# Security Order of Gate-Level Masking Schemes

Sofiane Takarabt*, Javad Bahrami**, Mohammad Ebrahimabadi**, Sylvain Guilley*, and Naghmeh Karimi**
* *Secure-IC S.A.S., Think Ahead Business Line, 75 014 Paris, France*
** *University of Maryland Baltimore County, Baltimore, US 21250*
* {*sofiane.takarabt, sylvain.guilley*}*@secure-ic.com*
** {*jbahram1, e127, nkarimi*}*@umbc.edu*

2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST) | 979-8-3503-0062-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/HOST55118.2023.10133208

***Abstract*—Masking schemes have been introduced to thwart side-channel attacks. In software applications, attackers can measure leakage at several points in time and combine them to defeat the masking. In hardware gate-level masking, all shares of a masked variable are manipulated at the same time in a nanoscale circuit. In this article, we focus on setups where the attacker uses one mesoscopic probe, which measures an aggregated leakage of all shares. We consider masking schemes where each bit is randomly split (by XOR) into so-called shares (two or more). We analyze two interesting case studies about the interrelationship of attack order vs. the number of shares. First of all, we show that when the unique probe is measuring the sum of each share's individual leakage (so-called Hamming weight model), one measurement can reveal the sensitive unshared value, provided the attacker is able to determine the leakage's least significant bit. Second, we analyze a hardware masking belonging to threshold schemes. Such schemes require fulfilling a so-called incompleteness property, whereby some input shares must be absent from output shares. We analyze a first-order incomplete scheme, i.e., where the number of missing input shares is equal to one. In schemes such as threshold implementation, this requires the number of shares to be strictly more than two. Hence the natural question is whether such a scheme would resist high-order attacks of order also strictly more than two? We answer by the negative, and show that the lowest attack order is two: the security of such a masking scheme is governed by the order of incompleteness and not by the number of shares. We verify our findings using four different sets of experiments including theoretical analysis, digital simulation, HSpice simulation and also real-silicon (FPGA emulation).**

***Index Terms*—Gate-level masking, number of shares, threshold schemes/threshold implementation (TI), incompleteness order, high-order monovariate attacks, statistical moments of a distribution, Hamming weight least significant bit leakage, 2nd-order leakage of threshold implementation style.**

## I. Introduction

Hardware chips which manipulate sensitive data, say cryptographic keys, can be attacked by differential power analysis [19] and other such vertical analyses [21, §6]. This necessitates protecting such chips against Side-Channel Analysis (SCA) attacks. To protect against such attacks, hiding [21, §7] and masking [21, §9] schemes have been proposed in the literature.

Hiding schemes try to eliminate any source of leakage, typically by balancing the observable side-channel dissipation irrespective of the input of the circuit, i.e., via equalizing the power consumption for all signal transitions. It is very complex to implement properly in practice, as any slight unbalance, including process-variation induced unbalance contributes to an exploitable leakage. On the other hand, masking consists of *dynamically* decorrelating the leakage from manipulated values via randomizing intermediate computations through secret sharing. To work properly, such technique requires the randomization of all intermediate variables. Masking has the decisive advantage that it works irrespective of the physical implementation since the protection is ensured at the logical level. For this reason, masking is often a preferred countermeasure amongst the wide array of existing side-channel protection technologies.

The proper implementation of masking at hardware level represents a relative challenge since high-level description can be abstract and succinct, whereby low-level representation (called a netlist) is particularly verbose. Indeed, netlists can consist of thousands to millions of logic gates. Thereby, ensuring the required masking of each and every gate requires a sound methodology.

A fundamental specificity of hardware logic gates is that they evaluate as soon as any input changes. In other words, hardware gates are non-synchronizing, or are simply *combinational*. The consequence is that hardware gates might well be masked at some time but be demasked transiently at some other time. Actually, some race conditions between signals can result in values changing during one evaluation. Such event is characterized as being a *glitch*.

Glitches are very hard to model exhaustively owing to the large combinatorics of possible races (exponential in the number of gates). Notice that in practice, little effort has been made by the Electronic Design Automation (EDA) industry to speed up the analysis of glitches. Indeed, as a matter of fact, glitches are dependent on the so-called PVTA (Process, Voltage, Temperature, and Aging) corners. Therefore, depending on the static design, on the aging condition of the chip, and on the environmental conditions in which it is operated, glitches do differ. For this reason, some masking schemes have been proposed which can, by design, even withstand leakage through glitches. For example, the Threshold Implementation (TI) of gate-level masking countermeasure [27] requires extra randomization to build glitch-resistant netlists constructively from the bottom up. Implementations of TI on some challenging designs (e.g., equations of high algebraic degree) require a sharing in large dimensions. For instance, the papers [5], [15], and [30] respectively study the implementation of TI on cryptographic algorithms AES, GIFT, and PRESENT.

TI netlists do feature glitches. Indeed, the TI netlists are made up of combinational gates, where any input and even any sequence of inputs are basically all permitted. However,

the *incompleteness* property of the TI netlists (refer to [25, Property 1 in §3.2]) ensures that any gate is driven by strictly less than number of shares in the corresponding TI implementation, thereby no combination of combinational signals can lead to a constructive demasking of the sensitive values, even during a sporadic glitch. Therefore, it is said that (properly designed) TI is resistant to leakage even in the presence of glitches. Though seminal TI has first-order incompleteness only, subsequent works, such as [6], have introduced high-order incompleteness as well.

**Contributions:** Our contribution consists of the security evaluation of hardware masking schemes. We disclose a property of hardware masking schemes that allows revealing unmasked data, regardless of their being glitch-resistant or not. This property is remarkable as it is applicable irrespective of the number of shares. We illustrate our results on the PRESENT 4-bit S-box.

In particular, we explain that TI with first-order incompleteness, irrespective of the number of shares, is no more than 1st-order secure. We have verified our findings using 4 different sets of experiments including theoretical analysis, digital simulation, HSpice simulation, and also real-silicon (FPGA emulation).

**Outline:** The rest of the paper is structured as follows. State-of-the-art side-channel attacks are recalled in Sec. II. Accordingly, a review of hardware masking schemes is provided in Sec. III. Sec. IV discusses the problems we are tackling in this paper, followed by the rationale of our novel attacks in Sec. V. Experimental results demonstrating the attacks' success on both synthetic traces as well as FPGA results are presented in Sec. VI. This section shows that simulated attacks on synthetic traces are not too idealistic, as they can be reproduced in practice. Also, our results explain the observation made by Moradi et al. in [24], that first-order incomplete TI cannot be attacked by a monovariate first-order statistical analysis but fails when facing a monovariate second-order statistical analysis. Eventually, Sec. VII concludes the paper.

## II. Side-Channel Attacks

Side-channel attacks represent one of the most powerful categories of attacks launched by adversaries to obtain secret information of a device, e.g., a cryptographic key. These attacks retrieve the secret key by analyzing the physical leakage emitted during the operation of a device (e.g., power consumption [4], [18], or electromagnetic radiation [12]), as this leakage can be statistically dependent on the secret key.

### A. Practice of side-channel attacks

In this paper, we consider attackers targeting hardware implementations. Hardware masking schemes usually process data (protected by *masking*, also called *random sharing*) in one given clock period. Therefore, the attacker can only capture one measurement (for a given plaintext) to try and attempt to defeat the masking scheme. Such attacks are thus termed *monovariate*. Notice that this situation differs from the attack of software schemes where each share is handled at a different time, hence can be captured individually (leading to *bivariate* attack when two leakages are captured for a given plaintext, etc.). So we consider single probe attacks, and subsequent (potentially high-order) analyses will need to cope with such specificity. This means that the attacker can collect (monovariate) leakage from several plaintexts, and analyse the leakage distribution: indeed, the masking countermeasure distributes the leakage as a "probability density function" (PDF). This PDF features different statistical moments (the lowest ones being mean, variance, skewness, and kurtosis), that can show a dependency in the (unmasked) sensitive variable. An analysis exploiting a high-order moment is called a high-order analysis. We detail now in Sec. II-B and II-C two analyses that are capable of exploiting high-order statistical moments of monovariate leakage traces in the sensitive variable.

### B. Background on Template Attacks

Profiled side-channel attacks are the most powerful type of attacks in which an attacker is able to characterize the leakage of a similar device and use the extracted information to break the targeted device. Template attacks are one of the most commonly used profiling attacks, and are the most powerful attacks from an information-theoretic point of view [10]. These attacks are launched in two phases: *training* and *matching*. In the training phase, the attacker has a full control on another copy of the protected device. He/She records a large number of traces of the cloned device, corresponding to random values of inputs (plaintexts and keys). These traces are utilized to build a template $y_k$ from the device, using key $k$. Then, in the matching phase, the recorded traces are classified according to the value of the hypothetical key $\hat{k}$, and the most likely template pinpoints the key value of the device under attack [10]. Let us represent the measured traces as a matrix $X$ of $D \times Q$ real numbers ($Q$ traces of $D = 300$ samples as an example). The value of $D$ is 1 for synthetic traces, but can be larger for real traces, as oscilloscopes capture entire waveforms. We then still term the attack as monovariate, provided that a single operation (e.g., a single clock period) is covered by the waveform. Assume we format the learned model $Y_k$ as a $D \times Q$ matrix. Then the template attack guesses a key by this computation [9, Theorem 1]:

$$\hat{k} \;=\; \underset{k\in\{0,1\}^4}{\operatorname{argmin}} \; \mathsf{tr}\left((X - Y_k)^{\mathsf{T}} \Sigma^{-1} (X - Y_k)\right) \qquad (1)$$

where $\Sigma$ is the $D \times D$ noise covariance matrix, $\mathsf{tr}$ is the trace operator, and $\operatorname{argmin}$ the operator that selects the value of $k$ (4-bit value) that results in the minimum value of its following function. We use interchangeably $\mathbb{F}_2^4$, $\{0,1\}^4$, and the integer conversion belonging to $\{0, 1, \ldots, 15\}$. Note that in general $k$ denotes to the key value (precisely: subkey entering a given S-box); thus for the PRESENT cipher we target in this study $k$ is a 4-bit value and therefore lives with the range of $[0, 15]$.

### C. Moments-Correlating Differential Power Attack (MC-DPA)

Template attacks allow distinguishing key hypotheses provided the leakage distribution differs from key to key. Although being optimal from the information-theoretic point of view, template attacks have a shortcoming: their problem is their lack of *explainability*.

Recall that distributions can be characterized by their moments. In the presence of noise, low order moments are most discriminating than high order moments. The rationale behind MC-DPA [24] is to perform attacks through their *smallest* moment, which depends on the sensitive variable.

An MC-DPA is applied at a given order: by definition, an MC-DPA of order 1 (resp. 2, 3 and 4) analyses the dependency of leakage *mean* (resp. *variance*, *skewness*, and *kurtosis*) in the sensitive variable. One difference between a template attack and an MC-DPA is that the template attack succeeds if any moment of the leakage differs, whereas an MC-DPA succeeds if the moment of the leakage differs *for the targeted order*. In this paper, we leverage MC-DPA not specifically to perform attacks *per se* (template attacks work in this respect), but to to test the exploitability of leakage at orders 1, 2, 3 and 4.

## III. Masking schemes suitable for hardware

### A. Gate-level masking schemes, from the sequential side

Hardware masking has received the lion's share of attention by the embedded security research community with the notion of *perfect masking*, coined by Blömer et al. in [7]. It gave rise to Boolean Masking (BM), which is easy to generalize from a first-order sharing to a multiple order sharing, say order $d$. Basically, one bit $x$ is randomly split into $d$ bits $x_i$, for $1 \leq i \leq d$, such that

$$x = \bigoplus_{i=1}^{d} x_i. \tag{2}$$

One drawback of BM is that, when manipulating vectorial words (say nibbles in PRESENT block cipher), each component is masked on its own. That is, each output bit is masked independently of the other bits pertaining to the same nibble. Thus, there is unfortunately no synergy between the masks of each bit index from the sensitive bit vector. To alleviate such shortcoming, the *security order amplification* concept has emerged. It consists in mixing different bits from different components in order to achieve a higher protection degree. Such a scheme is typically referred to as IPM, for Inner Product Masking [17]. However, both BM and IPM are vulnerable to glitches.

### B. Gate-level masking schemes, from the combinational side

Some masking schemes have been specialized for hardware implementation. What follows presents a few such state-of-the-art masking schemes. Interestingly there can be some leakage in the combinational logic even in the presence of the following masking schemes.

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

$$y_1 = GLUT(x_1, \ldots, x_d, y_2, \ldots, y_d)$$

such that:

$$y_1 \oplus \ldots \oplus y_d = S\text{-}box(x_1 \oplus \ldots \oplus x_d)\,.$$

The $GLUT$ function can be tabulated in a Read-Only Memory (ROM) or synthesized into a netlist of logic gates. However, owing to the exponential combinational complexity in $d$ of $GLUT$, usual implementations limit the value of $d$ to 2.

Clearly, GLUT can only work securely provided the ROM or the netlist does not locally unmask or weaken the $d$-th order scheme. This is the reason why ISW has been introduced.

**Gate-Level Masking via Random Sharing (ISW):** Proposed by Ishai, Sahai, and Wagner (ISW) [16], this secure implementation starts from an unprotected netlist, and gradually replaces the gates with their so-called gadgets, in order to deal with the non-linear gates. In this architecture, the gadget for the AND gate requires additional randomness. Let us give an example for $d = 2$. Given a random sharing $(a_1, a_2)$ of bit $a$ (where $a = a_1 \oplus a_2$), and a similar sharing for bit $b$, the AND of $a$ and $b$ denoted as $y$ is computed as:

$$\begin{cases} y_1 &= ((a_2 \wedge b_2) \oplus r) \oplus (a_1 \wedge b_1)\,, \\ y_2 &= ((a_1 \wedge b_2) \oplus r) \oplus (a_2 \wedge b_1)\,, \end{cases}$$

where $r$ is a random refresh. In the above equations, the order of operations (indicated by the parentheses) should be followed; thus the implementation must preserve the order of gates in the final netlist [3].

**Threshold Schemes:** Those are algorithmic countermeasures against side-channel analyses which benefit from multi-party computation and secret sharing [2]. Threshold Implementation (TI [26]) and Domain-Oriented Masking (DOM [13]) are two well-studied examples. From now on, we focus on TI, because it can be implemented fully in combinational logic, whereas DOM incurs some "pipelining" with register barriers. To preserve security, a TI implementation holds the following properties: 1) *Incompleteness*: Each output share is independent of at least one share in any of the input variables, 2) *Correctness*: The sum of the output shares gives the desired output, 3) *Uniformity*: The output distribution preserves the input distribution.

TI, alike GLUT or ISW, divides input bits into $d$ shares. Meanwhile, in TI, as opposed to ISW, the underlying logic can be aggressively optimized as it does not need to preserve gate order. Moreover, regarding its incompleteness property, in TI any output share only depends on less than $d$ shares of each input. Thus, glitches cannot lead to secret information disclosure in TI. The notion of incompleteness can be generalized: by definition [27, §4.3], if $N$ input shares are missing from the expression of any output share, the TI scheme is $N$th-order incomplete. In the PRESENT netlist, terms of algebraic order 3 (3-input AND gates) are required, hence 4 shares are needed; we managed to synthesize such 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 [11]).

## IV. Problem statement

The two problems we tackle in this paper are as below:

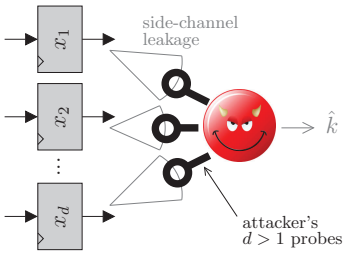



Fig. 1: Theoretical setup for multivariate side-channel analysis. Every probe introduces independent measurement noise. This is **not** the scenario we explore in this paper.

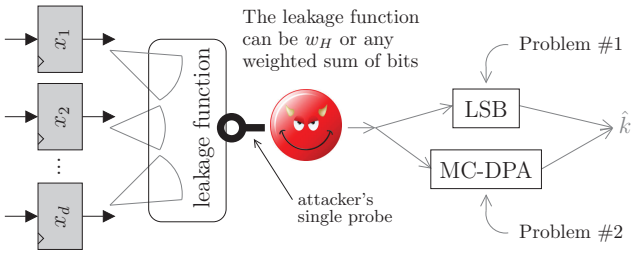



Fig. 2: Actual setup for multivariate side-channel analysis where aggregation is carried out by the leakage function. The measurement noise is only affecting the unique probe.

### *A. Problem #1: Can a first-order attack be devised on a gate-level masking scheme with $d$ shares?*

The classical leakage model considers a theoretical attacker who needs to probe all the $d$ shares $(x_i)_{1\leq i\leq d}$ of a variable to rebuild the sensitive value $x = \bigoplus_{i=1}^{d} x_i$. This is illustrated in Fig. 1. Note that such an attack is severely impeded as each probe brings its own amount of noise.

In practice, all shares are combined in an aggregated leakage function, as probes used in laboratories are not at the scale of registers holding the shares $(x_i)_{1\leq i\leq d}$. Typically, the attacker collects the Hamming weight of the leakage, i.e., their arithmetic sum. Probes are called *macroscopic* if they measure the leakage of the full circuit (e.g., a power probe), and *mesoscopic* if they perform a local measurement, which is nonetheless larger than the registers containing the $x_i$ values (e.g., an electromagnetic probe). Shall the different shares not be manipulated strictly at the same point in time, the capacitive property of the probe makes sure that leakage from different clock cycles to get mixed. See for instance the paper [23]. Thus, the attacker needs to measure only at one place to collect an already aggregated leakage function of all the shares in one go! This attack scenario is depicted in Fig. 2. It is the only one we pursue in the rest of this paper. The attacker simply needs to probe once (monovariate attack), and the noise is as well only applied on this one measurement. Such side-channel scenario can be considered from two angles. On the one hand, the leakage function (say the Hamming weight, $w_H$) clearly brings less information to the attacker (owing to the *data processing inequality*): from this point of view Hamming weight model is detrimental to the attacker compared to the multi-bit probing attack of Fig. 1. On the other hand, the Hamming weight leakage model already combines all the shares, which facilitates the subsequent statistical analysis of the leakage.

Considering the above discussion, a natural question can be whether *a gate-level masking scheme with $d$ shares can be compromised by a first-order analysis?* In the sequel, we will answer this question by the positive. As we shall see (in Sec. V-A), the analysis consists in extraction the least significant bit of the leakage prior to performing a simple (1st-order) correlation analysis. This analysis works on synthetic traces, but we will illustrate that its application to real-world traces is challenging. We'll therefore resort to more general template attacks.

### *B. Problem #2: How to characterize the security order of a gate-level masking scheme, such as threshold implementation?*

Threshold implementation featuring first-order incompleteness requires that each output share bit be shared in strictly more than two shares, though. The natural question is therefore to know whether such schemes are vulnerable to an analysis of order equal to the number of shares, or merely to the incompleteness order. We shall show (in Sec. V-B) that the resistance is only equal to the incompleteness order.

## V. Proposed Methodologies to Address the Two Problems

### *A. Attack methodology for Problem #1*

Let us assume a Hamming weight ($w_H$) leakage model. It can be written with the arithmetic summation:

$$w_H(x_d, \ldots, x_1) = \sum_{i=1}^{d} x_i \in \mathbb{N}.$$

(Namely, $\sum$ operates in the integers, whereas $\bigoplus$ operates in the bits.) Let us denote by LSB the Least Significant Bit of an integer $\ell$, that is $LSB(\ell) = \ell \bmod 2$. In particular, we have:

**Proposition 1** (Leakage in LSB)**.**

$$LSB\big(w_H(x_d, \ldots, x_1)\big) = \bigoplus_{i=1}^{d} x_i \in \mathbb{F}_2.$$

This proposition clearly states that **the LSB of the Hamming weight leakage of $d$ shares $(x_i)_{1\leq i\leq d}$ is exactly the (unmasked) sensitive data $x$.** Indeed, the LSB leakage in Prop. 1 directly inverses the masking scheme, as per Eqn. 2. That is, if the attacker is able to measure the leakage LSB, then he reads out the unmasked value, (incidentally) irrespective of the masking order $d > 1$.

Thus, the first problem is actually easily solved in practice. The aggregation of shares being already a completed step (by the device itself), it suffices for the attacker to extract the LSB. In a practical waveform, Fig. 3 shows a typical side-channel leakage, for an 8 bit sharing. It is apparent in that figure that the magnitude of leakage (voltage signal) is directly related to the number of bit-values in the shared sensitive value [22]. Extracting the LSB of the model requires framing the leakage from 0 to $d$ bits in the Hamming weight model. Figure 3 shows the minimum and maximum values; those can be used to

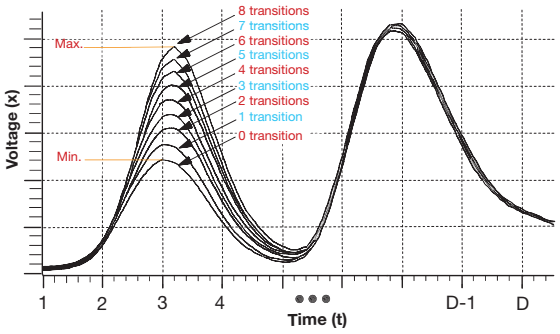



Fig. 3: Number of bit transitions versus power consumption [22]. The Min and Max values are represented. Here, for the sake of simplicity we show the case for $d = 8$.

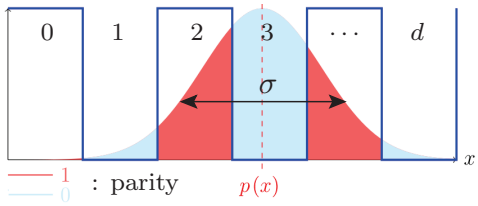



Fig. 4: Illustration of the probability to guess the unmasked value (= LSB of the leakage) as a function of the (normal) noise of variance $\sigma^2$. The red and blue colors illustrate the even and odd parities of the leakage $x = \sum_{i=1}^{d} x_i$ (thus $x \in \{0, 1, \ldots, d\}$).

calculate bin_width (voltage difference when one bit changes) and offset (voltage when all $x_i = 0$). Indeed, as depicted, the maximum voltage (power consumption) occurs for the case where all 8-bit are set. Thus:

- the offset value is exactly equal to Min;
- the bin_width value is equal to $(\mathsf{Max} - \mathsf{Min})/d$; the attacker shall know the number of shares $d$.

The method to extract the LSB (see the LSB box in the dataflow diagram of the analysis of Fig. 2) is thus simply to take the leakage value, subtract offset, divide by bin_width, and apply $\bmod 2$ operation (= parity extraction). Subsequently, a simple first-order analysis (e.g., Pearson correlation analysis [8]) can be carried out between this extracted LSB and the predicted one under key hypothesis $\hat{k}$.

In practice, traces are noisy. Namely, the $q$th ($1 \leq q \leq Q$) side-channel measurement $x_q$ consists in the Hamming weight leakage plus some noise, modeled as a normal distribution of variance $\sigma^2$. Hence the LSB extraction can be incorrect. This is illustrated in Fig. 4, where $d = 8$ and $\sigma = 3$. Notice that an error is seriously impeding the LSB attack as the attacker does guess the opposite of the LSB.

Still, in practice, identifying the leakage LSB is a complex operation, as the attacker shall infer two values (offset and bin_width) with great precision, which can be challenging when the traces are very noisy.

### B. Attack methodology for Problem #2

As mentioned in Sec. V-A, regarding TI, the LSB leakage does apply. However, this vulnerability is hard to exploit in practice. Therefore, we aim to analyze the security order beyond 1st order. Namely, we would like to show that, in terms of moments, the resistance order is driven:

- not by the *number $d$ of shares*, but
- by the *incompleteness order*.

In this respect, we leverage moments-correlating DPA (MC-DPA, recall Sec. II-C, because they target specifically a given leakage order (whether it is 1st-, 2nd-, 3rd- or 4th-order). In practice, MC-DPA is less efficient (in terms of number of traces to extract the key) than template attacks, but is useful to elucidate the existence of a vulnerability at a specific leakage order.

### C. Introduction to practical validations

In the next section VI, we deploy several dynamic analyses, i.e., analyses based on leakage measurement during the S-box evaluations. We show that the LSB analysis (solution of Problem #1) does work on ideal (synthetic) and HSpice traces. Then, we tackle Problem #2. Namely, we show the success of second-order MC-DPA when TI uses incompleteness of order-1 and $d = 4$ shares.

## VI. Theory Illustration and Experimental Results of Targeted Problems

**Experimental Setup:** In this research, we targeted the add-round-key and S-box operations in the first round of the PRESENT cipher protected with Threshold Implementation (TI). The PRESENT algorithm uses 4-bit substitution boxes (S-boxes), and when turned into a TI combinational netlist, it requires 4 shares for each component[1]. Hence the TI version of PRESENT S-box we study has 16 input bits and 16 output bits.

We then applied three sets of analyses at the simulation level; one based on the total number of toggling that the signals in the targeted implementation experience in each time stamp (1 ps intervals in our experiments), the second based on the Hamming weight of the included signals in each time stamp, and the third by using power traces extracted by HSpice simulations. Moreover, to show the applicability of our attack on real-silicon, we targeted the FPGA implementation of TI and showed how TI can be compromised by such an attack.

We perform two kinds of analyses: template attack (as per [28] – attack termed "Template-Based DPA Attack" in §2.3, and illustrated in Fig. 3), and MC-DPA (as per [24]).

In our HSpice simulation setup, totally we applied $200K$ input pairs to the circuit; each containing an initial value followed by a final value after 2 ns with the sampling rate of 50 GSamples/sec of the power traces when the final value is fed. While launching the attack on the real silicon, on an FPGA in our case, we applied $500K$ input pairs (we used more input traces in this configuration due to the existing of the measurement noise) where for every initial/final pair we sample the power traces after the final value is fed 2496 points in time (sampling rate of 5 GSamples/sec).

[1]Notice that it is possible to implement PRESENT S-box with $d = 3$ shares [29], but with a pipeline stage in its middle, which we want to avoid in our research. Indeed, other gate-level masking schemes (GLUT and ISW) manage to implement the PRESENT S-box with only combinational gates.

- All initial values are generated randomly, however, belonging to the class 0;
- All final values are generated randomly.

In our FPGA experiments, we targeted the FPGA implementation of the TI circuitry. We deployed a SAKURA-G board containing two Spartan-6 FPGAs. To reduce the noise impact, one FPGA was used to realize the controller part, and the other implemented the main logic, i.e., the measurement trigger signal, and the top state-machine were implemented in the controller FPGA, and also the input patterns were stored in the BRAM cells available in the fabric. The PRESENT add-round-key and the S-box were implemented on the main FPGA. The circuits operate at 48 MHz. Figure 5a depicts our setup. As mentioned, for each initial/final pair of input, we captured power samples after the transition of input from its initial to final values. To keep the FPGA implementation similar to its HSpice counterpart and its gate-level netlist, the post-synthesis netlist was used for FPGA implementation while avoiding the optimization by the Xilinx ISE tool. Apart from that, additional optimization attributes–(`keep_hierarchy = "yes"`)–shall be used to force the tool not to optimize out the area, and as a consequence, masks being removed. Table I depicts the resource utilization of our implementation.

TABLE I: Resource utilization of both main and controller FPGAs on SAKURA-G board.

| | Main FPGA | Controller FPGA |
|---|---|---|
| **LUTs** | 221 (0.5%) | 340 (5%) |
| **FFs** | 17 (4%) | 77 (22%) |
| **Slices** | 231 (2%) | 266 (18%) |
| **Block RAMs** | 0 | 32 (100%) |
| **Freq. (MHz)** | 48 | 48 |

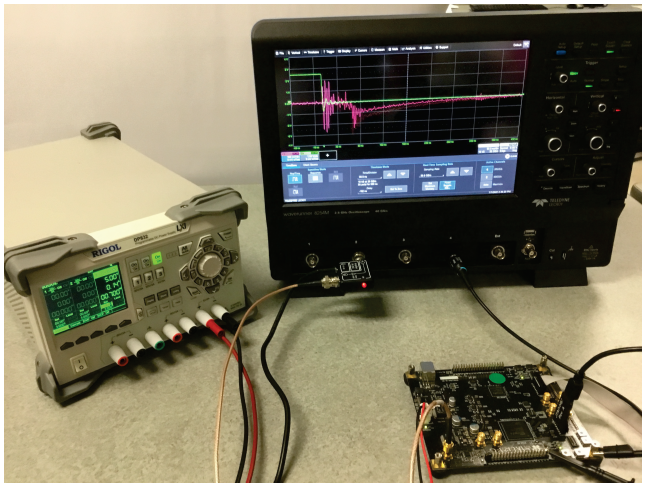
The aforementioned attribute was used next to all gate instantiations, so, the number of LUTs was exactly the same as the number of gates being utilized. This is not optimal in terms of gate count, but it allows to perform a comparison between HSpice and FPGA results. Also, because the design included the 1 and 2-input gates, while the native gates in FPGAs are 4-input LUTs (or even 5 or 6 input), FPGA traces experience more glitches due to the routing. Finally, power traces are amplified using Low Noise Amplifiers (LNAs) and sent out to the oscilloscope.

### A. Illustration of Problem #1

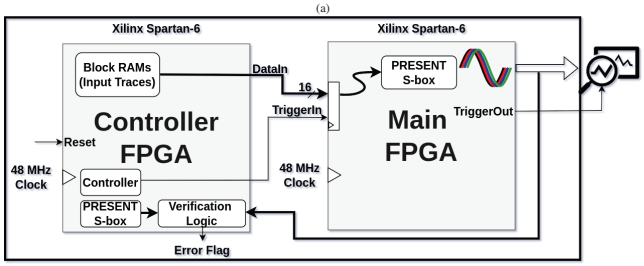
To demonstrate such an attack, we simulated power consumption traces based on the Hamming weight of the S-box output in our targeted TI circuitry.

Figure 6 shows, for all sixteen hypothetical keys (hence a $4 \times 4$ layout), the leakage distribution as a function of the unmasked sensitive variable. It clearly appears that a dependency exists between the secret value (the key) and the distribution profile of the leakage. Therefore, a template attack is expected to work. In addition, the order of the leakage can be quantified. From these graphs, we can see that (irrespective of the 16 key values):

- means are the same = 8 (hence no leakage),



(a)




(b)

Fig. 5: FPGA measurement setup. (a): Hardware. (b) Schematic.

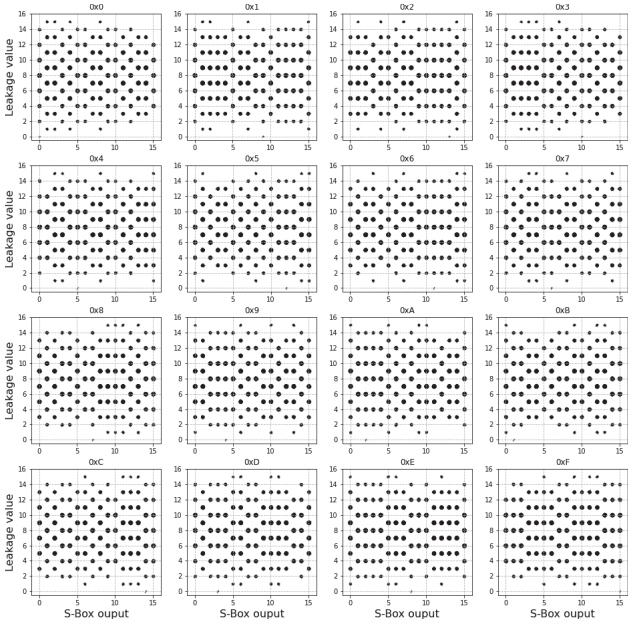



Fig. 6: Empirical leakage probability distribution function (PDF) for all key hypotheses extracted from synthetic simulations. Each graph manifests a different leakage profile. The x-axis shows the S-box outputs. A low Gaussian noise is also added to the S-box output class. In theory, the maximum value is 16, but owing to the S-box equations, the all-1 output never appears in our design.

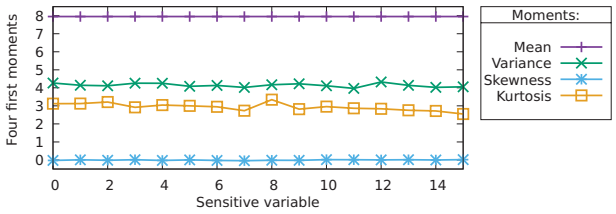
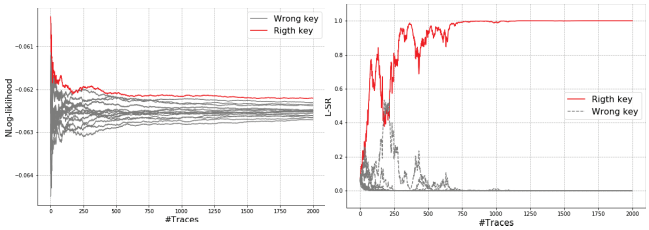



Fig. 7: First moments of the PDF (see Fig. 6) of our 1st-order incomplete TI.




(a) (b)

Fig. 8: Attack results on the synthetic traces. (a) Normalized likelihood over the 16 key hypotheses. (b) likelihood-based success rate (L-SR). The right key is shown in red. The best score is achieved by the right key (0xb) and can be distinguished with $Q = 500$ traces.

- variance differs according to sensitive variable (= abscissa axis value $\in \mathbb{F}_{16}$),
- skewness is the same (= 0) because distribution is symmetrical around mean = 8, and
- kurtosis differs according to sensitive variable (= abscissa axis value $\in \mathbb{F}_{16}$).

The moments are drawn in Fig. 7. We therefore expect that 1st- and 3r- order (resp. 2nd- and 4th-order) MC-DPA analyses work (resp. fail)[2]. To launch such an attack, we first profiled the 2D-leakage distribution with a known key. Then as a second step, using an unknown key dataset, we compared the reference distribution with the ones obtained for different key hypotheses using the likelihood metric. We note that equivalent results can be obtained using Euclidean distance metric.

Considering the distinguishability of the profiles for the theoretical model shown in Fig. 6, we can model the leakage $\mathcal{L}$, as a Gaussian mixture [20]. Thereby, for each sensitive value $y = \bigoplus_{i=1}^{d} y_i = S\text{-}box(x) \in \mathbb{F}_2^4$ and an observed (noisy) leakage $t(y)$, the PDF of the global leakage $\mathcal{L}(y)$ can be modeled by:

$$\mathcal{L}(y) = \sum_{i=1}^{m} w_i(y) \exp\left(-\frac{1}{2}\left(\frac{t(y) - \mu_i(y)}{\sigma_i(y)}\right)^2\right).$$

where $w_i(y)$ is the weight of the Gaussian component ($\sum_{i,y} w_i(y) = 1$) and $m$ is the number of leakage classes ($m = 4 \times 4 + 1$ in the theoretical model – Hamming weight of 16 bits, using TI with 4 shares). Let us assume a leakage model such as the traces $T$ are governed by the following law:

$$T = w_H(Y_1, \ldots, Y_d) + N,$$

where $(Y_1, \ldots, Y_d) = TI\text{-}S\text{-}box(X_1, \ldots, X_d)$ and $N \sim \mathcal{N}(0, \sigma^2)$ is the noise. Therefore, one has that:

$$\begin{aligned} w_j(y) &= \mathbb{P}(w_H(Y_1, \ldots, Y_d) = \mu_j) \\ &= \mathbb{P}(w_H(Y_1, \ldots, Y_d) = \mu_j | \bigoplus_{i=1}^{d} Y_i = y) \\ &= \mathbb{P}(w_H(Y_1, \ldots, \bigoplus_{i=1}^{d-1} Y_i \oplus y) = \mu_j) \\ &= \mathbb{P}(w_H(TI\text{-}S\text{-}box(X_1, \ldots, X_d)) = \mu_j | \bigoplus_{i=1}^{d} X_i = x) \end{aligned}$$

which depends only on the design of the masked S-box implementation. We have also $(T(Y) - w_H(Y_1, \ldots, Y_d) | w_H(Y_1, \ldots, Y_d) = \mu) \sim \mathcal{N}(0, \sigma^2)$, thus $\mathbb{P}(T(Y) - w_H(Y))$ is as equal to:

$$\sum_j \mathbb{P}(T(Y) - \mu_j | w_H(Y_1, \ldots, Y_d) = \mu_j) \mathbb{P}(w_H(Y) = \mu_j) = \sum_j w_j(Y) \mathbb{P}(T(Y) - \mu_j),$$

which yields: $T(Y) \sim \sum_j w_j(Y) \mathcal{N}(\mu_j, \sigma_j^2)$, where $\mu_j = \mu_i(y)$ and $\sigma_i^2 = \sigma_i^2(y)$ are respectively the mean and the variance (noise) of the class $i$.

Thus, for a key hypothesis $\hat{k} \in \{0, \ldots, 15\}$, we can evaluate its likelihood $L(\hat{k})$ based on the learned model by:

$$L(\hat{k}) = \prod_{x=0}^{15} \mathcal{L}\left(\text{S-box}(x \oplus \hat{k})\right)$$

or the log-likelihood $LL(\hat{k})$ by:

$$LL(\hat{k}) = \sum_{x=0}^{15} \log\left(\mathcal{L}(\text{S-box}(x \oplus \hat{k}))\right).$$

As the likelihood converges very fast to 0, we compute the log-likelihood instead. To normalize the convergence of the different key hypotheses scores, we adopt a normalized version, which therefore consists in dividing each score on the sum of the total scores (we refer to this metric as *NLL*, coined in Eqn. (2) of [28, §2.3]). And finally, to have a success-rate-like metric (we refer to it as *L-SR*), we calculate the normalized version of the real probability score by computing and normalizing the exponential of the final results of the log-likelihood. Formally, we have for each key hypothesis $\hat{k} \in \{0, \ldots, 15\}$:

$$NLL(\hat{k}) = \frac{LL(\hat{k})}{\sum_h LL(h)} \quad \text{and} \quad L\text{-}SR(\hat{k}) = \frac{L(\hat{k})}{\sum_h L(h)}.$$

At the next step, using the above leakage value, we launched a template attack. The result of the attack is shown in Fig. 8. As shown, the right key is distinguishable after processing $Q = 500$ traces. The curves of the likelihood success-rate get stable after 1200 traces. We also note that in this experiment, we added Gaussian noise to have an SNR equal to 4 as in the HSpice simulation and to slow down the attack. Otherwise, the key will be recovered with less than $Q = 10$ traces (hence success rate is hard to evaluate precisely).

As the S-box in TI scheme is protected at incompleteness order 1 with a sharing of 4, it would also be interesting to look at the leakages related to higher order moments, notably 2, 3, and 4.

[2]Notice that our goal is to use MC-DPA as an oracle to provide an evidence that 2nd- and 4th-order leakages result from the shape of the PDF distributions.

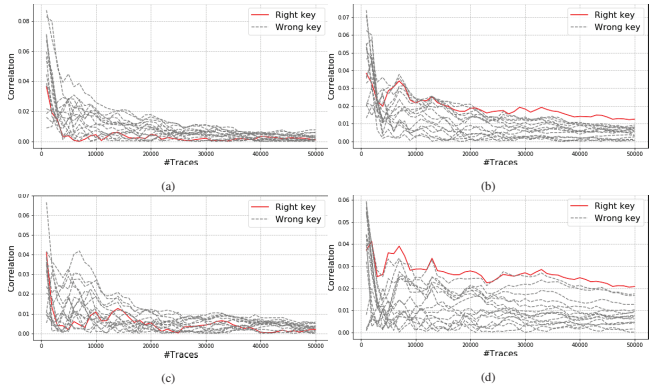



Fig. 9: Higher-order correlation results on the synthetic traces when launching MC-DPA attack. (a) First order. (b) Second order. (c) Third order. (d) Fourth order.

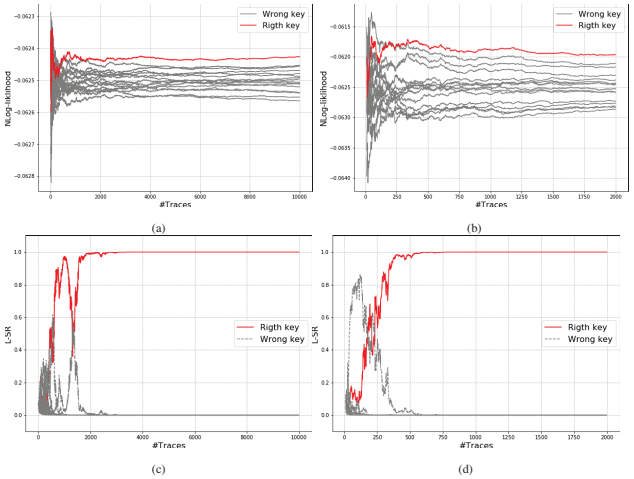



Fig. 10: Results of the template-based attack on the digital simulated traces (i.e., likelihood). The right key can be distinguished with 2000 traces when using dynamic power. (a,c): Attack targeting the dynamic leakage. (b,d): Attack targeting the static leakage. The attack works more easily in the case of static leakage.

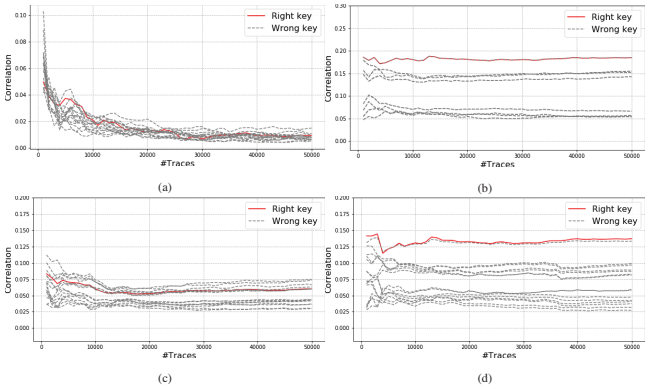



Fig. 11: Higher-order correlation results on digital traces (static power). (a) First order. (b) Second order. (c) Third order. (d) Fourth order.

MC-DPA attack results are shown in Fig. 9. As expected, the 2nd-order attack is successful. The takeaway point is that the order of incompleteness is the relevant security parameter, while the number of shares is not. Moreover, since the leakage is complex (recall Fig. 6), there is some still higher-order ($>2$) leakage. The success of the 4-order attack is related to the shape of the leakage model distribution and not the inclusion of 4 shares.

### *B. Illustration of Problem #2 and experimental results*

The previous results demonstrated a tiny leakage related to the Least Significant Bit (LSB). From the theoretical perspective, this manifests a weakness in all gate-level masking schemes including TI. Now we want to investigate practically whether this tiny leakage is exploitable for a secret key recovery. Thus, we carried out an extensive analysis at various abstraction levels all the way from digital netlists (with delay) to analog netlists (by HSpice simulations), and finally, real-silicon traces sampled by oscilloscopes from FPGA fabrics.

For the MC-DPA, we show the attack only on the static leakage, where the leakage can be exploited much faster compared to the dynamic part. Namely, for digital and HSpice simulations, we need 3 times more traces to distinguish the right key when considering dynamic power compared to its static counterpart.

*1) **Digital simulation**:* For the digital simulation, we have associated an identical propagation delay for the same gates, (but differently when the gates are not the same) to have a behavior relatively close enough to the real circuit (but without process variation). The traces are sampled each 100 ps. The power consumption is a combination of toggle count (dynamic leakage) and static state with 90% and 10% proportions respectively. We used artificial delays to model glitches, while at the same time ignoring the specificity of actual gates delays that do depend on the corners (process, voltage, temperature).

The outcome of this attack is shown in Fig. 10. The right key is distinguishable after processing around 700 traces (resp. 2000) in the case of static leakage (resp. dynamic leakage), where the likelihood success-rate reaches 100% after processing 500 traces (resp. 800) for static leakage (resp. dynamic leakage).

The results of conducting the MC-DPA are shown in Fig 11. The takeaway point from this figure is that for the synthetic traces (similar to the theoretical results) the higher order leakage is exploitable. It additionally informs us that the number of traces to succeed both 2nd- and 4th-order attacks are less than 1000 when exploiting static power.

*2) **Transistor level analysis (HSpice simulation)**:* In the next step, we targeted the TI implementation using HSpice power traces. Similar to the toggle count traces, the HSpice 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 final value. For the analysis, we considered only the power samples of the second part, i.e., when the cryptographic circuit transitions from the initial to the final value.

Every input initial and final segments stay on the bus for 2 ns while being sampled 100 times with a sampling rate equal to 20 ps. The simulations were conducted for the temperature

of 85°C, $V_{dd}$ = 1.2 V, and a operating frequency of 500 MHz using 45-nm technology extracted from the open-source NANGATE library [1] with the Synopsys HSpice analog simulation tool. The outcome of this attack is shown in Fig. 12.

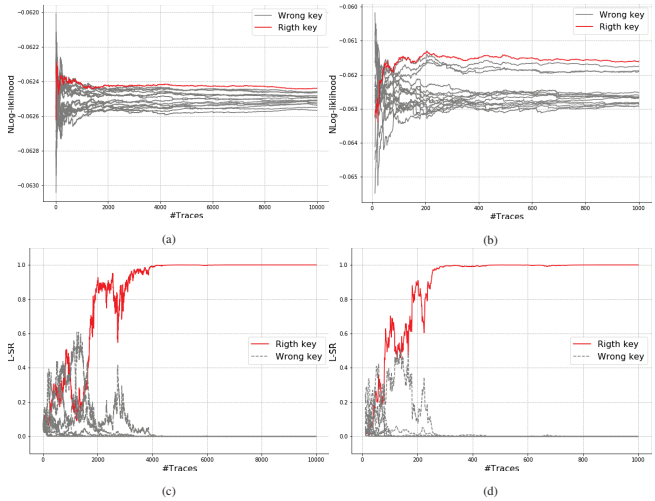



Fig. 12: Results of the template-based attack on the HSpice simulated traces. The right key can be distinguished with 8000 traces when exploiting dynamic power leakage. (a,c): Attack targeting the dynamic leakage. (b,d): Attack targeting the static leakage. The attack works more easily in the case of static leakage.

The results of conducting the MC-DPA are shown in Fig 13. As the analysis results show, all the theoretical, synthetic, and HSpice simulations agree; at every simulation level, the 1st- and 3rd-order attacks fail whereas the 2nd- and 4th-order attacks succeed. Based on the study of synthetic leakage (recall Fig. 6), we see that PDF distributions are all almost symmetrical with respect to mean = 8. This explains in our case why there is no (or at least little) leakage at order 3.

Note that those results are specific to the netlist. Indeed Moradi et al. showed that attacks were successful in both 2nd- and 3rd-order (Fig. 5 in [24]) but they did not investigate it further. Specifically, the original MC-DPA article [24] does not answer the question of the origin of the leakage. The natural question regarding the success of 2nd-order MC-DPA is:

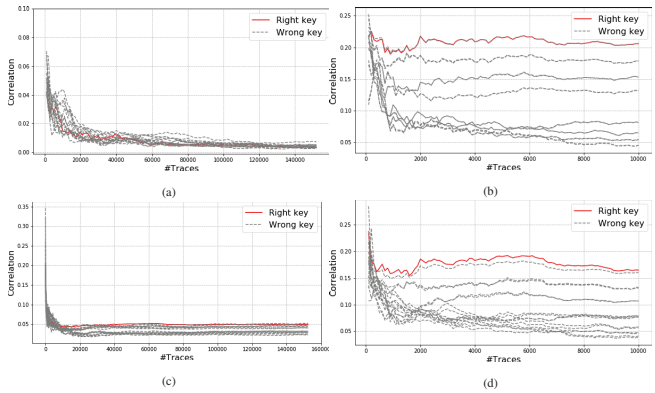



Fig. 13: Higher-order correlation results on HSpice traces (static power). (a) First order. (b) Second order. (c) Third order. (d) Fourth order.

- Is it due to an implementation bias (e.g., some strange ground coupling operated within the FPGA? Or some unexpected interaction through the powerlines? etc.), or
- Is it innate to the countermeasure?

We answer to this question: the problem is indeed innate to order-one incomplete TI, where the PDF features variances (= 2nd-order moments) that do depend on the sensitive variable.

*3) **Real analysis on FPGA**:* To investigate our findings on real silicon, we extracted power traces form the TI circuitry implemented on the Spartan 6 Xilinx FPGA. As mentioned earlier, we deployed SAKURA-G board [14] that includes two FPGAs on a single PCB. As the controlling modules and the main TI have been implemented on separate FPGAs, the extracted power traces are almost related to the TI module itself and are not affected by the controlling modules. In these experiments, every initial and final values are stable on the bus for around 500 ns while the power traces are sampled every 200 ps after feeding the circuit with the final values.

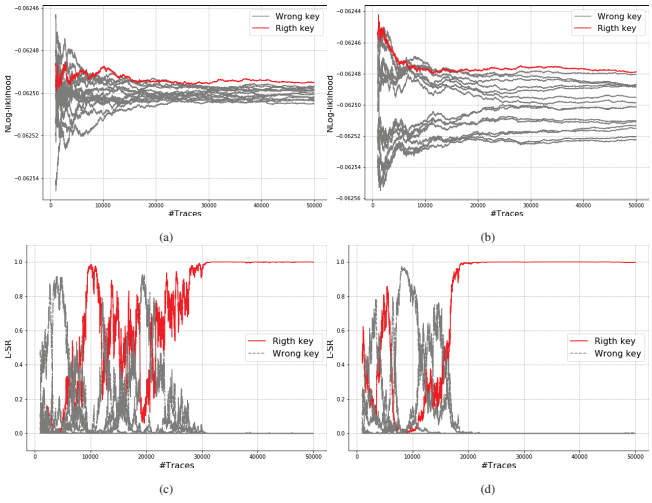



Fig. 14: Attack results on the FPGA simulated traces. The right key can be distinguished with 8000 traces. (a,c) Attack targeting the dynamic leakage. (b,d) Attack targeting the static leakage. The attack is easier in the case of static leakage.

The outcome of this attack on the FPGA traces is shown in Fig. 14. As shown, the number of traces required to recover the key in this case is much higher compared to the attacks launched on digital and HSpice traces due to the measurement noise in real-silicon (FPGA). Here, more than 30,000 traces are needed to get the result of the likelihood stable, and for the right key to become distinguishable.

The results for the FPGA traces are also consistent with all the other results extracted from the theoretical, synthetic, and HSpice simulations discussed previously. The takeaway point from these observations is that FPGA emulation completely follows the digital and analog (i.e., HSpice) simulations.

Fig .15 and Fig .16 depict the outcome of MC-DPA attacks using static and dynamic leakages, respectively. As shown, the 2nd- and 4th-order attacks using static leakage are successful. We also note that exploiting the higher-order moment is more difficult when the noise is higher. This is why the MC-DPA is not successful using more than 250,000 traces for the dynamic

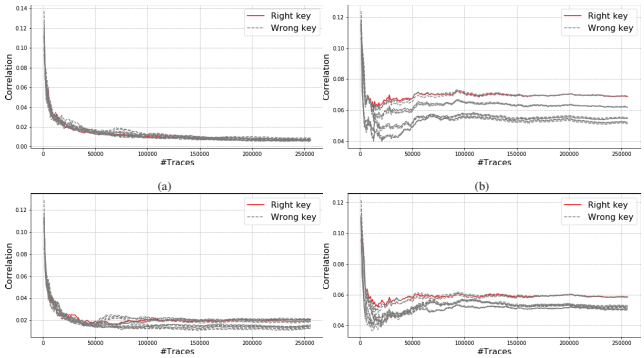



Fig. 15: Higher-order correlation results on FPGA traces – Static leakage. (a) First order. (b) Second order. (c) Third order. (d) Fourth order.

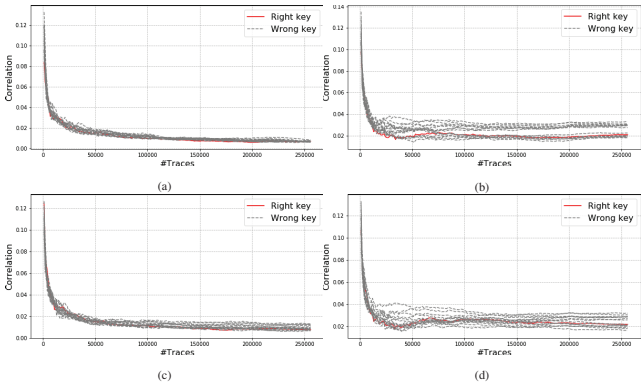



Fig. 16: Higher-order correlation results on FPGA traces – Dynamic leakage. (a) First order. (b) Second order. (c) Third order. (d) Fourth order.

leakage as shown in Fig .16. However, in this case the number of key hypotheses is just 3 when using 250,000 traces for the 2nd- and 4th-order attacks. Thus by increasing the number of traces these two attacks will be also successful and follow the same trend as our other experiments discussed earlier.

The reason for such undertaking is the validation of the practicality of the proposed attack in real silicon, where (unknown and uncontrollable) measurement noise is likely to hide the leakage in LSB (recall Prop. 1).

As the outcome of the attacks showed, the 3rd-order attacks failed in both HSpice and FPGA simulations while the 4th-order attacks were successful. This can be accounted for by computing leakage detection metrics. Let us recall the per-class definition of *skewness* $\gamma_x$ and *kurtosis* $\kappa_x$ moments:

$$\gamma_x = \mathbb{E}((X-\mu)/\sigma)^3 \quad \text{and} \quad \kappa_x = \mathbb{E}((X-\mu)/\sigma)^4.$$

In these equations, $\mathbb{E}$ is the expectation, $\mu$ (resp. $\sigma^2$) is the per-class leakage mean (resp. variance), and $X$ is the trace leakage (seen as a random variable). Namely: $\mu = \mathbb{E}(X|x)$ and $\sigma^2 = \mathbb{E}(X^2|x) - \mu^2$. The skewness measures the distortion of the distribution towards left or right compared to its mean. The kurtosis measures how concentrated the distribution is around its mean. A variation of skewness (resp. kurtosis) across classes indicates an exploitable bias in the leakage third-order (resp. fourth-order) moment. Therefore, we measure the variability of those statistics as their variance. They are represented in Fig. 17 and 18. As shown, the variability of the leakage skewness (which is the statistical leakage detection tool for the 3rd-order attack) is much less than the variability of kurtosis (which relates to the leakage detection of the 4th-order attacks). Note that the Y-axis range is different in these two figures for better visibility. This explains the reason why the 3rd-order attack failed on both HSpice and FPGA traces while the 4th-order attack succeeded in those cases.

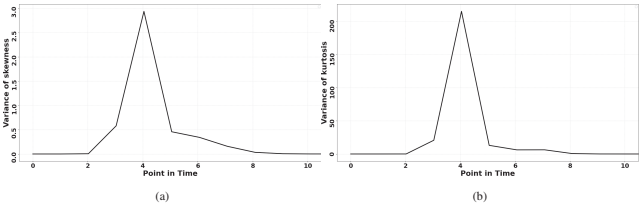



Fig. 17: Variance of (a) skewness $\gamma_x$ and (b) kurtosis $\kappa_x$ of HSpice traces.

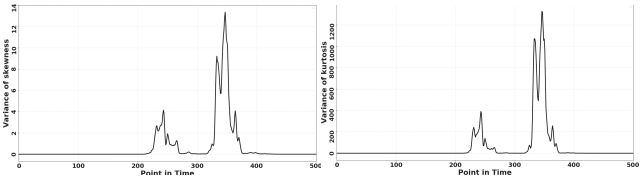



Fig. 18: Variance of (a) skewness $\gamma_x$ and (b) kurtosis $\kappa_x$ of FPGA traces.

## VII. Conclusions

This paper analyzes the security level of gate-level masking schemes. It identifies an innate vulnerability of such schemes: the Hamming weight of the (random) sharing is disclosing (deterministically) the sensitive bit value. This is based on our Proposition 1: namely, any gate-level masking scheme reveals its (unmasked) sensitive variable if the attacker is able to measure the Least Significant Bit of the leakage. We show the feasibility of the attack on synthetic traces, and explain how it is captured by template attacks.

Subsequently, we analyze a particular gate-level masking scheme, namely Threshold Implementation (TI). We show that the first-order leakage is hardly exploitable in practice, but further study the structure of the leakage. TI thwarts leakage arising from combinational logic (so-called glitches), by leveraging incompleteness property, which comes at the expense of extra sharing. Nonetheless, we show a second-order attack, which applies irrespective of the number of shares. This attack is validated both on simulated traces (ideal, without noise) and on real traces (noisy, captured from an FPGA). Our finding is modeled mathematically: we provide a theoretical explanation for the noting that correlated moments of first-order cannot break first-order TI, whereas second-order correlated moments do. As the future direction, we will analyze other masked implementations, including DOM which is similar to TI in terms of the unexploitability of glitches, to generalize our analysis.

## REFERENCES


[1] Nangate 45nm open cell library. "http://www.nangate.com".

[2] Javad Bahrami, Viet Ba Dang, Abubakr Abdulgadir, Khaled N. Khasawneh, Jens-Peter Kaps, and Kris Gaj. Lightweight Implementation of the LowMC Block Cipher Protected Against Side-Channel Attacks. *ACM Workshop on Attacks and Solutions in Hardware Security*, 2020.

[3] Javad Bahrami, Mohammad Ebrahimabadi, Jean-Luc Danger, Sylvain Guilley, and Naghmeh Karimi. Leakage power analysis in different s-box masking protection schemes. In *2022 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1263–1268, 2022.

[4] Javad Bahrami., Mohammad Ebrahimabadi., Sofiane Takarabt., Jean-luc Danger., Sylvain Guilley., and Naghmeh Karimi. On the practicality of relying on simulations in different abstraction levels for pre-silicon side-channel analysis. In *Proceedings of the 19th International Conference on Security and Cryptography - SECRYPT,*, pages 661–668. INSTICC, SciTePress, 2022.

[5] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A More Efficient AES Threshold Implementation. In *International Conference on Cryptology in Africa*, pages 267–284, 2014.

[6] B. Bilgin et al. Higher-Order Threshold Implementations. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT*, volume 8874 of *LNCS*, pages 326–343. Springer, 2014.

[7] Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably Secure Masking of AES. In *Selected Areas in Cryptography*, pages 69–83, 2004.

[8] Éric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *CHES*, pages 16–29, 2004.

[9] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Optimal side-channel attacks for multivariate leakages and multiple models. *J. Cryptographic Engineering*, 7(4):331–341, 2017.

[10] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *CHES*, pages 13–28, 2002.

[11] Mohammad Farmani. *Threshold Implementations of the PRESENT Cipher*. PhD thesis, July 2017. WORCESTER POLYTECHNIC INSTITUTE; https://web.wpi.edu/Pubs/ETD/Available/etd-090617-125035/unrestricted/MFarmani.pdf.

[12] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *CHES*, pages 251–261, 2001.

[13] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In Begül Bilgin, Svetla Nikova, and Vincent Rijmen, editors, *ACM Workshop on Theory of Implementation Security*, page 3. ACM, 2016.

[14] Hendra Guntur, Jun Ishii, and Akashi Satoh. Side-channel AttacK User Reference Architecture board SAKURA-G. In *IEEE Global Conference on Consumer Electronics, GCCE*, pages 271–274. IEEE, 2014.

[15] Naina Gupta, Arpan Jati, Anupam Chattopadhyay, Somitra Kumar Sanadhya, and Donghoon Chang. Threshold implementations of GIFT: A trade-off analysis. *IACR Cryptol. ePrint Arch.*, page 1040, 2017.

[16] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, volume 2729 of *LNCI*. Springer, 2003.

[17] J. Josep Balasch et al. Consolidating Inner Product Masking. In *Asiacrypt*, pages 724–754. Springer, 2017.

[18] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 1996.

[19] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO*, pages 388–397, 1999.

[20] Kerstin Lemke-Rust and Christof Paar. Gaussian Mixture Models for Higher-Order Side Channel Analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *LNCS*, pages 14–27. Springer, 2007.

[21] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006.

[22] Thomas S. Messerges and Ezzy A. Dabbish. Investigations of Power Analysis Attacks on Smartcards. In Scott B. Guthery and Peter Honeyman, editors, *Workshop on Smartcard Technology*. USENIX Association, 1999.

[23] Amir Moradi and Oliver Mischke. On the Simplicity of Converting Leakages from Multivariate to Univariate - (Case Study of a Glitch-Resistant Masking Scheme). In *CHES*, pages 1–20, 2013.

[24] Amir Moradi and François-Xavier Standaert. Moments-Correlating DPA. In *Workshop on Theory of Implementation Security*, pages 5–15. ACM, 2016.

[25] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches. In *ICICS*, volume 4307 of *LNCS*, pages 529–545. Springer, 2006.

[26] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Non-linear Functions in the Presence of Glitches. In *ICISC*, volume 5461 of *Lecture Notes in Computer Science*, pages 218–234. Springer, 2008.

[27] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.

[28] Elisabeth Oswald and Stefan Mangard. Template Attacks on Masking — Resistance Is Futile. In *CT-RSA*, pages 243–256, 2007.

[29] Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-Channel Resistant Crypto for Less than 2,300 GE. *J. Cryptol.*, 24(2):322–345, 2011.

[30] Pascal Sasdrich, René Bock, and Amir Moradi. Threshold Implementation in Software - Case Study of PRESENT. In *Constructive Side-Channel Analysis and Secure Design*, volume 10815 of *LNCS*, pages 227–244. Springer, 2018.