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 $x_0 ⊕ x_1 ⊕ ... ⊕ x_d = x$. 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...
higher-order attacks \((d+1\text{-order})\). 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.](image)

**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 appear as a resurgence [16]. In the scope of this paper, we

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| **Lemma 1.** | The leakage function \(f : t \in \mathbb{F}_2^n \rightarrow f(t) \in \mathbb{R}\) can be decomposed in the Fourier basis as: \(f(t) = \sum_{a \in \mathbb{F}_2^n} a \cdot f(t)(-1)^{a \cdot t}\), where \(a \cdot t\) denotes the canonical scalar product, that is \(a \cdot t = \sum_{i=1}^{n} a_i \cdot t_i\). This basis is orthonormal, in that \(\langle \psi_i | \psi_j \rangle = 0\) if \(i \neq j\) and 1 otherwise. There is a simple interpretation of the Fourier basis: vector \(\psi_u\) represents the interactions between bits of the input corresponding to the 1 inputs in \(u\) (the so-called support of \(u\)).

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| **C. Analysis of the Dataset** | The projection in the Fourier basis requires an estimation of the \(a_u\) spectral coefficients. 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 \(16 \times 16\) Walsh-Hadamard matrix.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| **Theoretical Tool for Spectral Analysis** | Gliitches 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 \(\psi_u\), 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, \(\psi_u\) function is \(t \in \mathbb{F}_2^n \rightarrow \psi_u(t) = \frac{1}{\sqrt{2^n}} (-1)^{u \cdot t}\), where the operation \(u \cdot t\) denotes the canonical scalar product, that is \(u \cdot t = \sum_{i=1}^{n} u_i \cdot t_i\). This basis is orthonormal, in that \(\langle \psi_i | \psi_j \rangle = 0\) if \(i \neq j\) and 1 otherwise. There is a simple interpretation of the Fourier basis: vector \(\psi_u\) represents the interactions between bits of the input corresponding to the 1 inputs in \(u\) (the so-called support of \(u\)).

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| **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 \(t (t \in \mathbb{F}_2^n)\). 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 transition to a value \(x\) from initial value \((0000)_2\) is \(x \oplus (0000)\), i.e., \(x\) itself.

### 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:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| **Fig. 2:** Walsh-Hadamard transform of \(f\). 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 \(16 \times 16\) Walsh-Hadamard matrix.
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:

\[
\text{LSB}(w_H(x_0, x_1, \ldots, x_d)) = \text{LSB}(\sum_{i=0}^{d} x_i) = \bigoplus_{i=0}^{d} x_i = x.
\]

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.

### 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.

### B. Protected Implementations

**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.

**Global Lookup Table (GLUT)**: GLUT masking is a function \( \mathbb{F}_2^4 \times \mathbb{F}_2^4 \times \mathbb{F}_2^2 = \mathbb{F}_2^1 \rightarrow \mathbb{F}_2^2 \) satisfying

\[
Y = \text{GLUT}(A, MI, MO)
\]

such that:

\[
Y \oplus MO = S-Box(A \oplus MI)
\]
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:

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

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

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

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 the underlying logic does not leak, the RSM is proved to 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 $2^d$ 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).

**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]).

**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 $85^\circ C$, $V_{dd} = 1.2$ 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 $\{0000\}_2$).](image)

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 ⊕ 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:

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

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 than RSM-ROM. The other masked implementations behave similarly in terms of average leakage power shown in Fig. 6.

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:

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

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. 6: Leakage power in different unprotected and masked-protected implementations of PRESENT S-Box in different points in time.](image)

![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.](image)
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., ≈0.50%) of the whole leakage while for the unprotected circuits on average this rate is ≈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.

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 determine the amount of leakage in different frequency bands. 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.

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 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