this figure, the total leakage (in each age of device) is the summation of the related sub-bars depicted as unfilled and solidly filled altogether. As expected, the two unprotected implementations are the most vulnerable ones among all, owing to their high leakage power. TI is shown to be the least secure masked implementation in this figure with the leakage power at around 45% of the unprotected optimized circuit (i.e., optimized LUT).

Based on the results presented in Fig. 7, RSM-ROM is less secure than ISW, RSM, and GLUT implementations. This is due to its long propagation time compared to other implementations. Such a long delay provides the attacker with more target points, thus facilitates the attack. ISW is the most resilient structure among the investigated implementations.

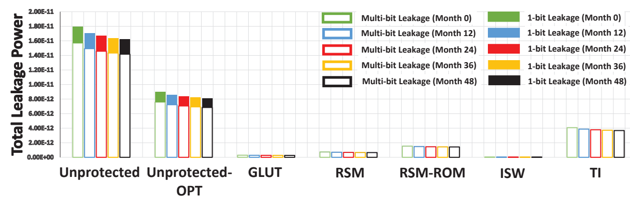

Fig. 7: Total leakage power in different fresh (age=0) and aged unprotected and masked-protected implementations of PRESENT S-Box over time. The glitch-related leakage is shown using unfilled sub-bars while single-bit leakages are depicted in solid sub-bars.

2) Effect of Aging on the Power Leakage: As discussed in Section II, when the device is aged the threshold voltage of the transistors and in turn the current passing through them is changed, thus the leakage that an adversary can exploit to launch an SCA attack is impacted. Accordingly, we depict quantitatively how device aging affects the security of the investigated masking (and unprotected) structures.

Figure 8 shows the Leakage Power for ISW, as the most secure candidate among all considered masking architectures based on the results shown in Fig. 6, when new (depicted as 0 month) and the ISW circuitry that has been used for 1 to 4 years. As expected, the leakage power decreases over time, and with a faster rate at the beginning (i.e., the degradation of leakage between new and the 1-year old device is more than the degradation rate between the 1- and 2-year old devices, etc).

Fig. 7 compares the total leakage power of the implemented architectures over the course of 4 years. The first observation is that, as expected, the total leakage power for each implementation decreases over time. The takeaway point from this observation is that the masking schemes (in contrast to dual-rail logic hiding implementations, e.g., Sense Amplifier Based Logic (SABL) [15]) do not become more vulnerable during the lifetime, thus if the protection is secure when the device is new, it will continue to behave similarly during the course of usage.

Another observation that can be made from Fig. 7 is that although the total leakage of each implementation degrades

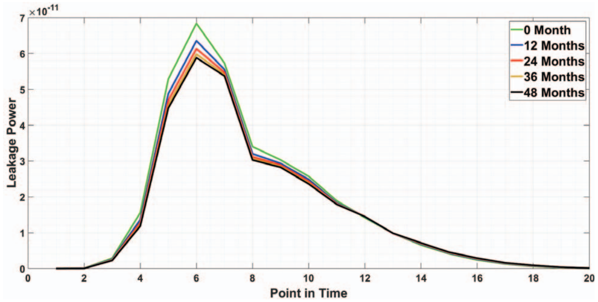

Fig. 8: Leakage power of the ISW implementation over 4 years of usage.

over time, the most secure implementations (ISW and then GLUT) still remain as so after aging. The same applies to the most vulnerable ones (unprotected ones and then TI). In other words, the order of preserving security for the considered implementations stays unchanged even after aging.

The next interesting observation is regarding the magnitude of single-bit leakage versus multi-bit leakage in each circuitry. As shown for all protected devices when aged, similar to when being new, the leakage is mainly due to the interaction of bits, i.e., they do not have a considerable single-bit leakage. Indeed the rate of single-bit leakage in protected circuits on average is less than 1% (i.e., $\approx$0.50%) of the whole leakage while for the unprotected circuits on average this rate is $\approx$ 14.044% when devices are new. This confirms that the masking implementations were secure (as expected) regarding single-bit leakage, which is the metric that they are mainly assessed (bit probing model [25]). However, this urges the community to take the bit interactions into account in the security sign-off as such interactions may result in leaking secure data. As Fig. 7 depicts, on average the percentage of single-bit leakage over the total leakage for the masked implementations is 0.511%, 0.511%, 0.512%, and 0.507% for the 1 to 4 year old devices, respectively and 14.051%, 14.035%, 14.017%, and 13.998% for the unprotected devices over the same course of usage. This shows that the percentage of single-bit leakage over the total leakage is almost constant over time in all implementations.

3) Discussion: Based on Tab. I, substantial variation of delay, area, and entropy is observed in the considered circuits. Also regarding the security metric, we shall consider side-channel leakage. The proposed spectral analysis allows to distinguish between two phenomena to account for the 1st-order leakage (despite the implementations are not supposed to have 1st-order leakage), namely single- and multiple-bit leakage.

It is clear, from Fig. 7, that only unprotected styles leak single bits (indicated by the "solidly filled" part of the bars in Figure 7). This justifies the requirement for protected styles. As a matter of fact, all protected implementations are secure against mono-bit leakage attacks. Still, we get as an outcome that:

- TI: by design, features no glitch. But the leakage is amplified by the large netlist.
- GLUT, RSM, RSM-ROM: they all improve security but less than ISW does.
- ISW: is compact, features no leakage owing from the glitches (by design – proved in the seminal paper). The only leakage is resulting from early evaluation [26]. The ISW is so good from a security standpoint because the PRESENT S-Box representation can be very optimized in terms of AND/OR gates low count.

VI. Conclusion and Future Directions

Masking schemes have received the utmost attention in protecting the cryptographic hardware implementations against side-channel analysis attacks. In this paper, we presented a Walsh-Hadamard based leakage analysis, and showed that although masking schemes do not leak in theory, an adversary may benefit from the interactions among the bits and leak the secret data, thanks to the existing glitches and/or structural leakages (as that of Theorem 1). We showed quantitatively that among the considered masking schemes, ISW remain the most secure. We will investigate our findings in real silicon as the continuation of this study.

Acknowledgement

The work described in this paper has been supported in part by the National Science Foundation CAREER Award (NSF CNS-1943224).

References

[1] "ISO/IEC 18033-3:2010," Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers.
[2] ISO/IEC 29192-2:2012, "Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers," p. 41, Publication date: 2012-01, Edition: 1. https://www.iso.org/standard/56552.html.
[3] H. Liu, G. Qian, S. Goto, and Y. Tsunoo, "Correlation Power Analysis Based on Switching Glitch Model," in *WISA*, 2010, pp. 191–205.
[4] H. Liu, G. Qian, Y. Tsunoo, and S. Goto, "The Switching Glitch Power Leakage Model," *JSW*, vol. 6, no. 9, pp. 1787–1794, 2011.
[5] W. Fischer and B. M. Gammel, "Masking at Gate Level in the Presence of Glitches," in *CHES*, 2005, pp. 187–200.
[6] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold Implementations Against Side-Channel Attacks and Glitches," in *ICICS*, 2006, pp. 529–545.
[7] S. Nikova, V. Rijmen, and M. Schläffer, "Secure Hardware Implementation of Non-linear Functions in the Presence of Glitches," in *ICISC*, 2008, pp. 218–234.
[8] ——, "Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches," *J. Cryptology*, vol. 24, no. 2, pp. 292–321, 2011.
[9] [Online]. Available: https://csrc.nist.gov/Projects/masked-circuits
[10] A. Wild et al, "GliFreD: Glitch-Free Duplication Towards Power-Equalized Circuits on FPGAs," *IEEE Trans. Computers*, vol. 67, no. 3, pp. 375–387, 2018.
[11] M. Taouil, A. Aljuffri, and S. Hamdioui, "Power side channel attacks: Where are we standing?" in *Design & Tech. of Integ. Systems in Nanoscale Era*, 2021, pp. 1–6.
[12] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *CHES*. Springer, 2004, pp. 16–29.
[13] F. Oboril and M. B. Tahoori, "Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level," in *IEEE DSN*, 2012, pp. 1–12.
[14] A. Gomez et al., "An early prediction methodology for aging sensor insertion to assure safe circuit operation due to nbti aging," in *VTS*, 2015, pp. 1–6.
[15] M. T. H. Anik et al., "On the impact of aging on power analysis attacks targeting power-equalized cryptographic circuits," in *IEEE ASP-DAC*, 2021, pp. 414–420.
[16] T. De Cnudde et al., "Does Coupling Affect the Security of Masked Implementations?" in *Constructive Side-Channel Analysis and Secure Design*, 2017, pp. 1–18.
[17] S. Guilley et al., "Stochastic side-channel leakage analysis via orthonormal decomposition," in *Information Technology and Communications*, 2017, pp. 12–27.
[18] Y. Fei, A. A. Ding, J. Lao, and L. Zhang, "A statistics-based success rate model for DPA and CPA," *J. Cryptographic Engineering*, vol. 5, no. 4, pp. 227–243, 2015.
[19] NIST, Computer Security Resource Center, "Circuit Complexity Project," https://csrc.nist.gov/Projects/circuit-complexity. Created Dec. 2016.
[20] C. Carlet and S. Guilley, "Side-channel indistinguishability," in *Hardware and Architectural Support for Security and Privacy*, 2013, pp. 1–8.
[21] M. Nassar et al., "Formal analysis of the entropy/security trade-off in first-order masking countermeasures against side-channel attacks," in *INDOCRYPT*, 2011.
[22] ——, "RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs," in *IEEE DATE*, 2012, pp. 1173–1178.
[23] S. Guilley, A. Heuser, and O. Rioul, "Codes for Side-Channel Attacks and Protections," in *Codes, Cryptology and Information Security (C2SI)*, 2017, pp. 35–55.
[24] M. Giaconia et al., "Area and power efficient synthesis of dpa-resistant cryptographic s-boxes," in *Int'l Conf. on VLSI Design*, 2007, pp. 731–737.
[25] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Annual Int'l Cryptology Conf.*, 2003, pp. 463–481.
[26] D. B. Roy et al, "Cc meets fips: A hybrid test methodology for first order side channel analysis," *IEEE Trans. on Computers*, vol. 68, no. 3, pp. 347–361, 2018.
[27] J. Bahrami et al., "Lightweight implementation of the lowmc block cipher protected against side-channel attacks," in *ASHES Workshop*, 2020, pp. 45–56.
[28] M. Farmani, "Threshold Implementations of the Present Cipher," Ph.D. dissertation, July 2017, https://web.wpi.edu/Pubs/ETD/Available/etd-090617-125035/unrestricted/MFarmani.pdf.
[29] "Nangate 45nm open cell library," "http://www.nangate.com".
[30] Synopsys, "HSPICE User Guide: Basic Simulation and Analysis," 2016.