Hardware Masking, Revisited

Thomas De Cnudde[1], Maik Ender[2] and Amir Moradi[2]

[1] KU Leuven, imec-COSIC, Belgium
[2] Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
thomas.decnudde@kuleuven.be,{maik.ender,amir.moradi}@rub.de

Abstract. Hardware masking schemes have shown many advances in the past few years. Through a series of publications their implementation cost has dropped significantly and flaws have been fixed where present. Despite these advancements it seems that a limit has been reached when implementing masking schemes on FPGA platforms. Indeed, even with a correct transition from the masking scheme to the masking realization (i.e., when the implementation is not buggy) it has been shown that the implementation can still exhibit unexpected leakage, e.g., through variations in placement and routing.
In this work, we show that the reason for such unexpected leakages is the violation of an underlying assumption made by all masking schemes, i.e., that the leakage of the circuit is a linear sum of leakages associated to each share. In addition to the theory of VLSI which supports our claim, we perform a wide range of experiments (based on an FPGA) to find out under what circumstances this causes a masked hardware implementation to show undesirable leakage. We further illustrate case studies, where publicly-known secure designs exhibit first-order leakage when being operated at certain conditions.

Keywords: Masking · Leakage · FPGA · Hardware

1 Introduction

The presence of side channels is a well-known weakness in the security of embedded cryptosystems. For every successful Side-Channel Analysis (SCA) attack exploiting a certain side channel, e.g. the sensitive data-dependent timing of a computation [Koc96] or its power consumption [KJJ99], various countermeasures have been constructed to thwart this vulnerability. Considerable attention has been devoted to the mitigation of the power and electromagnetic wave [GMO01, QS01] side channels as they are effective and cheap to mount, e.g., in the form of Differential Power Analysis (DPA) [KJJ99].

One way to categorize countermeasures against DPA is on how they decrease the sensitive information exposed in the side channel, or put differently, how they decrease the signal-to-noise ratio (SNR) captured by the measurements [MOP07]. Countermeasures based on secure logic styles, a subset of the hiding family of countermeasures, decrease the SNR by equalizing the data-related power consumption of a circuit implementation [TAV02, TV04]. An alternative way to decrease the SNR is to increase the noise component of the signal rather than reducing the informative signal component in the side channel. Another subset of the hiding countermeasures increases the noise component by randomly shuffling the operations in time [VMKS12]. The masking family of countermeasures pursues this latter path by processing on algorithmically-randomized data while maintaining the overall correctness of the circuit [CJRR99, GP99].

It has been shown in many works that equalizing the power consumption is challenging to achieve as strict assumptions need to hold (e.g., no early signal propagation, no

IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925, Vol. 2018, No. 2, pp. 123–148
DOI:10.13154/tches.v2018.i2.123-148

imbalanced routing) [MOP07], and becomes increasingly harder to attain in advanced technology nodes as parasitic effects increase [NKSF17]. Ad-hoc countermeasures that introduce noise by shuffling suffer security issues as well, as they are easily defeated by preprocessing the measured traces. Masking on the other hand offers provable security, and can therefore be made more robust against these issues [PR13].

A d^{th}-order Boolean masking represents all sensitive variables, e.g., x, in an implementation by $d+1$ shares such that $x = \bigoplus_{i=1}^{d+1} x_i$. The computations are then performed on the shares and at no point in time the values are unmasked. The number of shares a variable is split in is related to the order of the masking scheme and determines the effort an attacker has to invest to break the system. To perform a successful key-recovery attack on a d^{th}-order masking scheme, an attacker needs to observe at least the $d+1^{th}$-order statistical moment of a set of leaking shares, or observe the leakage of $d+1$ individual shares. The universal underlying assumption all masking schemes make, i.e., that the leakages of different shares are independent of each other, is thus crucial as otherwise the statistical moment of the set of leaking shares, or the number of observed shares can contain more information than allowed by the theory.

In order to mask a real-world implementation, a developer has the choice to implement the cryptosystems in either software or hardware. Software, at least traditionally, is inherently sequential. As a result, it becomes increasingly more expensive in terms of both timing and code size to implement higher-order masking schemes in software [GR17]. The flexibility (at higher cost) that hardware offers with respect to both customizing the operations of a core and its inherent parallelization is more suitable when speed and throughput are a limiting constraint. Masking schemes for hardware and software follow different requirements for optimization.

Motivation. Many hardware tailored masking schemes have been developed but mainly due to wrong assumptions not all of them result in actual secure designs. An illustration is reflected by some pioneering schemes [CJRR99, ISW03, Tri03, OMPR05, CB08] that assume transistor gates to execute in a sequential manner, which was shown to be overly optimistic due to glitches and early propagation of signals [MPO05, MOP07, MME10]. In order to make masking as independent of the target platform as possible, Nikova et al. introduced the Threshold Implementations (TI) masking scheme that inherently resists the security deterioration emerging from glitches [NRS11]. Optimizations building on TI have been presented in the form of Consolidated Masking Scheme (CMS) [RBN+15] and Domain-Oriented Masking (DOM) [GMK16, GMK17, GM17] to decrease the area overhead as well as the required randomness (both have an influence on the actual implementation cost). These recent developments towards the implementation and validation of masking schemes employing the theoretical minimal number of shares have been validated using theoretical measures, e.g., by toggle counts or formal tools [Rep16, BGI+18].

In practice, however, it has been observed by different researchers that such first-order masked implementations still exhibit first-order leakage when evaluated on FPGA [BPG18, PS16, pri], and a first hint as to its cause was given in [DBG+17] where it is shown that using an identical netlist, leakage can be induced through a particular placement and routing configuration. In this work, we scrutinize the reason behind such undesirable mismatch between theory and practice by showing the incorrectness of a fundamental assumption in masking schemes (explained below).

Our contribution. Suppose (x_1, x_2) form a 2-sharing representation of the secret value $x = x_1 \oplus x_2$. A universal assumption all hardware masking schemes make dictates that the leakage of the corresponding hardware is a linear sum of the leakages corresponding to each individual share, i.e., $\mathsf{l}_1(x_1) + \mathsf{l}_2(x_2)$. In this work, we show that this assumption does not necessarily hold in practice, as was hinted in theory in [MOP07]. More precisely, we show that the power consumption of a function operating on a share influences the amount of power consumption of other functions simultaneously operating on other shares. That means, the device leakage is not necessarily the sum of independent leakages associated to each share.

We note that this behavior is indeed motivated by the work in [GOKT16] showing that the activity of a circuit placed in a region on an FPGA causes fluctuations on the power supply voltage at other, unrelated/unconnected regions of the FPGA. Based on this concept, the recent work by Schellenberg et al. [SGMT18] (and later in [ZS18] and [RPD+18]) show that sensors can be built inside FPGAs to capture dynamic power/energy consumption of the FPGAs, and thereby conducting actual DPA attacks.

Apart from the theory of electronics justifying our claims, we present practical results (based on FPGA experiments) supporting our findings. An interesting research question that follows from our research is why several designs presented in the literature, e.g., [DRB+16, CFE16, MW15, PMK+11, DBR+16, BGN+15], did not observe first-order leakage, and reported the security of their constructions using e.g. non-specific t-test SCA evaluation [GJJR11, CDG+13, SM15]. We give closure to this arising questions by showing under which conditions the problem we point out leads to detectable leakage.

Although we show that this issue is independent of the security order the underlying masking scheme employs, we stress that it is more easily observable when the minimum number of two shares is chosen for a first-order secure implementation. We opt for leakage detection (as opposed to key retrieval) and an "artificial" lab environment (as opposed to "practical" measurements) in order to pinpoint the existence and sources of leakage in correctly-implemented masking schemes, which we attribute to platform-specific factors. We believe that this work can pave the way towards successful key recovery exploiting these leakage sources.

Related works. A theoretical analysis was performed in [DFS15] where it was shown that the security of masking schemes decreases with increasing magnitude of the joint leakage between different shares. Both crosstalk and power supply noise are physical sources that can lead to these degrading joint leakages.

Crosstalk arises from coupling capacitance between wires which is influenced by the switching activity and the values on the wires. Its effect on the power consumption is used in [DWS10, DWS12] to derive a more precise leakage model of an 8-bit bus. More recently, the delay introduced on wires by the crosstalk capacitances is used as a source of information leakage in [GE16].

Power supply noise was shown to couple logic gates in a countermeasure against timing violations where the gates are assumed to be independent, which led to a negative impact on its security [ZED+15]. Furthermore, coupling of different parts of a circuit through the power supply has recently been exploited as a hardware Trojan to leak side-channel information on an FPGA [SGMT18]. Specifically, in the context of masking, the effect of placement and routing was shown to influence the leakage of masked implementations, but no particular source for the caused leakage was isolated [DBG+17].

2 Background

2.1 Masking

A masking scheme defines what secret sharing scheme is used to split sensitive values and how to perform operations on these shares in a secure way (i.e., multi-party computation). In conjunction with a set of assumptions on the leakage behavior of the target device, a masking scheme offers security at a given order d.

Boolean masking in particular adopts the binary addition to share the sensitive variables: a sensitive value $x \in GF(2^m)$ is shared in $d+1$ shares $(x_1, ..., x_{d+1})$ such that $x = \bigoplus_{i=1}^{d+1} x_i$. The shares need to be uniform in order for the masking scheme to be secure. This can be achieved by drawing shares $x_1, ..., x_d$ from a uniform random distribution and by choosing

x_{d+1} such that the sharing is correct, i.e., $x_{d+1} = x \bigoplus_{i=1}^{d} x_i$. This is formulated as the correctness property and the uniformity property.

With Boolean masking, it is easy to see that securely evaluating linear and affine functions is straightforward. The function $f(x, y) = ax + by + c$ with a, b and c constants in $GF(2^m)$ and variables $x, y \in GF(2^m)$ can be implemented in a d^{th}-order secure way as follows.

$$\begin{aligned} f_1 &= ax_1 + by_1 + c \\ f_i &= ax_i + by_i \qquad , 2 \leq i \leq d+1 \end{aligned}$$

The correctness of these component functions is easily checked as $f(x, y) = \bigoplus_{i=1}^{d+1} f_i = a \bigoplus_{i=1}^{d+1} x_i + b \bigoplus_{i=1}^{d+1} y_i + c = ax + by + c$.

Non-linear operations however require a more subtle approach for the masking scheme to preserve its d^{th}-order security. We use a masked multiplication as an example to show different approaches. In order to perform the masked multiplication $z = xy$, the cross products $x_i y_j, \forall i, j \in \{1, \ldots, d+1\}$ will have to be calculated. This gives a total of $(d+1)^2$ contributions that need to be taken into account to obtain the masked output and the question becomes how to reduce this number of shares back to $d+1$ in a secure way. The secure redistribution of the cross products is approached differently in different Boolean masking schemes and affects the performance in terms of area, amount of needed randomness and required clock cycles. We briefly illustrate two approaches: Threshold Implementations [NRS11] and Domain-Oriented Masking [GMK16].

Threshold Implementation. One masking scheme that has shown promise and practical applicability is the Threshold Implementation (TI) masking scheme. It has gained widespread popularity from its inherent ability to resist leakage from glitches through its non-completeness property. In contrast to the preceding masking schemes, like the ISW scheme [ISW03] or the Trichina AND-gate [Tri03], no evaluation order or limits on propagation delays has to be imposed for the security to hold [MPO05]. The non-completeness property dictates that "any d component functions should be independent of all unshared values to achieve d^{th}-order security" [BGN+14]. This way, no matter how many glitches occur in a component function, enough information to unmask any inputs is never leaked. By extension, no matter how many glitches occur in d component functions, no information (up to d^{th}-order) can be retrieved to fully recover the unmasked inputs.

Considering the aforementioned multiplier, an example with first-order security is given below.

$$\begin{aligned} z_1 &= x_1y_1 \oplus x_1y_2 \oplus x_2y_1 \oplus r_1 \oplus r_2 \\ z_2 &= x_2y_2 \oplus x_2y_3 \oplus x_3y_2 \oplus r_1 \\ z_3 &= x_3y_3 \oplus x_1y_3 \oplus x_3y_1 \oplus r_2 \end{aligned}$$

Note that here three shares are used to achieve the first-order security, and the uniformity is achieved by adding (uniform) fresh randomness r_1 and r_2. This cost can be reduced by increasing the number of shares, which helps achieving the uniformity, hence no need for fresh randomness. We refer the interested reader to the work of Bilgin [Bil15].

Domain-Oriented Masking. An alternative approach is to perform the refreshing and share compression in two clock cycles. First, the cross products are calculated and randomness is added in specific places (in a similar way as randomness is added in [BBP+16]). Second, the resulting terms p_i are synchronized in a register and followed by a compression stage, that reduces the $(d+1)^2$ shares back to $d+1$ output shares z_i.

An example with first-order security is given below.

$$\begin{aligned} p_1 &= x_1y_1 & & \\ p_2 &= x_1y_2 \oplus r_1 & \qquad z_1 &= p_1 \oplus p_2 \\ p_3 &= x_2y_1 \oplus r_1 & \qquad z_2 &= p_3 \oplus p_4 \\ p_4 &= x_2y_2 & & \end{aligned}$$

Compared to a traditional TI, the number of shares and randomness is reduced at the cost of an extra clock cycle. An extra condition on the input shares is that they should be independent. More information can be found in [GMK16, GMK17, GM17].

In both TI and DOM the underlying security model is the same. The scheme is secure if the underlying assumption holds: "the power consumption of the component functions must be independent of each other". If this assumption is violated in any way, e.g., through coupling, then the security of the scheme will deteriorate.

2.2 Supply Voltage Fluctuation

In the areas of Very-Large-Scale Integration (VLSI) in advanced technology nodes, transient voltage fluctuation has been reported among the factors with the most critical impacts on the circuit's timing margins [DWB15, ZWM+14]. The source of such transient fluctuations includes the circuit switching activity, clock gating as well as power gating [DWB15, GOJ+07]. This fluctuation in the supply voltage level affects the delay of those gates which are supplied by the same voltage source (Vdd), hence changing the circuit timing characteristics. It has been shown that the transient supply voltage drop has the highest impact on the circuit timing even compared to process variation and temperature variation [GOKT16].

To this end, voltage stability is commonly analyzed on the level of Power Distribution Networks (PDN)[ZWM+14, LSB11], consisting of several resistive, capacitive and inductive (RCL) components. Among the known sources for voltage drop, the main concern in advanced technology nodes is the change in current over time (known as di/dt drop) [DWB15, ZWM+14, ASM07, Lar99, MF04]. Such voltage drops can be as fast as the circuit operating frequency originating from the inductivities throughout the PDN. These di/dt drops fade out rapidly, e.g., after some nanoseconds, but a sudden increase in the circuit switching activity can lead to a remarkably-high voltage drop. In [GOJ+07] a fine-grained PDN of a quad-core processor is modeled, and the on-chip supply voltage fluctuation of different pads are obtained by means of SPICE simulations. It has been shown that if only one core runs, the local voltage drop on the same core is lower compared to the case when three cores run at the same time. This issue is not dedicated to ASICs; the FPGAs which are also based on modern nano-scale technology face the same challenge [GOKT16].

Related to this topic, we would like to remind how power consumption traces are measured in the context of SCA. Figure 1 shows one of the common options, i.e., placing a shunt resistor at the Vdd path of the chip and measuring the dynamic voltage drop over the resistor (AC coupled measurement). Note that it is the scenario that SCA evaluation boards (SASEBO, SAKURA [sak]) are designed for[1]. The shunt resistor in Vdd path is favorable (than in GND path) particularly in modern technology nodes since the Vdd core and Vdd I/O are supplied by separate voltage sources.

This way of measurement traditionally observes the changes in current flowing in the chip since they directly lead to changes in voltage over the shunt resistor. However, by this way, any change at the output of the Voltage Regulator Module (VRM) would also be captured. This means that the above explained voltage fluctuations due to the circuit

[1]See also the "Quick Start Guide" at http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html

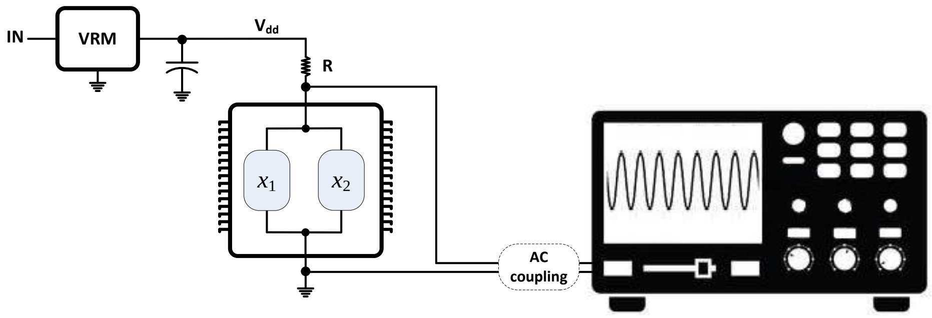


Figure 1: A typical SCA measurement circuitry.

activity are also observed in SCA power traces. Note that the VRM, which usually operates in switched mode and supplies the Vdd for the chip, is regulating for a configured voltage. Its regulation loop is, however, usually at a much lower frequency than that of the circuit activity, i.e., glitches happening in the combinatorial circuit. This means that any sudden switching activity by the chip faster than the regulation loop would lead to a voltage drop at the VRM output till it becomes again stable [GOKT16]. It has also been shown in [DWB15, GOJ+07] that a resonance between on-chip decoupling capacitors and resistive and inductive components can lead to even less voltage stability. Such transient voltage drops are also captured by the way that we collect power traces for an SCA. This means that even if the shunt resistor is replaced by a $0.0\,\Omega$ wire (`R` in Figure 1) the fluctuations on the VRM output caused by the circuit switching activity are captured and SCA power traces can be collected.

2.2.1 Voltage Fluctuation vs. Masking

Let's consider the first-order Boolean masking, where secret value x is represented by two shares x_1 and x_2. Assume also a linear function $\mathsf{f}(.)$ to be operated on each of the shares, to securely (up to first-order) compute $\mathsf{f}(x)$. The operation of each instance of such a function causes switching activity (e.g., glitches) and hence energy consumption. Let us model their corresponding exclusive power consumption by $\mathsf{l}_1(x_1)$ and $\mathsf{l}_2(x_2)$ respectively. As explained above, this leads to transient supply voltage drop due to the foundation of the PDN as well as the slow stabilization of the VRM. Therefore, when the sub-circuit computing $\mathsf{f}(x_1)$ is active, the resulting supply voltage drops affect the delay of the gates involved in the other sub-circuit (computing $\mathsf{f}(x_2)$), and hence its power consumption pattern. In other words, when these functions operate in parallel (see Figure 1) they have influence on each other's power consumption. Hence, the aforementioned assumption of their independent power consumption (in any masking scheme) does not hold. More precisely, we **cannot** model the total power consumption of the circuit by a linear sum of the power consumption of each sub-circuit as $\mathsf{l}_1(x_1) + \mathsf{l}_2(x_2)$.

Here, the question is whether such an effect (and violating the assumption of independent power consumption of the shares) can be relevant in practice. Specifically, with respect to the above-given example, can this circumvent the expected first-order security? In other words, would the total power consumption of the circuit **in average**[2] be dependent on x? In the next section, we present the result of our extensive experiments conducted to answer this question.

[2]The underlying concept of first-order security is the first-order statistical moment, i.e., the average.

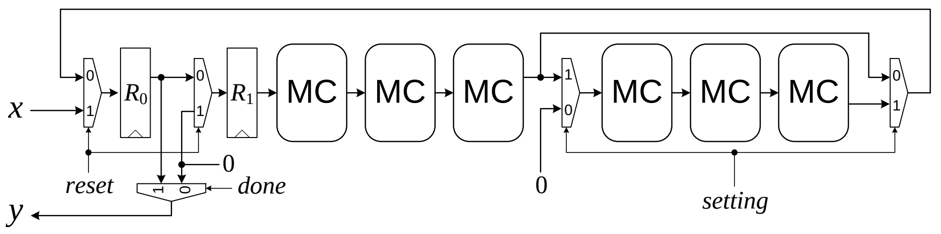


Figure 2: `iterated_MC` module.

3 Experiments

Setup. We conducted our practical experiments on the target FPGA (Spartan-6) of a SAKURA-G board [sak]. In all experiments and case studies, we measured the power consumption of the target FPGA through the Vdd path (exactly like what is shown in Figure 1) using a digital oscilloscope at a sampling rate of 500 MS/s. For the entire measurements, we monitored the output of the AC amplifier embedded on the SAKURA-G. As it is stated in the further parts of this section, the designs implemented on the target FPGA were clocked at different frequencies 6 MHz, 12 MHz, 24 MHz, or 48 MHz. Note that we externally supplied the clock for the target FPGA, and without changing the design (a fixed bitstream) we could run it at different clock frequencies.

As an evaluation metric to examine the existence of detectable leakage, we followed the procedure explained in [SM15] and conducted fixed-versus-random t-test. More precisely, the random numbers are generated by the control FPGA, by which the initial sharing, as well as the decision for random or fixed input, is also done on the control FPGA. The target FPGA just receives the shared input and sends back the output also in a shared form (with the same number of shares as of the input). We further kept **hierarchy** in all our implementations on the target FPGA to make sure that the non-completeness in our designs is not violated. Our focus is to investigate first-order leakage, but for the sake of completeness we also conduct univariate t-tests at higher orders depending on the order of the underlying masking scheme of each design.

3.1 Case Study 1

As the first experiment, we designed a module (so-called `iterated_MC`) shown in Figure 2 consisting of two 32-bit registers R_0 and R_1 and six MC modules, each of which is a realization of multiplication by the AES MixColumns matrix [DR02]. As shown by the graphics, during the reset phase the 32-bit input x is stored in register R_0 and the same time the second register R_1 is reset, i.e., filled by 0. In the next clock cycles the loop is closed, and $MC(0) = 0$ is stored in R_0 and x in R_1. This iteration leads to interleaving between 0 and a function of x at the input of the combinatorial circuit consisting of the MC modules. The *setting* signal selects either three or six MC modules to be active. The circuit is designed so that – independent of the given input x – the second three MC modules receive 0 when they should be disabled. By this we can control the depth of the combinatorial circuit, and hence the amount of its power consumption. We made sure that the *setting* bit is directly supplied by a register to avoid any glitches (unexpected power consumption) at the disabled MC modules. We further took attention that MC modules are not merged together, and each module is implemented separately. We keep this circuit running for 16 clock cycles. Since $MC^4(x) = x$, depending on the *setting* signal after either every 8 or 4 clock cycles again x is stored in R_1 and processed by the combinatorial circuit.

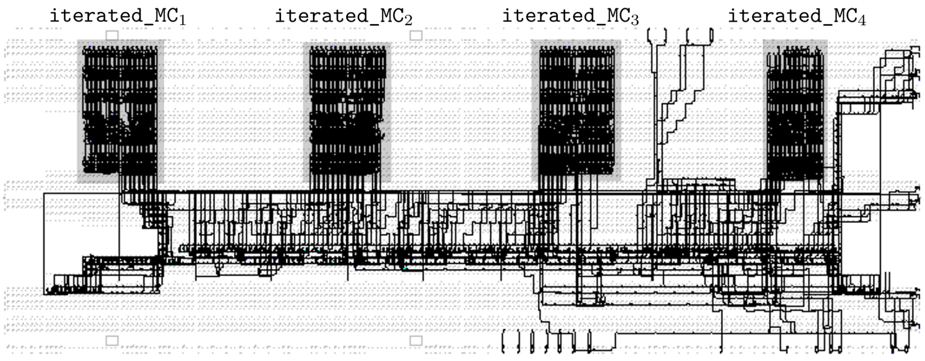


Figure 3: The floorplan of the target FPGA in case study 1 (rotated counter clockwise)[3]

As a side note, in our implementations the control FPGA sends x to the target FPGA in small (4-bit) chunks. The registers which store such chunks and provide the input of the first MUX in Figure 2 are reset simultaneously with x being stored in register R_0. After that (i.e., the reset phase) we further kept several clock cycles delay before enabling the circuit to iterate 16 clock cycles. Afterwards, the *done* signal becomes high and lets the content of R_0 be present at y signal. Similar to the *setting* signal, *done* is supplied directly by a register cell to avoid any glitches. All these considerations are to ensure that the leakages, which we observe later in this section, are not due to any other source than the `iterated_MC` module.

The motivations to select the AES MixColumns matrix include (1) its linear property making it transparent to Boolean masking, and (2) its implementation which contains only XOR gates leading to a glitchy circuit hence high power consumption. We have implemented four instances of such `iterated_MC` module at particular places in the target FPGA. As shown by Figure 3, we kept a remarkable distance between their placements, one at the most top and one at the most bottom part, keeping the maximum possible distance between $\texttt{iterated\_MC}_1$ and $\texttt{iterated\_MC}_4$. It is noteworthy that we made use of **Isolated Design Flow** (IDF) of Xilinx [McN] to guarantee that different `iterated_MC` modules do not share any resources including switch boxes involved in their routed signals. We should highlight that the routed signals which are externally connected to each `iterated_MC` module (can be seen in Figure 3) are x, y, *setting* and *done*, which are stable and independent of the given input when `iterated_MC` is activated.

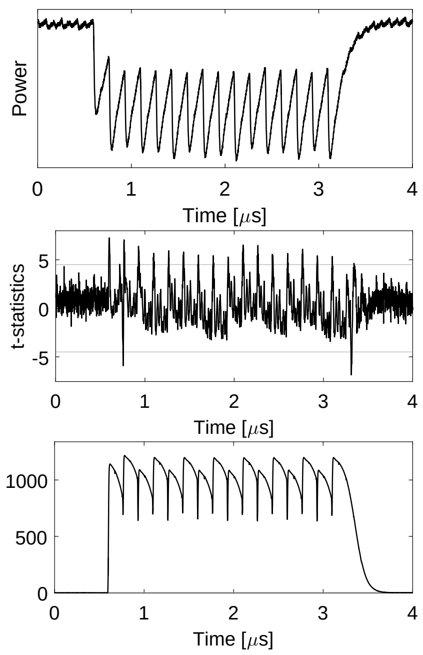
Below we consider several different parameters and conditions to examine their effect on first-order leakage. Note that for the entire experiments of this case study, we used a single design (i.e., a single bitstream for the target FPGA). We enabled or disabled different parts of the design (e.g., the *setting* signal) by means of particular configuration bits which are sent from the control FPGA to the target. Further, in order to disable a particular `iterated_MC` module, we kept its x input at 0.

Voltage Supply and Shunt Resistor. We first focus on first-order masking, i.e., with two shares. We selected the two most far modules $\texttt{iterated\_MC}_1$ and $\texttt{iterated\_MC}_4$ to process the two shares of the Boolean masked input, and enabled only the first three MC modules in the loop of each `iterated_MC` module (see Figure 2).

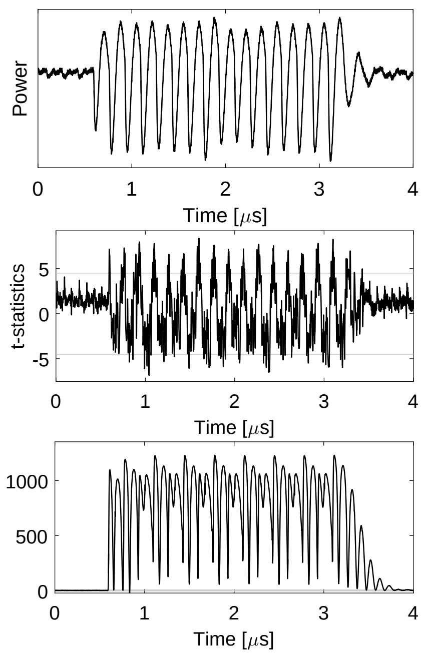
We performed five experiments being different in only the supply voltage of the target FPGA and the value of the shunt resistor:

- Vdd: 1.0 v, shunt: 1.0 Ω,

[3]The routed signals between $\texttt{iterated\_MC}_3$ and $\texttt{iterated\_MC}_4$ are due to the I/O pads of the FPGA fabric.



(a) Vdd 1.0 v, shunt 1.0 Ω



(b) Vdd 1.0 v, shunt 0.0 Ω

Figure 4: Case study 1: `iterated_MC`$_{1,4}$, 3 *MC* modules, clock 6 MHz, room temperature, (top) sample power trace, (middle) 1st-order and (bottom) 2nd-order *t*-test results using 100 million traces.

- Vdd: 1.0 v, shunt: 0.0 Ω,
- Vdd: 1.2 v, shunt: 1.0 Ω,
- Vdd: 1.2 v, shunt: 0.0 Ω,
- Vdd: 1.3 v, shunt: 0.0 Ω.

For the shunt 0.0 Ω, the resistor was replaced by a 0 Ω jumper[4]. Note that the Vdd was adjusted to the above values[5] when the shunt resistor was removed and the target FPGA did not sink any current. According to [Xil], the maximum V_{CCINT} for Spartan-6 is 1.32 v, hence we did not examine higher supply voltages. We should also mention that the SAKURA-G board is typically (by shipment) adjusted to supply the target FPGA with Vdd 1.2 v and an on-board soldered shunt resistor 1.0 Ω.

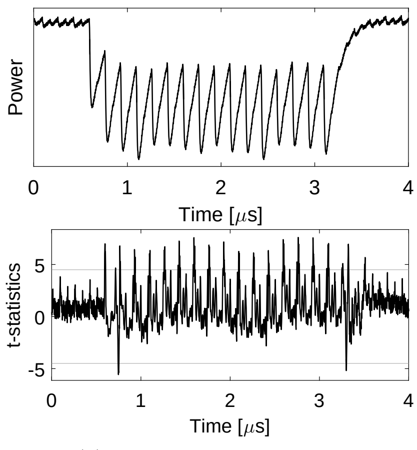
All five experiments have been conducted when the target FPGA was operating with a 6 MHz clock. For each case, we collected 100 million traces and – as stated before – performed fixed-versus-random *t*-test. The results, depicted in Figure 4 to Figure 6, surprisingly show detectable first-order leakage, particularly for high Vdd supply voltages and a low shunt resistor.

We should highlight that the underlying design is a linear function on Boolean masked data, hence independent of the masking scheme (i.e., TI, CMS, DOM, etc.). The `iterated_MC` modules neither communicate with each other nor share any resources, but they consume substantial energy. This in fact confirms the theory – stated in Section 2 – that energy consumption of shares are not independent of each other. An overview about the maximum *t*-statistics over the number of traces is shown by Figure 6b. As shown

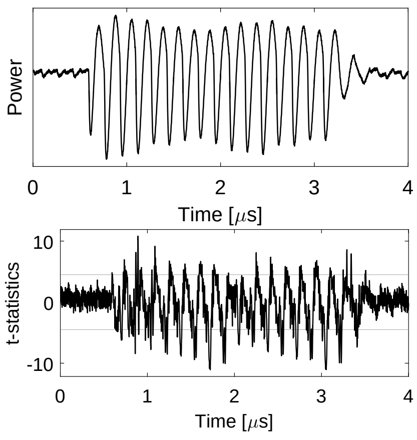
[4]The quality of the jumper to not exhibit any resistance was verified.
[5]by easily adjusting the trim potentiometer on the SAKURA-G board.

by this experiment, the Vdd that the FPGA is supplied with plays an important role in how easily the leakage can be detected. Placing the shunt resistor leads to a lower supply voltage that the FPGA observes, and hence hardening the leakage detection.

Note that such two-share designs strongly exhibit second-order leakage. As a reference, the t value exceeds the threshold after a couple of thousand traces, and reaches a value higher than 1000 considering all 100 million traces. Therefore, second-order attacks will be a natural choice of the attacker. However, we are presenting practical evidences of first-order leakage for the designs which are theoretically supposed to be first-order secure.

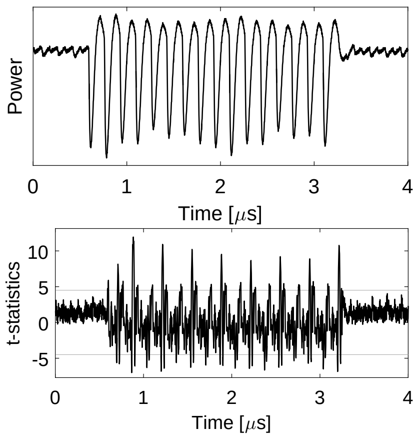


(a) Vdd 1.2 v, shunt 1.0 Ω

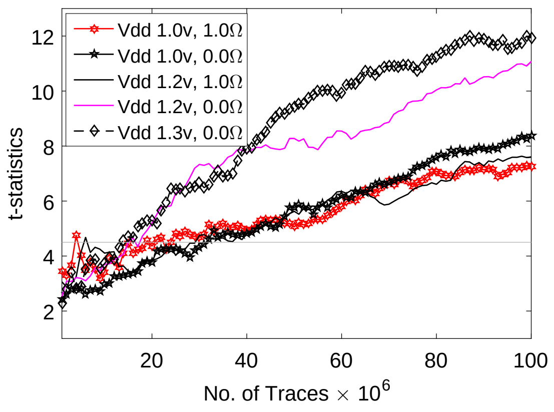


(b) Vdd 1.2 v, shunt 0.0 Ω

Figure 5: Case study 1: `iterated_MC`$_{1,4}$, 3 MC modules, clock 6 MHz, room temperature, (top) sample power trace, (bottom) 1st-order t-test results using 100 million traces.



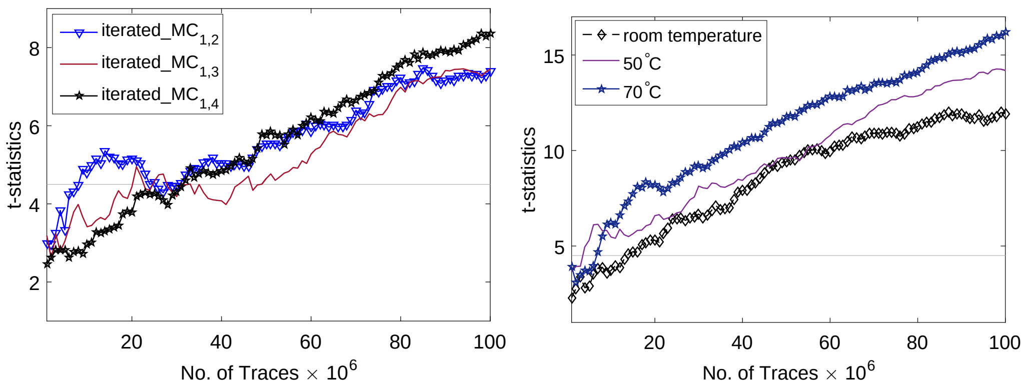
(a) Vdd 1.3 v, shunt 0.0 Ω



(b) $max(|t|)$ 1st-order, over the number of traces

Figure 6: Case study 1: 2 shares, `iterated_MC`$_{1,4}$, 3 MC modules, clock 6 MHz, room temperature, (left, top) sample power trace, (left, bottom) 1st-order t-test results using 100 million traces.

Distance. Let $x_{i\in\{1,\ldots,4\}}$ be the input of the `iterated_MC`$_i$ module. Staying with first-order masking (two shares), we considered three different settings: (x_1, x_2), (x_1, x_3), and (x_1, x_4) as the two shares of the Boolean masked input. More precisely, with two shares we



(a) room temperature, Vdd 1.0 v, different distances(b) `iterated_MC`$_{1,4}$, Vdd 1.3 v, different temperatures

Figure 7: Case study 1: 2 shares, shunt 0.0 Ω, 3 MC modules, clock 6 MHz, $max(|t|)$ 1st-order, over the number of traces.

conducted three experiments while processing the first share always with `iterated_MC`$_1$, but giving the second share to either `iterated_MC`$_2$, `iterated_MC`$_3$, or `iterated_MC`$_4$. For all these experiments only the first three MC modules in the loop were active.

The target FPGA was operating at a 6 MHz clock, with Vdd 1.0 v, and shunt resistor 0.0 Ω. We again collected 100 million traces for each experiment. The results, shown in Figure 7a, indicate that when the `iterated_MC` modules are close to or far from each other, the result is not much different. From here on we only show the evaluation results over the number of traces. We should emphasize that we conducted another experiment when the two `iterated_MC` modules were placed as close as possible[6]. The result did not show higher detectable leakage than the former three experiments.

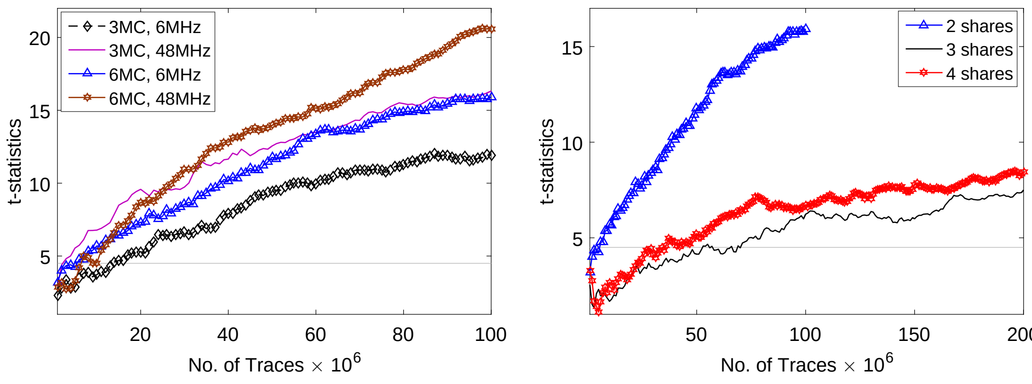
For the rest of the investigations (unless otherwise stated) we kept the following conditions: 2 shares processed by `iterated_MC`$_1$ and `iterated_MC`$_4$, Vdd supply 1.3 v, and shunt resistor 0.0 Ω.

Temperature. All the above experiments have been done at room temperature, that was controlled at around 21°C. We also used a climate chamber to operate the device at higher temperate. In these experiments we enabled only the first three MC modules in each loop. Among the measurement setup, only the SAKURA-G board was placed in the climate chamber. We performed the same experiments at 50°C and 70°C. The results are shown in Figure 7b, which express the extreme effect of temperature on the detected first-order leakage. Note that we have verified that the output of the voltage regulator of the SAKURA-G stays at 1.3 v at those high temperatures. We also note that the quality of the random numbers generated by the control FPGA is independent of the temperature.

Circuit Size and Clock Frequency. In an another experiment, by adjusting the *setting* signal (see Figure 2) we enabled all six MC modules in the loop. Hence, the circuit expectedly consumed more energy at every clock cycle, i.e., higher peak-to-peak power consumption traces. As stated before, in the previous experiments the target FPGA was clocked at the frequency of 6 MHz. By increasing the clock frequency, the peak-to-peak power consumption also becomes higher. Therefore, we conducted three more experiments:

- three MC modules active, 48 MHz clock,
- six MC modules active, 6 MHz clock,

[6]but with a row of slices in between to employ a physical gap [HBW+07, Cor].



(a) 2 shares, `iterated_MC`$_{1,4}$, different circuit sizes and clock frequencies (b) 6 *MC* modules, clock 6 MHz, different number of shares

Figure 8: Case study 1: Vdd 1.3 v, shunt 0.0 Ω, room temperature, $max(|t|)$ 1st-order, over the number of traces.

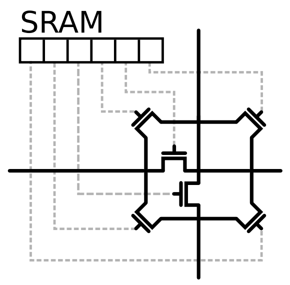
- six *MC* modules active, 48 MHz clock.

The corresponding results, compared to the former case (with three *MC* modules active, 6 MHz clock), are shown in Figure 8a. It can obviously be seen than by either increasing the clock frequency or more glitchy combinatorial circuit (where in both cases more energy is consumed), the leakage can be detected with lower number of traces.

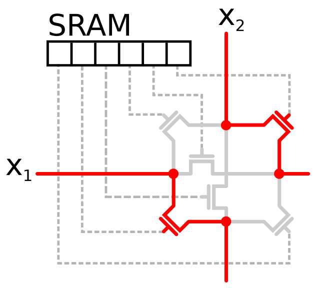
Number of Shares. As the last experiment of this case study we examined the effect of higher-order masking. With a 6 MHz clock frequency, and when all six *MC* modules were active in each `iterated_MC` module, we checked the second- and third-order Boolean masking. More precisely, with 3 shares we employed `iterated_MC`$_1$ to `iterated_MC`$_3$ and for the 4-share case we used all four `iterated_MC` modules. We have noticed that in these two cases the first-order leakage is harder to detect compared to the 2-share settings. Therefore, we extended our analyses to use 200 million traces. The results are presented in Figure 8b. We should highlight that with 3 shares we did not observe any second-order leakage, and no second- and third-order leakages in case of the 4-share settings (see Appendix A).

Discussions. By the above experiments we practically showed under which conditions the unexpected and undesired first-order leakage of just linearly processing the masks (at different orders) can be detected. Although it confirms the theory that energy consumption of processing the shares are not independent (see Section 2), we should highlight the following fact.

In the above experiments, only linear functions (several AES MixColumns) have been realized, and no non-linear function operates on shares. At the same time, no other circuit or module were active in the FPGA during the measurements. This leads to a very focused leakage, i.e., without any noise by other components (switching noise [MOP07]). We further, amplified the leakage by intentionally enlarging the glitchy combinatorial circuit. All these points helped us in our experiments to detect the first-order leakage. However, it is not necessarily the case in every masked implementation of a cipher due to the measurement noise, conditions of the measurement setup, the switching noise of the other components of the design, etc. Nevertheless, this problem is easily observed in masked implementation of any cipher with the minimum number of two shares. To the best of our knowledge, the first-order leakage of such 2-share designs is hardly vanished



(a) A possible realization of a switch box using pass transistors controlled by SRAM cells.



(b) A switch box configuration that potentially leads to coupled shares.

Figure 9: The routing corresponding to two shared wires can be coupled through open pass transistors in the switch box, which could fail the assumption of independent leakage between the shares.

by trial and error, e.g., changing the hardware description of the functions, changing the finite state machine, increasing the noise by internally generating the masks on the target FPGA.

3.2 Case Study 2

Signal routings are realized by active components in FPGAs, i.e., switch boxes. An implementation of a switch box is shown in Figure 9a. Inputs of a switch box can be connected to or disconnected from outputs by setting bits to 1 or 0 in the SRAM cells during the FPGA configuration. The question is whether the routing can affect the leakage in masked hardware, for example if the signals associated to different shares are routed through common switch boxes. This potential issue is illustrated in Figure 9b. If we would route two wires x_1 and x_2 corresponding to a shared value $x = x_1 \oplus x_2$ through the same switch box, it could happen that they have a number of open pass transistors between them. These open pass transistors (depicted in gray) do not create functional contact between the two wires, but are unlikely to be entirely free of leakage current, which can result in the power consumption of share x_1 influencing the power consumption of share x_2 and vice versa.

Note that we do not have access to the proprietary implementation details of the Xilinx FPGA, but modeling the switch box as pass transistors is a reasonable assumption [Wol04]. For this reason it makes sense to design an experiment that can rule out whether or not the effect of routing has a detrimental effect on the security of masked hardware. As an example, the effect which has been reported in [DBG+17] was mainly due to changes in placement and routing. We attempt here to pinpoint and isolate these sources.

In order to examine the hypothesis that "the more open pass transistors are available between shared wires, the higher the leakage" we conduct three experiments. Each experiment has a different number of open pass transistors between shares and we refer to *forks* as a metric for this number. More formally, for two given input wires to a switch box we list all possible output wires they can be routed to and call the cardinality of their intersection the number of forks. In other words, given two input wires to a switch box, a fork is defined as the number of output connections these two wires have in common.

We used the Xilinx Design Language (XDL) [BKT11] and RapidSmith [HNH15] to quantify this amount per set of input wires for a given switch box.

As the reference, we considered the design (of the last case study) with 2 shares employing `iterated_MC`$_1$ and `iterated_MC`$_2$ modules, each of which with all six MC

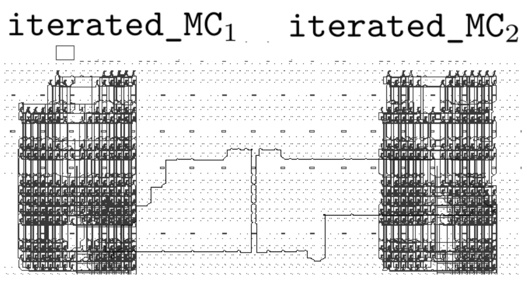


Figure 10: The pipe is placed between the `iterated_MC`$_1$ and `iterated_MC`$_2$ modules.

modules enabled. We have taken the least significant bit (LSB) of the last MC module of both `iterated_MC`$_1$ and `iterated_MC`$_2$, and changed their routings in the new three experimental designs. These two LSBs have been selected since they are one of the last signals in the large combinatorial circuit (of six cascaded MC modules), and hence among the most glitchy signals in every `iterated_MC` module.

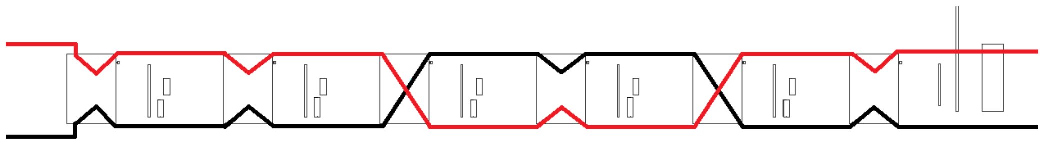
In order to achieve minimal variations over the three experiments w.r.t. the routing, we used a structure we call a *pipe*. This pipe is a series of adjacent switch boxes through which we route the 2 LSB wires and which we place in between the `iterated_MC`$_1$ and `iterated_MC`$_2$ components (Figure 10). The only variation across our three scenarios is found in this pipe, and on 1 of the 2 LSB wires only. In our three scenarios we provided a design with no shared open pass transistors (so-called Pipe_0forks), a second design where we routed the wires through 1 switch box with shared open pass transistors (Pipe_20-00forks), and a third design that routes the wires through 2 switch boxes with shared open pass transistors (Pipe_20-16forks). Note that compared to the previous case study, here we made three more FPGA bitstreams, being different only in the routing of the aforementioned two LSB signals. We now detail the three scenarios in the following.

Scenario 1: Pipe_0forks. A picture of the wires in the pipe is given in Figure 11a. The LSB wires traverse six common switch boxes, and the inputs of each switch box are chosen such that no forks are present in any of them. If our hypothesis is right, we expect this design to leak the least.

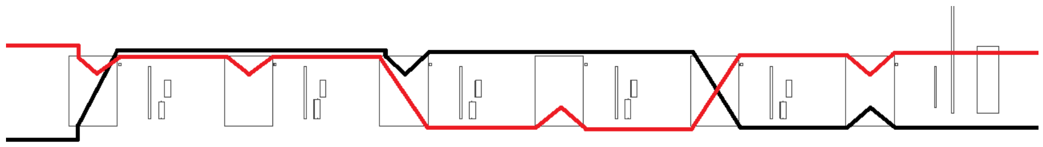
Scenario 2: Pipe_20-00forks. In this scenario (see Figure 11b), the LSB wires now traverse four common switch boxes in order to make the wires show forks in one switch box. We chose the second common switch box (or the third in the pipe) to have 20 forks between the LSB shares. All other switch boxes have no forks. From our hypothesis we expect this design to leak more than the design without forks, i.e., Pipe_0forks.

Scenario 3: Pipe_20-16forks. Figure 11c shows the corresponding picture of the LSB wires in the pipe. The wires again traverse four common switch boxes and are now made to show forks in two switch boxes. We chose the second common switch box to have 20 forks and the fourth one to have 16 forks. The first and the third common switch boxes have no forks between the LSB shares. From the hypothesis we expect this design to leak the most.

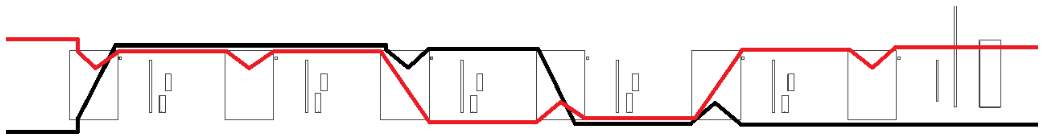
We have collected 200 million traces for each of these designs. The results, depicted in Figure 12, do not show a remarkable difference. The curves, of maximum t-statistics over the number of traces, do not show a recognizable dependency on the associated designs. We can conclude that the effect of routing is almost negligible in this case study, and does not have much effect on the observed leakage.



(a) Pipe_0forks: the LSB wires traverse six common switch boxes, each having 0 fork structures.



(b) Pipe_20-00forks: the LSB wires traverse four common switch boxes, the first having 0 forks, the second having 20 forks, the third having 0 forks and the fourth having 0 forks.



(c) Pipe_20-16forks: the LSB wires traverse four common switch boxes, the first having 0 forks, the second having 20 forks, the third having 0 forks and the fourth having 16 forks.

Figure 11: The pipe configurations for the 3 scenarios. The wire for the LSB of `iterated_MC`$_1$ is fixed (red) whereas the LSB wire of `iterated_MC`$_2$ (black) is varied.

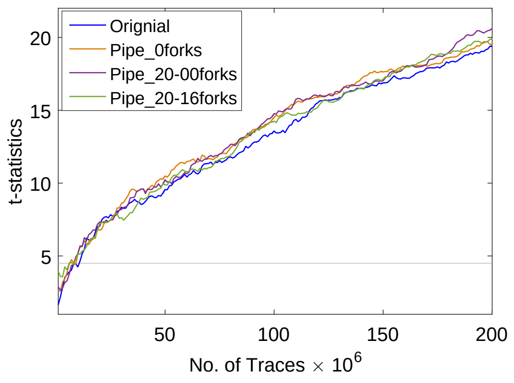
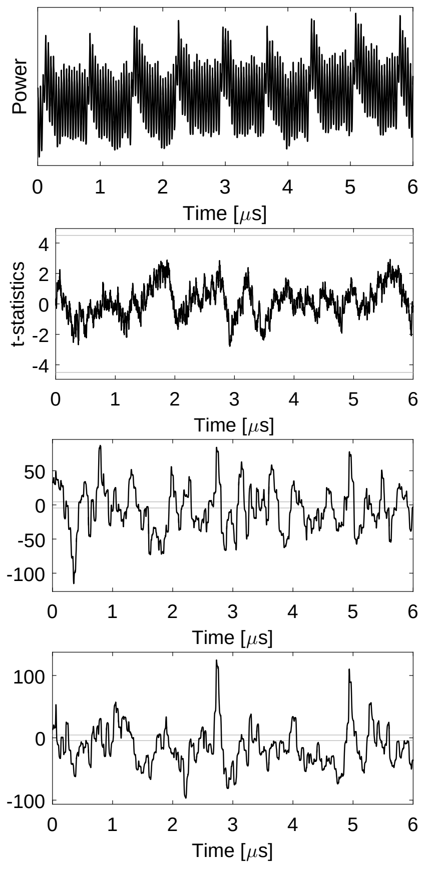


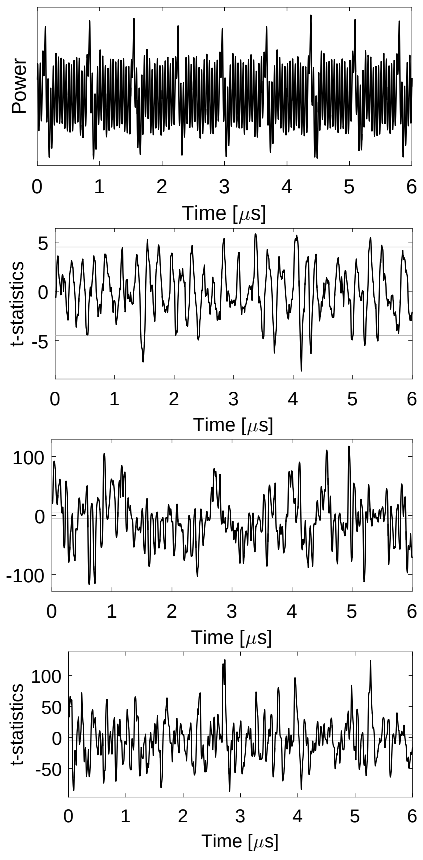
Figure 12: Case study 2: Vdd 1.3 v, shunt 0.0 Ω, 2 shares, `iterated_MC`$_{1,2}$, clock 6 MHz, room temperature, different routings, $max(|t|)$ 1st-order, over the number of traces.

3.3 Case Study 3

In addition to the above-presented case studies, where we intentionally made the leakage of linear functions dominant, we investigated a more realistic scenario. We have taken the TI PRESENT design [PMK+11] which is one of the first applications of uniform TI with 3 shares. It is noteworthy that such a design has been used in many research projects, and on its basis many analyses have been performed and several designs have been developed [EGMP17, MS16, MW15, SMG16, DN17]. It is a nibble-serial implementation of PRESENT-80 encryption function with only one (TI) Sbox instance. At every clock cycle, one nibble is processed while the permutation layer is performed in parallel. Excluding the



(a) Vdd 1.0 v, shunt 1.0 Ω



(b) Vdd 1.3 v, shunt 0.0 Ω

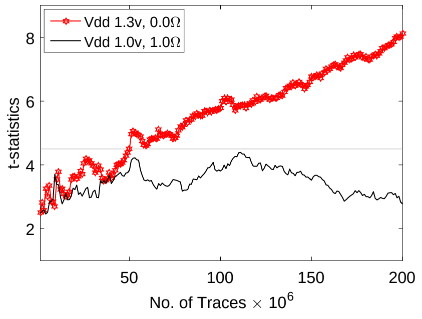


Figure 13: Case study 3: PRESENT TI, 3 shares, clock 12 MHz, room temperature, (top) average of 1000 power traces, (2nd row) 1st-order, (3rd row) 2nd-order and (4th row) 3rd-order t-test results using 200 million traces, (bottom) $max(|t|)$ 1st-order over the number of traces.

initial masking of the plaintext[7], no extra randomness is required (thanks to its underlying uniform TI Sbox). In total, it requires 547 clock cycles to finish an encryption.

We have implemented this design and examined its detectable leakage under two different conditions:

- Vdd supply 1.0 v, shunt resistor 1.0 Ω, and
- Vdd supply 1.3 v, shunt resistor 0.0 Ω.

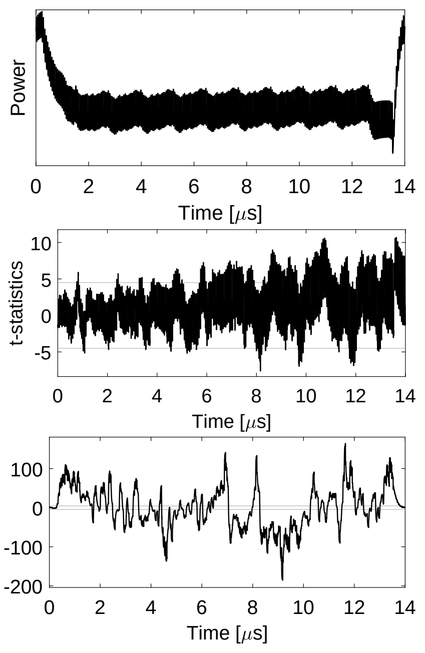
[7]The key is not masked.

Both experiments have been performed at room temperature while the target FPGA was clocked at a frequency of 12 MHz. The results using 200 million traces are shown in Figure 13, and indicate that with a low supply voltage and a high shunt resistor (which is a common configuration) the design shows no first-order leakage, while this picture changes at a high supply voltage and when the shunt resistor is short-circuited. Note that due to its high number of clock cycles per encryption, we covered 8 encryption rounds in this experiment (observable in Figure 13).

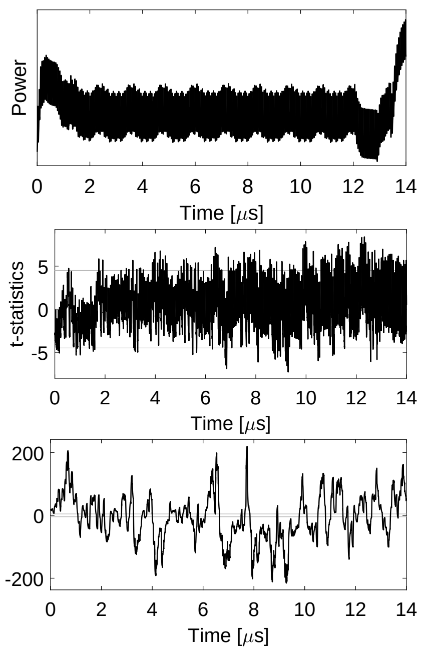
This indeed again confirms our theoretical findings that the shares (and the functions being operated on them) can affect each other through the amount of their energy consumption. It is just a matter of conditions (Vdd, shunt, frequency, temperature, noise) whether such an effect can be seen in practice or not.

3.4 Case Study 4

As the last case study, we have evaluated two publicly-known masked AES implementations with a different number of shares. We have taken the AES DOM implementation [GMK17] which is a nicely developed HDL code, where the desired security order (the number of shares) as a parameter can be easily adjusted (available through the authors GitHub `github.com/hgrosz/aes-dom`). As explained in Section 2.1, it is a $d+1$ masking scheme, i.e., for the desired security order d it makes use of $d+1$ shares (independent of the algebraic degree of the underlying functions, in contrast to TI). We have implemented two cases of this design with 2 shares and 3 shares on our measurement setup, and conducted the same evaluations as before. We further have implemented similarly both 2-share and 3-share versions of another $d+1$ masked AES design, presented in [DRB+16], which is based on the Consolidated Masking Scheme (CMS) [RBN+15]. Since the only difference

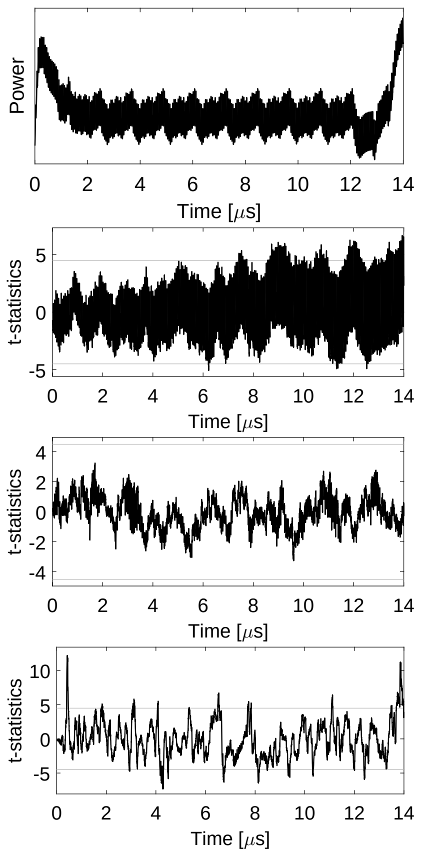


(a) AES DOM, 2 shares

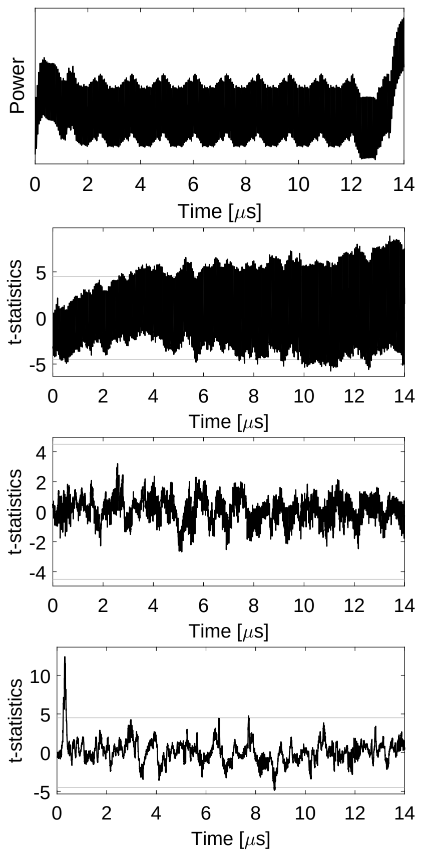


(b) AES CMS, 2 shares

Figure 14: Case study 4: AES, DOM & CMS, 2 shares, Vdd 1.2 v, shunt 1.0 Ω, clock 24 MHz, room temperature, (top) average of 1000 power traces, (middle) 1st-order and (bottom) 2nd-order t-test results using 100 million traces.



(a) AES DOM, 3 shares



(b) AES CMS, 3 shares

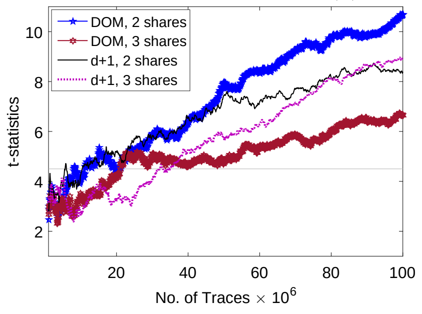


Figure 15: Case study 4: AES, DOM & CMS, 3 shares, Vdd 1.2 v, shunt 1.0 Ω, clock 24 MHz, room temperature, (top) average of 1000 power traces, (2nd row) 1st-order, (3rd row) 2nd-order and (4th row) 3rd-order t-test results using 100 million traces, (bottom) $max(|t|)$ 1st-order over the number of traces.

between these two designs (DOM and CMS) is in their masked AES Sbox, we have taken the parametric DOM AES HDL code and only replaced the masked Sbox with that of CMS. Since the CMS Sbox design [DRB+16] requires two less stage registers compared to the that in DOM [GMK17], we added two dummy register stages at the Sbox output of CMS to keep the rest of the DOM design (particularly the state machine and the controlling signals).

As given in Section 2.1, such masking schemes – in contrast to uniform TI – need fresh randomness during the operation of non-linear functions. To this end, to generate every single random bit per clock cycle we made use of a dedicated 64-bit LFSR with feedback function $x^{64} + x^{63} + x^{61} + x^{60} + 1$. Each LFSR instance has been initialized with a random value enabling each to generate a stream with cycle $2^{64} - 1$. The required number of random bits per design (per clock cycle) is 38 (2-share DOM), 72 (2-share CMS), 84 (3-share DOM), and 162 (3-share CMS). During the measurements, each design was clocked by a 24 MHz oscillator, and we collected 100 million traces covering the entire encryption process with Vdd 1.2 v and shunt resistor 1.0 Ω in room temperature[8]. The results are shown in Figure 14 and Figure 15 indicating detectable first-order leakage in all cases.

4 Conclusions

Hardware platforms offer high throughput due to their parallelism feature. At the same time, when masking schemes are realized in hardware, their security proof relies on a principle assumption that the power consumption associated to different shares are independent of each other. We illustrated that it is an optimistic assumption, and in reality it can be violated under certain conditions, e.g., high supply voltage, high temperature, high clock frequency, low noise, etc. Supported by extensive practical experiments, we showed that when such an assumption does not completely hold, the power consumption of masked hardware becomes – in average – dependent on the secret intermediates, i.e., shows first-order leakage.

To identify the array of sources that influence the security order, our experiments used a high number of traces measured in a low-noise setting, and we leveraged the speed of leakage detection tests as opposed to more time consuming and computationally heavy key retrieval experiments. We note that a real-world attacker would not have access to this many clean traces and would not exploit first-order leakage if there is stronger, more informative leakage present in the higher order. We, however, believe that our work can supply an attacker with an advantage, of which the specifics remain to be uncovered in future work. On the other hand, our results indicate that SCA practical evaluations of masked implementations by means of t-test leakage assessment can show detectable leakage independent of the soundness of the underlying masking scheme and the correctness of the implementation.

All the experiments we showed here were dedicated to FPGA-based designs, particularly those inspecting the effect of routing. Hence, we cannot directly conclude that our findings definitely hold in any ASIC implementation. However, since supply voltage fluctuation is a known challenge in the areas of VLSI design originating from the PDN and the VRM, we expect our findings to be observable in ASIC prototypes as well, that is for sure among our future works. With respect to this topic, we should emphasize that FPGA designs usually consume more energy compared to their ASIC counterpart. We also showed that the higher power consumption, the less measurements are required to observe the first-order leakage due to violating the aforementioned assumption. Therefore, we predict this issue to be harder to observe in ASIC implementations with modern technology nodes.

Since the voltage fluctuations cannot easily be hindered, the solutions for hardware masking include:

- Sequential operations instead of parallelism by defining **temporal** non-completeness. That is, for d^{th}-order security, it should not be allowed to process more than d shares at every point in time (every clock cycle). It obviously reduces the throughput and avoids making use of the full parallelism feature of hardware, and might be still violated at high clock frequencies.

[8]Such settings were arbitrarily selected.

- Embedded voltage regulators (VRM) inside the chip, which is already being done commonly in commercial smart cards. In this case, an interesting question is whether EM signals also exhibit such an issue.
- Support by the chip having separate Vdd lines to supply functions associated to each share independently. This may solve the problem when the functions are linear and the masking scheme Boolean. Otherwise, it is not clear how to supply the functions which operate on a set of (non-complete) shares.

Acknowledgments

This work is supported in part by NIST with the research grant 60NANB15D346 and the German Research Foundation (DFG) through the project NaSCA (Nano-Scale Side-Channel Analysis). Thomas De Cnudde is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

[ASM07] Karim Arabi, Resve A. Saleh, and Xiongfei Meng. Power Supply Noise in SoCs: Metrics, Management, and Measurement. *IEEE Design & Test of Computers*, 24(3):236–244, 2007.

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness Complexity of Private Circuits for Multiplication. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648. Springer, 2016.

[BGI+18] Roderick Bloem, Hannes Gross, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal Verification of Masked Hardware Implementations in the Presence of Glitches. In *Advances in Cryptology - EUROCRYPT 2018*, Lecture Notes in Computer Science, 2018. https://eprint.iacr.org/2017/897.

[BGN+14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-Order Threshold Implementations. In *ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[BGN+15] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Trade-Offs for Threshold Implementations Illustrated on AES. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.

[Bil15] Begül Bilgin. *Threshold Implementations As Countermeasure Against Higher-Order Differential Power Analysis.* PhD thesis, KU Leuven (Belgium) and University of Twente (The Netherlands), 2015.

[BKT11] Christian Beckhoff, Dirk Koch, and Jim Tørresen. The Xilinx Design Language (XDL): Tutorial and use cases. In *ReCoSoC*, pages 1–8. IEEE, 2011.

[BPG18] Florian Bache, Christina Plump, and Tim Güneysu. Confident Leakage Detection - A Side-Channel Evaluation Framework Based on Confidence Intervals. In *DATE 2018*, 2018. to appear.

[CB08] D. Canright and Lejla Batina. A Very Compact "Perfectly Masked" S-Box for AES. In *ACNS 2008*, volume 5037 of *Lecture Notes in Computer Science*, pages 446–459, 2008.

[CDG+13] Jeremy Cooper, Elke Demulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test Vector Leakage Assessment (TVLA) Methodology in Practice. International Cryptographic Module Conference, 2013.

[CFE16] Cong Chen, Mohammad Farmani, and Thomas Eisenbarth. A Tale of Two Shares: Why Two-Share Threshold Implementation Seems Worthwhile - and Why It Is Not. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 819–843, 2016.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[Cor] John D. Corbett. The Xilinx Isolation Design Flow for Fault-Tolerant Systems. `https://www.xilinx.com/support/documentation/white_papers/wp412_IDF_for_Fault_Tolerant_Sys.pdf`. WP412 (v1.1) October 16, 2013.

[DBG+17] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does Coupling Affect the Security of Masked Implementations? In *COSADE 2017*, volume 10348 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2017.

[DBR+16] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-Order Threshold Implementation of the AES S-Box. In *CARDIS 2015*, volume 9514 of *Lecture Notes in Computer Science*, pages 259–272. Springer, 2016.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device. In *EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

[DN17] Thomas De Cnudde and Svetla Nikova. Securing the PRESENT Block Cipher Against Combined Side-Channel Analysis and Fault Attacks. *IEEE Trans. VLSI Syst.*, 25(12):3291–3301, 2017.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[DRB+16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 Shares in Hardware. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 194–212. Springer, 2016.

[DWB15] Shidhartha Das, Paul N. Whatmough, and David M. Bull. Modeling and characterization of the system-level Power Delivery Network for a dual-core ARM Cortex-A57 cluster in 28nm CMOS. In *ISLPED 2015*, pages 146–151. IEEE, 2015.

[DWS10] Geir Olav Dyrkolbotn, Knut Wold, and Einar Snekkenes. Security Implications of Crosstalk in Switching CMOS Gates. In *ISC*, volume 6531 of *Lecture Notes in Computer Science*, pages 269–275. Springer, 2010.

[DWS12] Geir Olav Dyrkolbotn, Knut Wold, and Einar Snekkenes. Layout Dependent Phenomena A New Side-channel Power Model. *JCP*, 7(4):827–837, 2012.

[EGMP17] Maik Ender, Samaneh Ghandali, Amir Moradi, and Christof Paar. The First Thorough Side-Channel Hardware Trojan. In *ASIACRYPT 2017*, volume 10624 of *Lecture Notes in Computer Science*, pages 755–780. Springer, 2017.

[GE16] Ilias Giechaskiel and Ken Eguro. Information Leakage Between FPGA Long Wires. *arXiv preprint arXiv:1611.08882*, 2016.

[GJJR11] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. A testing methodology for side channel resistance validation. In *NIST non-invasive attack testing workshop*, 2011.

[GM17] Hannes Groß and Stefan Mangard. Reconciling d+1 Masking in Hardware and Software. In *CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 115–136. Springer, 2017.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. *IACR Cryptology ePrint Archive*, 2016:486, 2016.

[GMK17] Hannes Groß, Stefan Mangard, and Thomas Korak. An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order. In *CT-RSA*, volume 10159 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2017.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[GOJ$^+$07] Meeta Sharma Gupta, Jarod L. Oatley, Russ Joseph, Gu-Yeon Wei, and David M. Brooks. Understanding voltage variations in chip multiprocessors using a distributed power-delivery network. In *DATE 2007*, pages 624–629. EDA Consortium, San Jose, CA, USA, 2007.

[GOKT16] Dennis R. E. Gnad, Fabian Oboril, Saman Kiamehr, and Mehdi Baradaran Tahoori. Analysis of transient voltage fluctuations in FPGAs. In *FPT 2016*, pages 12–19. IEEE, 2016.

[GP99] Louis Goubin and Jacques Patarin. DES and Differential Power Analysis (The "Duplication" Method). In *CHES*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How Fast Can Higher-Order Masking Be in Software? In *EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597, 2017.

[HBW$^+$07] Ted Huffmire, Brett Brotherton, Gang Wang, Timothy Sherwood, Ryan Kastner, Timothy E. Levin, Thuy D. Nguyen, and Cynthia E. Irvine. Moats and Drawbridges: An Isolation Primitive for Reconfigurable Hardware Based Systems. In *IEEE S&P*, pages 281–295. IEEE Computer Society, 2007.

[HNH15] Travis Haroldsen, Brent E. Nelson, and Brad L. Hutchings. RapidSmith 2: A Framework for BEL-level CAD Exploration on Xilinx FPGAs. In *FPGA*, pages 66–69. ACM, 2015.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[Lar99] P. Larsson. Power supply noise in future IC's: a crystal ball reading. In *IEEE Custom Integrated Circuits Conference*, pages 467–474, 1999.

[LSB11] Z. Liu, S. Sun, and P. Boyle. FPGA core PDN design optimization. In *IEEE International Symposium on Electromagnetic Compatibility*, pages 411–416, 2011.

[McN] Steve McNeil. Developing Secure Designs with the Spartan-6 Family Using the Isolation Design Flow. https://www.xilinx.com/support/documentation/application_notes/xapp1145_S6Secure_Designs.pdf. XAPP1145 (v1.2) September 8, 2015.

[MF04] Andrey V. Mezhiba and Eby G. Friedman. Scaling trends of on-chip power distribution noise. *IEEE Trans. VLSI Syst.*, 12(4):386–394, 2004.

[MME10] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2010.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully Attacking Masked AES Hardware Implementations. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[MS16] Amir Moradi and François-Xavier Standaert. Moments-Correlating DPA. In *ACM Workshop on Theory of Implementation Security, TIS@CCS*, pages 5–15. ACM, 2016.

[MW15] Amir Moradi and Alexander Wild. Assessment of Hiding the Higher-Order Leakages in Hardware - What Are the Achievements Versus Overheads? In *CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2015.

[NKSF17] Kashif Nawaz, Dinal Kamel, François-Xavier Standaert, and Denis Flandre. Scaling Trends for Dual-Rail Logic Styles Against Side-Channel Attacks: A Case-Study. In *COSADE*, volume 10348 of *Lecture Notes in Computer Science*, pages 19–33. Springer, 2017.

[NRS11] 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.

[OMPR05] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In *FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.

[PMK$^+$11] 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. Cryptology*, 24(2):322–345, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks: A Formal Security Proof. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[pri] Private discussions with various researchers at RUB, KUL, IAIK.

[PS16] Santos Merino Del Pozo and François-Xavier Standaert. A note on the security of threshold implementations with $d+1$ input shares. Cryptology ePrint Archive, Report 2016/420, 2016. https://eprint.iacr.org/2016/420.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[RBN$^+$15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In *CRYPTO (1)*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[Rep16] Oscar Reparaz. Detecting Flawed Masking Schemes with Leakage Detection Tests. In *FSE*, volume 9783 of *Lecture Notes in Computer Science*, pages 204–222. Springer, 2016.

[RPD$^+$18] Chethan Ramesh, Shivukumar B. Patil, Siva Nishok Dhanuskodi, George Provelengios, Sebastien Pillement, Daniel Holcomb, and Russell Tessier. FPGA Side Channel Attacks without Physical Access. In *Symposium on Field-Programmable Custom Computing Machines, FCCM 2018*. IEEE Computer Society, 2018.

[sak] Side-channel AttacK User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html.

[SGMT18] Falk Schellenberg, Dennis R.E. Gnad, Amir Moradi, and Mehdi B. Tahoori. An Inside Job: Remote Power Analysis Attacks on FPGAs. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2018*. IEEE, 2018.

[SM15] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.

[SMG16] Pascal Sasdrich, Amir Moradi, and Tim Güneysu. Affine Equivalence and Its Application to Tightening Threshold Implementations. In *SAC 2015*, volume 9566 of *Lecture Notes in Computer Science*, pages 263–276. Springer, 2016.

[TAV02] K. Tiri, M. Akmal, and I. Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Proceedings of the 28th European Solid-State Circuits Conference*, pages 403–406, Sept 2002.

[Tri03] Elena Trichina. Combinational Logic Design for AES SubByte Transformation on Masked Data. *IACR Cryptology ePrint Archive*, 2003:236, 2003.

[TV04] Kris Tiri and Ingrid Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA implementation. In *DATE*, pages 246–251. IEEE Computer Society, 2004.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note. In *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.

[Wol04] W. Wolf. *FPGA-Based System Design*. Prentice Hall modern semiconductor design series. Pearson Education, 2004.

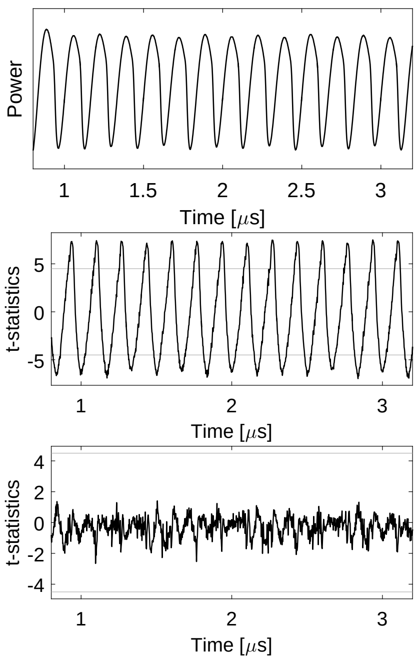
[Xil] Xilinx. Spartan-6 FPGA Data Sheet: DC and Switching Characteristics. `https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf`. DS162 (v3.1.1) January 30, 2015.

[ZED+15] Loïc Zussa, Ingrid Exurville, Jean-Max Dutertre, Jean-Baptiste Rigaud, Bruno Robisson, Assia Tria, and Jessy Clédière. Evidence of an information leakage between logically independent blocks. In *CS2@HiPEAC*, pages 25–30. ACM, 2015.

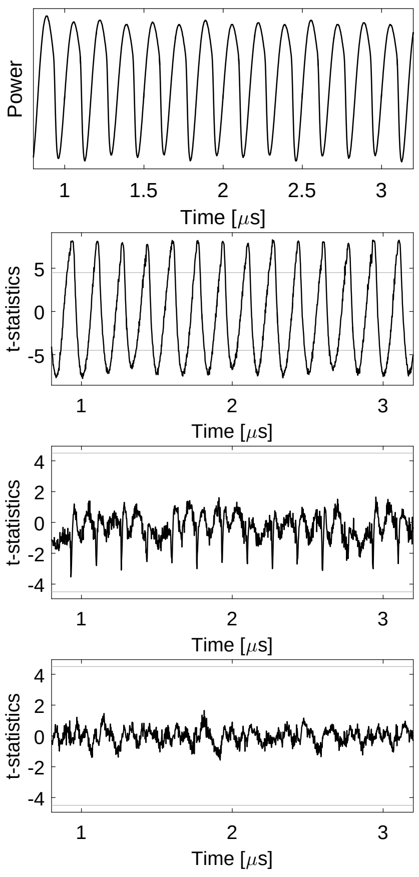
[ZS18] Mark Zhao and G. Edward Suh. FPGA-Based Remote Power Side-Channel Attacks. In *2018 IEEE Symposium on Security and Privacy, SP 2018*, pages 839–854. IEEE Computer Society, 2018.

[ZWM+14] Runjie Zhang, Ke Wang, Brett H. Meyer, Mircea R. Stan, and Kevin Skadron. Architecture implications of pads as a scarce resource. In *ISCA 2014*, pages 373–384. IEEE Computer Society, 2014.

A Additional Figures



(a) 3 shares



(b) 4 shares

Figure 16: Case study 1: Vdd 1.3 v, shunt 0.0 Ω, 6 *MC* modules, clock 6 MHz, room temperature, (top) sample power trace, (2nd row) 1st-order *t*-test results, (3rd row) 2nd-order *t*-test results, (bottom) 3rd-order *t*-test results using 200 million traces.