Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model

Sebastian Faust[1], Vincent Grosso[2], Santos Merino Del Pozo[3,4], Clara Paglialonga[1] and François-Xavier Standaert[3]

[1] Technische Universität Darmstadt, Darmstadt, Germany.
[2] Radboud University Nijmegen, Digital Security Group, The Netherlands.
[3] Université catholique de Louvain, ICTEAM/ELEN/Crypto Group, Belgium.
[4] DarkMatter LLC, Abu Dhabi, United Arab Emirates.

Abstract. Composability and robustness against physical defaults (e.g., glitches) are two highly desirable properties for secure implementations of masking schemes. While tools exist to guarantee them separately, no current formalism enables their joint investigation. In this paper, we solve this issue by introducing a new model, the robust probing model, that is naturally suited to capture the combination of these properties. We first motivate this formalism by analyzing the excellent robustness and low randomness requirements of first-order threshold implementations, and highlighting the difficulty to extend them to higher orders. Next, and most importantly, we use our theory to design and prove the first higher-order secure, robust and composable multiplication gadgets. While admittedly inspired by existing approaches to masking (e.g., Ishai-Sahai-Wagner-like, threshold, domain-oriented), these gadgets exhibit subtle implementation differences with these state-of-the-art solutions (none of which being provably composable and robust). Hence, our results illustrate how sound theoretical models can guide practically-relevant implementations.

Keywords: Side-channel analysis, security proofs, physical defaults, composability.

1 Introduction

State-of-the-art. Protecting hardware and software implementations against side-channel attacks is an important challenge in cryptographic engineering. The masking countermeasure is among the most popular solutions for this purpose, due to the good understanding of its security requirements [CJRR99, ISW03, PR13, DDF14, DFS15]. Intuitively, masked implementations can be viewed as implementations performing computations on secret-shared data. Under the fundamental assumptions that (*i*) each leakage (sample) depends on a limited number of shares (ideally one)[1], and (*ii*) the leakages of the shares are sufficiently noisy, masking guarantees that the measurement complexity of any side-channel attack grows exponentially in the number of shares. Since the implementation cost of a masking scheme only grows (roughly) quadratically in the number of shares, it therefore provides a theoretically sound principle to prevent side-channel attacks for any cryptographic primitive.

Unfortunately, ensuring these (independence and noise) requirements is non-trivial:

First, a lack of composability (typically caused by an insufficient refreshing of the shares) can reduce the security order in the *probing model* of Ishai et al. [ISW03]. Such

[1] Or is a linear combination of the shares' leakages in the case of parallel implementations [BDF+17].

IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925, Vol. 2018, No. 3, pp. 89–120
DOI:10.13154/tches.v2018.i3.89-120

a flaw is illustrated by the FSE 2013 attack by Coron et al. [CPRR13]. Considering a masking scheme with d shares, it means that a combination of $d' < d$ shares is sufficient to extract sensitive information. The natural solution to avoid it is to use composable (e.g., SNI [BBD+16]) gadgets, and/or to test the (hardware or software) description codes of the implementations (i.e., all the operations manipulating the shares) thanks to formal methods [BBD+15].

Second, even if a combination of d shares is required to extract sensitive information (i.e., if probing security is guaranteed), physical defaults can reduce the security order in the *bounded moment model* of Barthe et al. [BDF+17]. It then means that the lowest key-dependent statistical moment of the leakage distribution is lower than the optimal d. Concretely, such reductions happen because the leakage function recombines the shares to some extent. Typical examples of physical defaults include glitches (i.e., combinatorial recombinations) [MPG05, MPO05], transition-based leakages (i.e., memory recombinations) [CGP+12, BGG+14] and potentially couplings (i.e., routing recombinations) [CBG+17].[2] In practice, the security order in the bounded moment model can be determined using the TVLA methodology [CMG+, GJJR11] or variations thereof [MOBW13, SM16, DS16]: we refer to [Sta17] for a recent discussion.

Third, and even if security is guaranteed in the probing and bounded moment models, the noise condition may be challenging to achieve too, possibly leading to an insufficient security in the (most practically relevant) *noisy leakage model* of Prouff and Rivain [PR13]. Concretely, this noise condition depends on two main parameters. On the one hand, the physical noise of the operations exploited (e.g., the operations that depend on an enumerable part of the key if one considers standard "divide-and-conquer" side-channel attacks), which is generally assumed more or less equal for all the operations. On the other hand, the number of operations exploited. In this respect, two recent works showed how an adversary can efficiently exploit these multiple leakages thanks to multivariate (aka horizontal) attacks [BCPZ16, GS18]. The core intuition behind these works is that as the number of shares in a masking schemes increases, the number of exploitable operations does too, implying that larger physical noise is necessary for masking to deliver security.[3] In practice, the global noise level necessary for secure masking (which roughly corresponds to the ratio between the physical noise and the number of exploitable operations in the implementation) can be determined/estimated thanks to an information theoretic analysis [GS18].

Motivation & goals. Based on this state-of-the-art, it appears that one key remaining challenge in the design of secure masking schemes is to minimize the amount of (bad) surprises at implementation time. In this respect, our starting observation is that existing models provide a principled path for this purpose. That is, and as illustrated in Figure 1, this challenge can be split in three parts: first obtain security in the (abstract) probing model (which guarantees composability), second ensure that concrete leakages do not (completely) recombine the shares in the bounded moment model, finally evaluate the concrete security level in the noisy leakage model. Since the evaluation of the noise condition is discussed in [GS18], we focus on the first conditions: how to guarantee security in the probing model (i.e., to refresh sufficiently) and bounded moment model (i.e., to mitigate physical defaults), and what are the interactions between these requirements?

Admittedly, specific answers to solve these two issues separately already exist. As previously mentioned, security in the probing model can be ensured thanks to composable gadgets and/or formal methods. As for security in the bounded moment model, it is well known that certain algorithmic features offer excellent ways to mitigate the shares' recombinations. The most famous example is the case of threshold implementations [NRS11],

[2] Although their exploitability in concrete attacks is not yet fully demonstrated.

[3] Masking based on LUTs make this issue more critical because of larger performance overheads [TWO13, BGNT15].

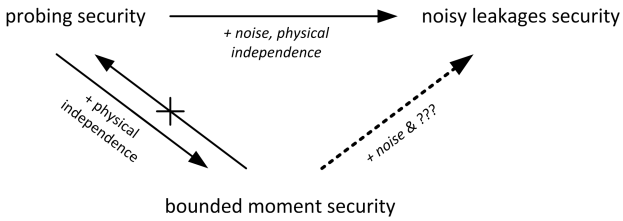

Figure 1: Reductions between leakage security models (borrowed from [BDF+17]).

where a non-completeness property is used to prevent that any combinatorial logic has access to all the shares of an encoded sensitive variable. The lazy engineering solution in [BGG+14] is another example where increasing the number of shares allows ruling out their full (memory) recombination. But open questions remain regarding whether these two threats (and possibly couplings) can be captured jointly? Also, the generalization of threshold implementations to higher-orders in [BGN+14] has been shown to suffer from refreshing issues in [Rep15], and the heuristic solution in [RBN+15a] does not yet provide a systematic way to evaluate randomness requirements – something probing security is very good at. So in general, a model allowing to capture both composability issues and to mitigate physical defaults would be very handy. These examples also question why first-order threshold implementations do not suffer from refreshing issues (despite low randomness requirements [PMK+11])?

Our contribution. We start with the case of first-order threshold implementations and show that their low randomness requirements can be explained thanks to (a slight variation of) the notion of Strong Non Interference (SNI) introduced in [BBD+16]. We next discuss the "number of shares vs. cycle count" tradeoff for such implementations. For this purpose, we observe that the correctness property of threshold implementations is in fact not needed for their intermediate results (i.e., we only want the final result to be correct). It allows us to exhibit examples of 4-bit S-boxes that globally match the definition of first-order threshold implementations in two shares and two cycles (a similar example is given in [CFE16] for the Simon S-box). We then exploit this observation in order to (slightly) refine the exhaustive decomposition in [BNN+12, RBN+15b] for certain S-boxes. We conclude by discussing the additional challenges raised by higher-order glitch-free implementations, and use them in order to motivate the need of a new model. We follow with our main contribution, which is to provide a formal tool to analyze such higher-order masked implementations. For this purpose, we introduce a new *robust probing model* which tweaks the original probing model in order to capture a wide class of physical defaults and can naturally be combined with existing notions of composability. Thanks to this model, we first discuss (and sometimes conjecture) simple propositions regarding the combination of physical defaults. We then study concrete constructions of masked and threshold implementations. One important conclusion of these investigations is that a 2-cycle implementation of Ishai et al.'s (slightly tweaked) multiplication algorithm [ISW03] (or the parallel multiplication algorithm in [BDF+17]) offers good robustness against physical defaults (which we confirm with FPGA experiments), while also offering good composability by design (as guaranteed by theoretical analysis). We note that besides the interesting consolidating nature of the designs and proofs we provide, a recent follow-up work [MMSS18] showed that the lack of probing security proofs in previous hardware-oriented glitch-resistant masking schemes (e.g., [RBN+15a, CRB+16, GMK16, GMK17, GM17]) actually leads to probing security flaws as the number of shares in these schemes increases. It also shows the necessity of the robust probing model by exhibiting that satisfying glitch-resistance (thanks to the non-completeness property of threshold implementations) and composability (thanks to SNI) separately is not enough to be glitch-robust and composable.

2 Background

2.1 Circuit model

For our circuit model, we borrow the solution of [ISW03] and represent a deterministic circuit C as a directed acyclic graph whose vertices are *combinatorial gates* and edges are wires carrying elements from a finite field $\mathbb{F}$. The simplest case is when $\mathbb{F}$ is the binary field so that wires carry bits and gates are Boolean operations AND and XOR. Yet, the addition and multiplication algorithms of secure masking schemes (e.g., described in Section 2.3) can run in larger fields $\mathbb{F}_{2^n}$: we then consider arithmetic circuits and gates rather than Boolean ones. In all cases, we denote field additions (resp., multiplications) by $\oplus$ (resp., $\odot$). Since masking gadgets are randomized circuits, the model in [ISW03] augments the previous deterministic circuits with *random gates* with fan-in 0: they produce a uniformly random element of the considered field. Eventually, robust masking requires circuits to be stateful (e.g., threshold implementations cannot maintain the non-completeness property discussed in the next sections otherwise [NRS11]). For this purpose, we use *memory gates* which, on every invocation of the circuit, output the previous input to the gate and stores the current input for the next invocation. We note that these abstractions can be reasonably and efficiently instantiated in practice, using true- or pseudo-random number generators for the random gates and registers (synchronized by a clock signal) for the memory gates.

2.2 Probing security and (Strong) Non Interference

In order to formalize the security of a masking scheme, Ishai et al. introduced in [ISW03] the *q-probing model*, in which an attacker is allowed to read up to q intermediate wires of a target circuit. In order to protect a circuit in this model, every sensitive value k is split into at least $q+1$ values, called *shares*, such that their sum gives k. The security of a randomized circuit modeled as in the previous paragraph (which transforms a randomly encoded input into a randomly encoded output) can then be expressed in various ways. Since our following discussions will consider both composable and non-composable gadgets, we next provide three different definitions. The first one, which is limited to non-composable security, was given by Rivain and Prouff in a CHES 2010 work that initiated the use of the probing model in order to analyze the security of concrete masking schemes:

Definition 1 ($q-$probing security [ISW03, RP10])**.** A circuit gadget G is *$q-$probing secure* iff every q-tuple of its intermediate variables is independent of any sensitive variable.

We sometimes refer to this security notion as security at order q in the probing model. In the case of block ciphers, sensitive variables typically correspond to partial computation results depending on the plaintext and key [CPR07]. Security in the probing model can also be expressed with the existence of a *simulator*, which can mimic the adversary's view using only black-box access to G, i.e., without the knowledge of any internal wire but only q shares of each secret input. We use the definition of Barthe et al. for this purpose:

Definition 2 ($q-$NI [BBD+16])**.** A gadget G is *$q-$Non Interfering* iff for any set of q_1 probes on its intermediate values and every set of q_2 probes on its output shares with $q_1 + q_2 \leq q$, the totality of the probes can be simulated with $q_1 + q_2$ shares of each input.

In other words, a circuit gadget is called NI if no *distinguisher* is able to tell apart the adversary's view from the simulation. In this respect, one important technical clarification is that in the definition of Barthe et al., the distinguisher can access the joint distribution of the (simulated) probes and input shares. As a result, NI is a stronger notion than the previous probing security. Eventually, when gadgets are composed for producing a more complex circuit, it is needed to take into account that using an output of a gadget as input of another one can give additional information to the attacker. In this case, the

definition of q-NI is not sufficient anymore to ensure global security of the circuit. A stronger property, called $q-$*Strong Non Interference* (or $q-$SNI), was also introduced by Barthe et al. in order to capture this requirement and is recalled in the following:

Definition 3 ($q-$SNI [BBD+16])**.** A gadget G is $q-$ *Strong Non Interfering* iff for any set of q_1 probes on its intermediate values and every set of q_2 probes on its output shares with $q_1 + q_2 \leq q$, the totality of the probes can be simulated with q_1 shares of each input.

Intuitively, this property does not only require that the adversary's view can be simulated with q secret shares as for $q-$NI security, but also that the number of shares needed for the simulation to succeed is independent from the number of output wires that are probed. How to use/combine NI and SNI gadgets in order to build secure circuits based on simple and sound composition rules will be discussed/recalled in Section 8.

2.3 The ISW multiplication algorithm

The first probing secure multiplication algorithm was introduced in the seminal work of Ishai et al. [ISW03], and has been proved to be $q-$SNI in [BBD+16].[4] As shown in [RP10], such an algorithm generalizes to larger fields, given that the refreshing is adapted to make it composable as proposed in [CPRR13]. In the following, we will use the slight variation depicted in Algorithm 1. The only difference is in the way we organize the intermediate results, which is better suited to prevent physical defaults (see the discussion in Section 5.2).

Algorithm 1 Modified ISW multiplication algorithm with $d \geq 2$ shares.

Input: shares $(a_i)_{1\leq i\leq d}$ and $(b_i)_{1\leq i\leq d}$, such that $\bigoplus_i a_i = a$ and $\bigoplus_i b_i = b$.
Output: shares $(c_i)_{1\leq i\leq d}$, such that $\bigoplus_i c_i = a \odot b$.

```
for i = 1 to d do
  for j = i + 1 to d do
    r_{i,j} ←$ F_{2^n};
    u_{i,j} ← r_{i,j} ⊕ a_i ⊙ b_j;
    u_{j,i} ← r_{i,j} ⊕ a_j ⊙ b_i;
  end for
end for
for i = 1 to d do
  c_i ← a_i ⊙ b_i ⊕ ⨁_{j=1,j≠i}^{d} u_{i,j};
end for
```

3 The special case of 1st-order TIs

From the performance point-of-view, one important feature of the previous multiplication algorithm is that it requires fresh randomness for every multiplication in the circuit to protect. Yet, the Threshold Implementations' (TIs) literature shows that it is sometimes possible to protect a full block cipher execution with very minimum randomness (i.e., the block size, typically) [PMK+11]. This suggests that such implementations benefit from some sort of composability.[5] In this section, we investigate this interesting property of 1st-order TIs. For this purpose, we first recall that TIs are a type of masking scheme aimed at counteracting power (or electromagnetic) analysis attacks in the presence of glitches. In

[4] Note that the original ISW multiplication algorithm was using $2q + 1$ shares. In this paper, we focus on the variant with $q + 1$ shares that is more interesting from the performance viewpoint.

[5] Which, as will be clarified in the rest of this section, cannot be strictly defined as composability.

the 1st-order case, a TI takes a function $\mathsf{f}(x)$ with a uniform sharing of the input x, next denoted as $\boldsymbol{x} = (x_1, \ldots, x_m)$ such that $x = x_1 \oplus \cdots \oplus x_m$. The function $\mathsf{f}(.)$ is then shared in a vector of component functions $(\mathsf{f}_1, \ldots, \mathsf{f}_m)$ which needs to satisfy:

1. **Correctness**: $y = \mathsf{f}(x) = \bigoplus_{i=1}^{m} \mathsf{f}_i(\boldsymbol{x})$.
2. **Non-Completeness**: any component functions f_i of f must be independent of at least one input share. (Or any combination of up to q component functions f_i of f must be independent of at least one input share in the higher-order case [BGN+14]).
3. **Uniformity**: denoting the vector of the output shares as $\boldsymbol{c} = (\mathsf{f}_1(\boldsymbol{x}), \ldots, \mathsf{f}_m(\boldsymbol{x}))$, the probability $\Pr(\vec{C} = \vec{c} | c = \bigoplus_{i=1}^{m} c_i)$ must be a fixed constant $\forall \vec{c}$.

Note that the non-completeness property is not related to the refreshing (composability) issues that we discuss in this section and rather relates to the modeling and analysis of physical defaults that will be carefully discussed in Section 4.1 and following.

3.1 Pseudo−NI and pseudo−SNI security

Let us now consider the 3×1-bit function $\mathsf{f}(x, y, z) = (x \odot y) \oplus z$ which is at the core of many efficient S-box decompositions for TIs. In this case, it is easy to find a 1st-order TI with only 3 shares, given by the following set of equations:

$$\begin{aligned} c_1 &= (x_2 \odot y_2) \oplus (x_2 \odot y_3) \oplus (x_3 \odot y_2) \oplus z_2, \\ c_2 &= (x_3 \odot y_3) \oplus (x_3 \odot y_1) \oplus (x_1 \odot y_3) \oplus z_3, \\ c_3 &= (x_1 \odot y_1) \oplus (x_1 \odot y_2) \oplus (x_2 \odot y_1) \oplus z_1. \end{aligned} \tag{1}$$

Interestingly, the addition of the third variable z to the non-linear part $x \odot y$ guarantees the uniformity of the outputs. However, even if this gadget is "ideally implemented" in a single clock cycle (i.e., intermediate computations such as $x_2 \odot y_2$, $x_2 \odot y_3$, ... do not leak and no probes are allowed on them), it is not 1−SNI nor even 1−NI.[6] For example, a single probe on c_1 (meaning $q_1 = 0$ and $q_2 = 1$) cannot be simulated with a single share per input. This is because the computation of c_1 requires two shares of x and two shares of y, and there is no internal randomness in the gadget that can help the simulation. By contrast, this gadget is 1−probing secure (since the c_i's are independent of x, y and z). So the standard notions of NI and SNI security cannot directly capture the low randomness requirements of 1st-order TIs. This is in fact natural since the main idea behind 1st-order TIs is to leverage the uniformity of the shares. Therefore, and in order to exhibit an intuitive connection between TIs and composable masking schemes, we propose the following (slight) variation of existing NI/SNI definitions:

Definition 4 (Pseudo−randomized gadgets)**.** The pseudo−randomization G' of a gadget G is the gadget G modified such that any input share coming from a uniform encoding and appearing only once and as a monomial of degree one in the algebraic circuit description of the gadget G is removed from the gadget inputs and replaced by internal uniform randomness in G'. We denote these monomials as pseudo−randomized monomials.

Definition 5 (Pseudo−q−NI/SNI)**.** A gadget G is pseudo−q−NI (resp., pseudo−q−SNI) if and only if the pseudo-randomization G' of this gadget G is q−NI (resp., q−SNI).

Based on these definitions, we now have that the gadget of Equation 1 is pseudo−1−SNI, since the outputs c_i's can be simulated thanks to uniform randomness. (We will prove in Section 5.1 that this gadget is even pseudo−2−SNI). Of course, pseudo−SNI is a weaker notion than SNI and it does not guarantee composability: it rather guarantees

[6] Actual implementations are admittedly not ideal, as will be discussed in Section 4.

Table 1: Number of times that the output shares (c_1, c_2, c_3) occur for a given input (x, y, z) in a 3-share Toffoli TI gate (as given in [Bil15], Section 3.3, Figure 3.1).

	(c_1, c_2, c_3)							
(x, y, z)	000	011	101	110	001	010	100	111
000	16	16	16	16	0	0	0	0
010	16	16	16	16	0	0	0	0
100	16	16	16	16	0	0	0	0
111	16	16	16	16	0	0	0	0
001	0	0	0	0	16	16	16	16
011	0	0	0	0	16	16	16	16
101	0	0	0	0	16	16	16	16
110	0	0	0	0	16	16	16	16

"pseudo−composability" in case the pseudo−randomized monomials are manipulated with care, which is in fact exactly what state-of-the-art (1st-order) TIs exploit cleverly.

We illustrate this fact based on the excellent survey of TIs given in [Bil15]. Again ignoring glitches for now, one can observe that the gadget of Equation 1 fulfills the uniformity requirement by looking at Table 1. This is done by checking that each non-zero entry of the table equals $\frac{2^{n\cdot(d-1)}}{2^{m\cdot(d-1)}}$ with $n = 3$ and $m = 1$ the function's input and output bit-sizes, respectively, and d the number of shares. Now imagine that we want to build a 3×3-bit function based on this Toffoli gate. In a first case, we use $d = (x \odot y) \oplus z$, $e = x$, $f = y$. In a second case we use $d = (x \odot y) \oplus z$, $e = x$, $f = z$. By computing tables similar to Table 1 for these two functions (that we do not reproduce for brevity), we find out that the non-zero entries equal 1 (as required by the uniformity property) in the first case, and 2 on the second case. This indicates that the first function's output shares can directly serve as a uniform input sharing for another function. By contrast, the second function's output shares are not uniform, which is intuitively explained by observing that it forwards the pseudo–randomized monomial z (that should be used once, as per Definition 5).

We insist that our motivation for defining pseudo−SNI is only explanatory. Namely, this definition allows us to put forward the important conceptual differences between TIs and standard composable masking schemes. For example, the pseudo–composability of a single gadget such as the Toffoli gate of Equation 1 is not sufficient to ensure composability. Using this property in proofs would be delicate since it should be combined with a more global condition on the circuits to mask. So we use it next to motivate the need of a new model, and leave the investigation of alternative formal tools able to exploit and analyze pseudo–composability (e.g., in order to reduce the randomness requirements of higher-order masked implementations) as an interesting scope for further research. We also note that our pseudo–SNI definition may not be directly applicable to all TIs (and it is another interesting open problem to find out whether it can be further generalized).

3.2 The number of shares vs. cycle count tradeoff

In general, obtaining function decompositions that guarantee non-completeness and uniformity is a non-trivial task [BNN+12]. For most TIs, this comes at the cost of additional shares (e.g., in the previous instance 1st-order security is obtained with three shares rather than the minimal two). We now discuss a natural tradeoff between the number of shares and the cycle count of TIs. For this purpose, we start from the two main observations:

1. The TIs of complex circuits (e.g., S-boxes) generally result from a composition of simpler stages of gadgets, where memory elements separate the stages in order to "block" the propagation of glitches. But nothing prevents trying to split the gadgets

more than what is strictly needed for glitch-freeness (e.g., the previous Toffoli gate was implemented in one cycle, but one could also do it in two cycles).

2. In general, the correctness property is not necessary for the intermediate stages of the computation: it is sufficient that the final result is correct.

Based on these observations, it is easy to see that one possible solution to implement $\mathsf{f}(x, y, z) = (x \odot y) \oplus z$ in only two shares is given by:

$$\begin{aligned} c_1 &= \Big[\big[(x_1 \odot y_1) \oplus z_1\big] \oplus (x_1 \odot y_2)\Big], \\ c_2 &= \Big[\big[(x_2 \odot y_2) \oplus z_2\big] \oplus (x_2 \odot y_1)\Big], \end{aligned} \tag{2}$$

where the [.] parentheses are used to denote the clock cycles. Functionally, the multiplication is similar to the ISW one, but it again exploits the XOR with z in order to make the gadget pseudo−composable. Such an implementation is illustrated in Figure 2, where the circled boxes are functions and the darker rectangles are memory elements. We now have that only the result in the second stage is correct. By contrast, the intermediate stage is not (it is not even a deterministic function of the unmasked inputs). Yet, each stage of this decomposition is non-complete and uniform (w.r.t. their inputs). As in the previous subsection, this can be explained by observing that each stage of the decomposition is pseudo−1−SNI, and that the pseudo-randomized monomials are only used once in the circuit, which provides a probing-based explanation to the recent results in [CFE16].

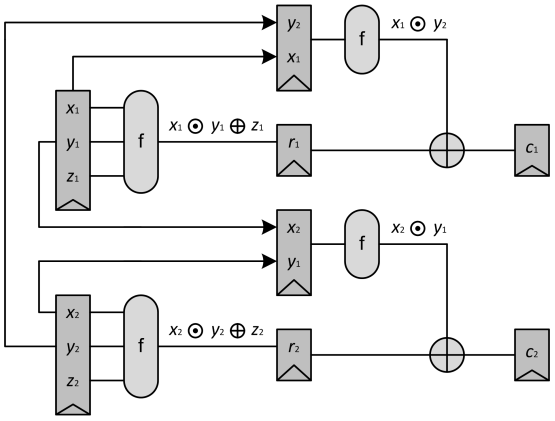

Figure 2: Example of 2-share/2-cycle decomposition of a Toffoli gate.

3.3 Generic decomposition for unbalanced Feistel networks

We finally observe that the previous Toffoli gate can be viewed as an unbalanced Feistel network with 3 branches and a degree 2 function. We systematize it to unbalanced Feistel networks with λ inputs in the left branch (entering the function f) and ρ inputs in the right branch. More precisely, the function we want to protect can be written as:

$$\mathsf{f}(\underbrace{u, v, \ldots}_{\lambda \text{ inputs}}) \oplus \underbrace{x \oplus y \oplus z \oplus \ldots}_{\rho \text{ inputs}}.$$

As illustrated in Appendix A, Figure 10, a TI can be obtained for such a function with 2 shares in at most $\frac{2^\lambda}{2\rho} + 1$ stages, which we detail as follows. First observe that if we use two shares, we have 2^λ different "non-complete sets" of shares, containing only one

share of each secret input. On each of these non-complete sets, we may need to compute a non-complete component function f_i (and strictly have to when f is of degree λ). Thus we have 2^λ partial results that we need to add to the right part of the input (that does not go through f). Next observe that in a single stage it is possible to add the output of 2ρ component functions to the 2ρ (untouched) shares of the right branch (which play the same role as the z bit in the previous subsection). This implies that we (roughly) need $\frac{2^\lambda}{2\rho}$ stages to implement the full function. Note that the generalized Feistel structure ensures that each stage is a bijection of the shares, which guarantees the uniformity property, as mentioned in [BGG+16]. Eventually, we need one more stage to compress the right branch of the network (i.e., to add the first shares together). This decomposition allows us to slightly refine the exhaustive search in [BNN+12, RBN+15b], by exhibiting different tradeoffs between the number of shares, registers and cycles in the TIs of 4-bit S-boxes. Keeping this previous work's notations where $\mathcal{Q}^4_{\mathsf{xxx}}$ denotes the quadratic class indexed xxx, and $\mathcal{C}^4_{\mathsf{xxx}}$ denotes the cubic class indexed xxx, we first remark that some classes can be written as an unbalanced Feistel network (see Appendix B). By checking the uniformity of various compositions of such networks, we found that $\mathcal{Q}^4_4, \mathcal{Q}^4_{12}, \mathcal{Q}^4_{293}, \mathcal{Q}^4_{294}$, and $\mathcal{Q}^4_{299}$ can be masked with two shares in two stages, without additional registers for the intermediate stage (needed in [RBN+15b]). We also found that C^4_1 and C^4_{13} can be masked with two shares and four stages (rather than four shares and one stage in [BNN+12] – we refer to Appendix C for the details).

4 Robust and composable probing security

In order to motivate our new model, we now argue that higher-order secure gadgets combining resistance against physical defaults and composability are not straightforward to design with existing tools. For this purpose, we once more start by ignoring physical defaults and consider the following 3-share gadget:

$$\begin{aligned} c_1 &= (x_2 \odot y_2) \oplus (x_2 \odot y_3) \oplus (x_3 \odot y_2) \oplus r_1, \\ c_2 &= (x_3 \odot y_3) \oplus (x_3 \odot y_1) \oplus (x_1 \odot y_3) \oplus r_2, \\ c_3 &= (x_1 \odot y_1) \oplus (x_1 \odot y_2) \oplus (x_2 \odot y_1) \oplus r_3. \end{aligned} \tag{3}$$

It corresponds to a variant of Equation 1 with a simple refreshing that sums a share of 0 to the partial products. Clearly, if one assumes that no information is leaked about (i.e., no probes are given on) the internal values $x_i \odot y_j$ and their intermediate sums, this implementation is 2−SNI (the proof is identical to the one given in Section 5.1, Proposition 2 for the gadget of Equation 1). The problem is that such a model is unrealistic. More precisely, a concrete hardware implementation may (and usually will [MPG05, MPO05]) leak about intermediate values via glitches (or other physical defaults). So despite this gadget is probing secure in the presence of glitches thanks to the non-completeness property, it is not SNI in this context because the intermediate randomness can be leaked due to glitches, preventing any successful simulation. The latter shows that composability alone is not sufficient to reason about higher-order masked implementations in hardware.

Taking the opposite side of the problem, it has been shown that while non-completeness and uniformity are sufficient conditions for the composability of first-order glitch-resistant circuits, it does not easily scale to higher security orders. More precisely, while so-called higher-order TIs maintain security against glitches [BGN+14], they suffer from composability issues [Rep15]. This shows that these two properties alone are not enough to reason about higher-order masked implementations in hardware. As later discussed in [RBN+15a], the addition of resfreshing gadgets is needed for this purpose. Intuitively, this is easily understood based on the discussion of pseudo–composability in the previous

section. Namely, the relevance of the uniformity property is actually related to the fact that in the context of first-order threshold implementations, one only has to prevent univariate attacks (i.e., attacks exploiting a single probe or targeting a single point in time of the leakage traces). By contrast, higher-order security requires considering multivariate attacks (i.e., attacks exploiting multiple probes or targeting multiple points in time of the leakage traces), which are not captured by the (original) definition of uniformity. Such multivariate attacks are actually the origin of the issue pointed out in [Rep15]. In this respect, one option could naturally be to try generalizing the notion of uniformity. But this would imply imposing a more global (computationally harder to assess) condition to the implementations as q increases (i.e., to get away from the concept of composability pursued in this work).

Based on these observations, we can summarize the state-of-the-art higher-order masking schemes as follows. On the one hand TIs maintain good security against shares' recombinations due to glitches but do not provide a systematic way to determine the type and amount of refreshings needed to guarantee composability. On the other hand, the probing model provides a way to reason about composability thanks to the notion of SNI but, in its original description, this model does not capture physical defaults such as glitches. In the following, we show that there is a natural generalization of the probing model that allows combining the best of these two worlds, i.e., to analyze masking gadgets that are both composable and robust against a wide class of physical defaults.[7]

4.1 Modeling physical defaults

As a starting point, we recall that the analysis of physical security properties always requires a description of the target. This is in fact already true in the (abstract) probing model, where one captures implementations as lists of (leaking) operations. Quite naturally, this requirement becomes more critical if one wishes to obtain some robustness against physical defaults. Since our goal is to incorporate a possibly large set of such defaults in our abstractions, we need to start by describing them in a more detailed manner.

For this purpose, we use the example of threshold implementation in Figure 3 where the three types of physical defaults listed in introduction are illustrated. First, combinatorial recombinations (e.g., glitches) potentially mix (and therefore recombine) the inputs of the component functions f_i. Second, memory recombinations (e.g., transitions) potentially mix (and therefore recombine) the content of the memory elements in consecutive invocations/ cycles. In Figure 3, this would typically happen if the same memory gate is used to store the y_i's by erasing the x_i's. Third, routing recombinations (i.e., couplings) potentially mix (and therefore recombine) the shares manipulated by adjacent wires.

In order to capture physical defaults, we propose to use a natural tweak of the probing model where probes are specifically or generically ϵ-*extended*. Generic extensions mean that the model is independent of the circuit topology, specific extensions are dependent on it. More precisely, we first consider the following three specific models:

Specific model for glitches. *For any ϵ-input circuit gadget G, combinatorial recombinations (aka glitches) can be modeled with specifically ϵ-extended probes so that probing any output of the gadget allows the adversary to observe all its ϵ inputs.*

Note that, as first detailed in [FG05] and recently revisited in [BM16], such a (worst-case) recombination actually happens in most cases for standard CMOS circuits. As a

[7] We insist that robust and composable gadgets are sufficient for designing higher-order secure masked implementations. Yet, it is not always necessary that all gadgets in an implementation satisfy both properties [BBP+16]. So our results leave ample room for optimization, by taking advantage of formal methods to analyze full codes [BBD+15], exploiting pseudo–randomized monomials in the higher-order setting, or observing that composability or physical defaults may be too small for being exploitable [DFS15, Dae16].

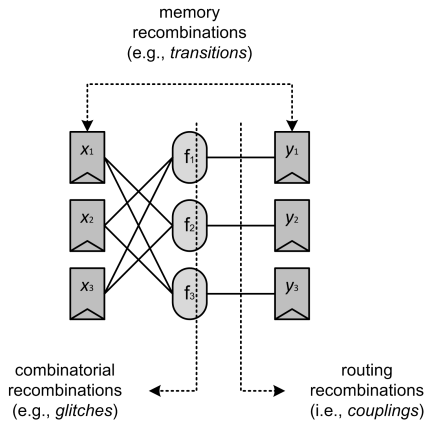

Figure 3: Physical defaults for an exemplary TI (borrowed from [BDF+17]).

result, it directly imposes a natural restriction on the topology of (robust) masked circuits. Namely, defining the shares fan-in of a gadget as the number of shares of a sensitive variable at its inputs, we generally need that *the shares fan-in of each gadget in a masked circuit should be limited* (for example, the shares fan-in of the 1st-order TIs in Section 3.1 is 2), and *any composition of gadgets with limited shares fan-in should be separated by memory elements*. The latter requirement directly comes from the fact that composing gadgets without adding memory elements in between may further increase the shares fan-in (as well known in the TI literature [Bil15]). In the rest of the paper, we will therefore mostly consider masked circuits topologies that follow these minimum guidelines.

Note also that when reasoning about composability with glitches, one generally needs to consider both extended probes and non-extended probes for some sensitive values (i.e., their glitchy signal before storage in a register, and their stable signal stored in a register). For example, this happens for the output values of the implementation in Section 5.2 which are stored in registers. It allows the q_2 probes on the stable output shares (which are excluded from the count in the SNI definition) to be non-extended, while their glitchy counterpart (which can be extended) is counted as part of the q_1 internal probes.

Note finally that similar abstractions have been used in order to capture glitches in the heuristic analyzes of [RBN+15a], and more recently in the automated analyzes of [BGI+18].

Specific model for transitions. *For a memory cell* m*, memory recombinations (aka transitions) can be modeled with specifically* 2*-extended probes so that probing* m *allows the adversary to observe any pair of values stored in 2 of its consecutive invocations.*

Note that this model exactly corresponds to the transition-based leakages introduced in [BGG+14] which have been shown a good abstraction of memory recombinations.

Specific model for couplings. *For any set of adjacent wires* $\mathcal{W} = (w_1, \ldots, w_d)$*, routing recombinations (aka couplings) can be modeled with specifically* c*-extended probes so that probing one wire* w_i *allows the adversary to observe* c *wires adjacent to* w_i*.*

Note that $c = 0$ means no couplings. Admittedly, this last physical effect is the most prospective one and it may be harder to evaluate c in practice. We add it in our modeling in order to enable the discussion of Section 4.3 and as a potential tool to state design guidelines (e.g., the limitation of c to low values) that could be combined with algorithmic properties. In general, we insist that these different models are not expected to perfectly reflect physical defaults, but to capture them sufficiently well to guide algorithmic designs with better robustness against them. They can be changed into their generic version by extending the probes without link to the circuit topology (except its maximum shares fan-in). For example, for a circuit with maximum shares fan-in f, generic glitches then

"translate" any probe in f probes (independent of whether they correspond to the same gadget); transitions translate any probe in two probes (independent of whether they correspond to the same memory cell); and generic couplings translate any probe into $c+1$ probes (independent of whether they observe adjacent wires of the circuit).

We then say that a gadget is secure in the (g,t,c)–robust q–probing model if:

- the probes are extended with glitches (iff $g=1$),
- the probes are extended with transitions (iff $t=1$),
- the probes are extended with c-couplings (for an integer $c \geq 1$),

and we use the same probe extensions in order to define (g,t,c)–robust q–NI/SNI security. The classical $q-$probing model is thus the $(0,0,0)$–robust $q-$probing model.

4.2 Worst-case generic bound

The previous model directly implies the following worst-case bound (which corresponds to a careless implementation where all physical defaults occur and are combined):

Proposition 1. *Any $2f(c+1)q$–probing secure masked circuit with maximum shares fan-in f is $(1,1,c)$–robust q–probing secure with generically extended probes.*

Note that this proposition only holds for probing security (not for NI/SNI) for a similar reason as in Section 3.1. As will be clear in Section 5.2, arguing about robust composability requires a more subtle discussion of the extended probes' positions. The proof is obvious: it simply exploits the fact that any probe is then "multiplied" by f (because of glitches), by 2 (because of transitions) and by $c+1$ (because of couplings). It directly implies that one needs $2f(c+1)q+1$ shares to obtain robust q–probing secure circuits. Naturally, one may expect that exploiting an appropriate circuit topology leads to better results, which we will discuss in Section 5. Beforehand, we discuss physical defaults' combinations and whether a more specific physical model may already improve the previous proposition.

4.3 Physical defaults combination

Looking back at Figure 3, it is clear that some types of physical defaults' combinations are unavoidable. In particular, there is no physical argument allowing one to rule out that couplings can be combined with transitions if the adversary probes adjacent memory cells. And similarly, one could combine couplings and glitches: take for example an adversary probing y_1 with a glitch-extended probe (allowing him to observe x_2 and x_3) and assume that the wire carrying x_2 is coupled with x_1 in Figure 3. So for $f=2$, a loss by a factor $2(c+1)$ in Proposition 1 seems founded, and the main question is whether the additional factor 2 corresponding to the combination of transitions and glitches is too. We discuss this question with the circuit examples given in Figure 4, which allow two observations.

First, certain transitions can be simulated by glitches. Take for example the upper circuit of the figure: in the second storage cycle, the top memory cell witnesses a transition $x_1 \Rightarrow y$. But a glitch-extended probe on y allowing an adversary to observe x_1 and x_2 can simulate this transition, since $y = \mathsf{f}(x_1, x_2)$. So there is no combination of transitions and glitches in this case. Yet, this positive result does not always hold since, for example, the $x_3 \Rightarrow y$ transition in the lower circuit cannot be simulated by a glitch-extended probe.

Second, transitions and glitches cannot be combined if the leakage samples corresponding to the storage and computation in a circuit are independent of each other. Such independent leakage samples would correspond to the oversimplified model of the top figure. If that model was perfect (which is admittedly not expected in practice), the adversary would have to choose between a glitch-extension and a transition-extension of his probes (leading to a factor $2(c+1)$ rather than $4(c+1)$ in Proposition 1).

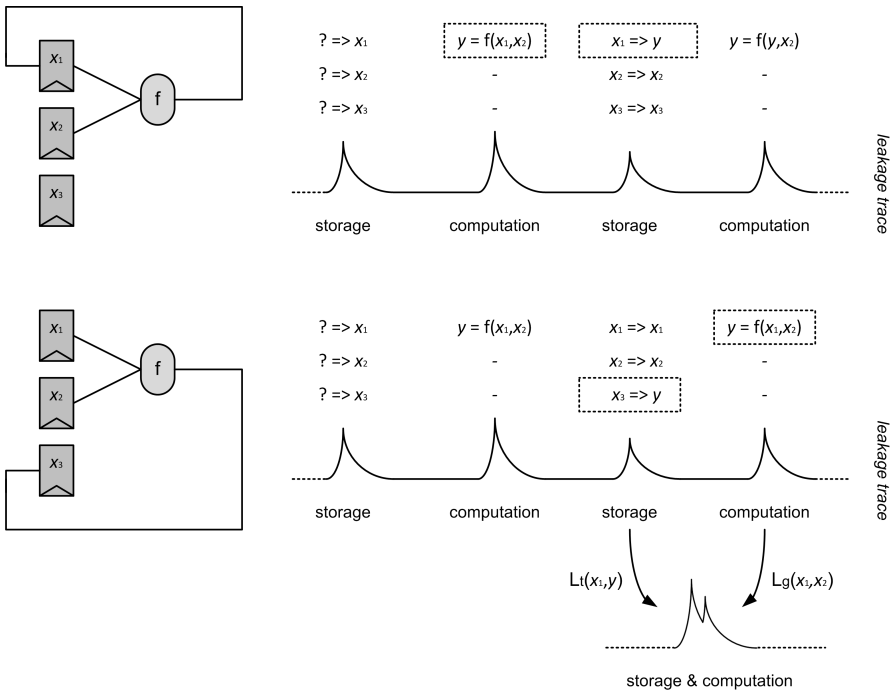

Figure 4: Exemplary combinations of transitions and glitches.

Based on these observations, we can conclude that the main question regarding the combination of transitions and glitches in masked implementations relates to their dependency, which leads to another pair of important facts: First, in practice computations within gadgets occur extremely fast after the storage, leading these two steps to overlap, as at the bottom of Figure 4. Second, such an overlap can be viewed as a type of parallel implementation (since the leakage samples due to the combinatorial gates are combined with those of the memory gates), which are known to be difficult to capture with the probing model and are better reflected by the bounded moment model [BDF+17].

In this context, we first note that whether the combination of transition-based leakages and glitch-based leakages, denoted as $\mathsf{L}_\mathrm{t}(.)$ and $\mathsf{L}_\mathrm{g}(.)$ in Figure 4, reduce the security order in the bounded moment model essentially depends on the algebraic degree of the combination function. As shown in [BDF+17], Lemma 1, a linear combination of $\mathsf{L}_\mathrm{t}(.)$ and $\mathsf{L}_\mathrm{g}(.)$ (e.g., a sum in $\mathbb{R}$) will not reduce this security order. By contrast, a non-linear one will. We then just observe that such a non-linear combination of $\mathsf{L}_\mathrm{t}(.)$ and $\mathsf{L}_\mathrm{g}(.)$ in fact exactly corresponds to the couplings of Section 4.1. Namely, couplings typically imply that the leakage of adjacent wires (or combinatorial gadgets, memory gates) are combined non-linearly, which is reflected by the extension factor c in the probing model, and is captured by an algebraic degree $c+1$ for the combination function in the bounded moment model. This reasoning finally leads us to the following conjecture:

Conjecture 1 (informal)**.** *Any* $\max(2, f)q$*–probing secure masked circuit with maximum shares fan-in* f *is* q*th-order secure in the bounded moment model if it has transitions & glitches but no non-linear combinations of transitions & glitches (i.e., couplings).*

We believe this conjecture leads to interesting guidelines for cryptographic hardware designers. It suggests that if couplings can be kept negligible within an implementation (which depends on the noise level: see [DDF14], Section 4.2), then combinations of glitches and transitions should not be detrimental to its concrete security level. We next describe experiments confirming that there are contexts in which this assumption holds.

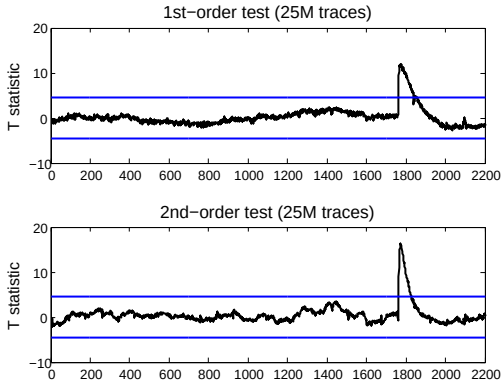

Figure 5: Non-specific T-test results for a first-order TI of the PRESENT S-box tweaked so that the register storing x_1 and y_1 in Figure 3 is re-used without refreshing.

4.4 Experimental validation

We implemented a first-order TI of the PRESENT S-box using two stages similar to the one pictured in Figure 3 and following the guidelines in [MW15], Figure 3, in a Xilinx Spartan-6 FPGA that we measured on the SAKURA-G board.[8] It provides built-in attack points to measure the voltage drop over a 1Ω shunt resistor placed in the Vdd path of the target FPGA that, by means of the corresponding voltage regulator, was supplied at 1.2 V. We ran our device at 3MHz and performed measurements by means of a Teledyne Lecroy HRO66Zi WaveRunner 12-bit digital oscilloscope (DSO) at a sampling rate of 500 MS/s and a bandwidth limit of 20 MHz to reduce the environmental noise. We used a passive probe (i.e., a SMA-to-BNC coaxial cable) that avoids the additional noise induced by, e.g., active components in differential probes. This allowed us to first reproduce the previous results of Moradi and Wild [MW15]. We then tweaked the design in two different ways.

First, rather than using six different registers to store the input and output shares $x_1, x_2, x_3, y_1, y_2, y_3$, we used the same register to store x_1 and y_1. This change is expected to lead to first-order leakages due to transitions. Second, we refreshed the output of f_1 with uniform randomness before storing it in the re-used register storing x_1. This refreshing should not improve the security order in case glitches and transitions are combined (since a glitch-extended probe on y_1 should then give this additional randomness to the adversary), and it should improve it if glitches and transitions are not combined (since the adversary should then choose between a glitch-extended probe on y_1 before it has been stored in the register, and a transition-extended probe on y_1 after it has been stored in the register).

Based on these implementations, and since only interested in the security order of our designs, we launched CRI's non-specific T-test to detect differences between the traces corresponding to fixed and random inputs [GJJR11, CMG+]. The results of these experiments are reported in Figures 5 and 6. For completeness, an exemplary trace is given in Appendix A, Figure 11. Figure 5 exhibits a first-order leakage (presumably due to transitions) when no refreshing is used. Figures 6 suggests the cancellation of this first-order leakage when the refreshing is activated. More precisely Figure 5 shows a first-order leakage of similar amplitude as the second-order one. As per [DDF14], Section 4.2, this implies that the first-order leakage will be exploitable with a similar amount of traces as the second-order one, and comparatively less when the noise increases in the measurements. Figures 6 shows a reduction of this first-order leakage to negligible for the noise level in our measurements, since a second-order leakage is then more easily detected (so the best adversarial strategy is then to estimate a second-order moment). The latter confirms that

[8] http://satoh.cs.uec.ac.jp/SAKURA/index.html

there are certain types of transitions that do not combine detrimentally with glitches. The further investigation of these combinations in different contexts is an interesting research direction.

5 Concrete constructions

We now consider the case of a couple of popular constructions from the literature and discuss if and how they differ from the previous worst-case predictions.

5.1 Equation 1 leads to pseudo$-(1,0,0)-$robust $1-$probing security

As a first example of application, we can consider the 1st-order TI gadget discussed in Section 3.1, which is a typical basis for the first-order TI of block cipher S-boxes. In our hardware implementation case, registers are selected in order to avoid transition issues so that $t = 0$ in the robust probing model. That is, the Toffoli gadget is computed in one cycle and its outputs are stored in memory gates. We first show with Proposition 2 that when implemented ideally (i.e., without glitches) the scheme is pseudo–2–SNI.

Proposition 2. *The ideal 1-cycle TI implementation of Equation 1 is pseudo–2–SNI.*

Proof. According to Definition 5, in order to prove that the gadget in Equation 1 is pseudo–2–SNI, we need to prove that its pseudo-randomization, let it be G', is 2–SNI. The algorithm G' corresponds to Equation 1, with the difference that the inputs are only the shares $x_1, x_2, x_3, y_1, y_2, y_3$ and the values z_1, z_2, z_3 are assigned uniformly at random. Let $\Omega = \{w_1, w_2\}$ be a set of 2 adversarial observations on the pseudo–randomized gadget G'. Since the implementation of the scheme is only in one cycle, the adversary does not have internal probes. Therefore the probes can only lie in one of the following two groups:

(1) the input shares x_i and y_j with $i, j \in \{1, 2, 3\}$;

(2) the output shares c_1, c_2, c_3.

Let q_1 (resp., q_2) be the number of observations on the input (resp., output) values (with $q_1 + q_2 \leq 2$). We first define two sets of indices I and J such that $|I| \leq q_1$ and $|J| \leq q_1$ and the values of the probes can be perfectly simulated given only the knowledge of $(x_i)_{i\in I}$ and $(y_j)_{j\in J}$. The sets are constructed as follows:

- Initially I and J are empty.

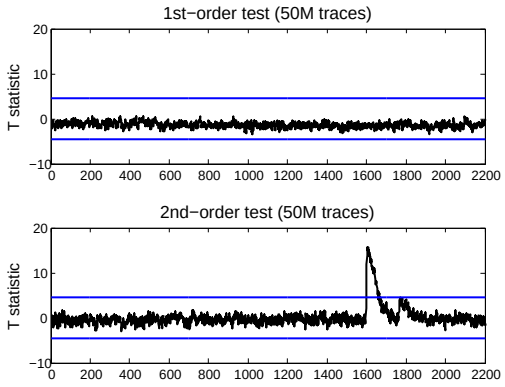

Figure 6: Non-specific T-test results for a first-order TI of the PRESENT S-box tweaked so that the register storing x_1 and y_1 in Figure 3 is re-used with refreshing.

- For every probe as in group (1), add i to I and j to J.

Since the adversary is allowed to make at most q_1 probes on the input values, it holds that $|I| \leq q_1$ and $|J| \leq q_1$. In order to prove the SNI property, we next show the simulation phase, by distinguishing the three different cases listed next:

1. If $q_1 = 2$, then the probes w_1 and w_2 are both in group (1) and, by definition of the set I, the simulator has access to the observed shares x_i and y_i.

2. If $q_1 = 1$ and $q_2 = 1$, then wlog w_1 is in group (1) and w_2 is in group (2). By definition of the sets I and J, the simulator has access to the observed shares, therefore w_1 can be perfectly simulated. As for w_2, thanks to the random value z_h with $h \in \{1, 2, 3\}$, the probe can be simulated by assigning a random and independent value.

3. Finally, if $q_2 = 2$, then w_1 and w_2 are both in group (2) and they are of the form $x_i y_i + x_i y_j + x_j y_i + z_i$ with $i \in \{1, 2, 3\}$. Since the (pseudo–randomized) shares of z appearing in their computation are different in each output share, the simulator can also assign w_1 and w_2 to a random and independent value.

In all the cases listed above, the probes w_1 and w_2 can be perfectly simulated with q_1 shares of the input. We finally note that if $|\Omega| = 1$, then the simulation of the probe trivially follows the procedure of one of the previous cases. Therefore we conclude that the gadget G' is 2–SNI, completing the proof. □

Combining this result and Proposition 1, it follows that this TI gadget is pseudo–(1,0,0)–robust 1–probing secure with 3 shares. In other words, it uses an additional share to prevent glitches, exactly following worst-case analysis. Note that (as mentioned in Section 4.2) the resulting gadget is not pseudo–(1,0,0)–robust 1–SNI (since glitch-extended probes cannot be simulated with one share per input). Yet, it is sufficient to argue about the security of TIs for full ciphers. Assuming the uniformity condition in Section 3.1 is fulfilled, such "full TIs" are pseudo–2–SNI without glitches. By invoking Proposition 1 only once, we have that they are also (1,0,0)–robust 1–probing secure.

5.2 ISW is $(1, 0, 0)$-robust $q-$SNI with $q + 1$ shares in 2 cycles

We now show that when moving to higher-orders the ISW multiplication actually beats our worst-case bound, and therefore provides an excellent solution for robust and composable gadgets. More precisely, it is proven in [BBD+16] that this algorithm is (0,0,0)–robust q–SNI, using $q + 1$ shares. We next show formally that the scheme is additionally (1,0,0)–robust q–SNI (or glitch–robust q–SNI for short), if one precisely follows the guidelines of Section 4.1 and limits the shares fan-in to 1. For this purpose, we will consider an implementation of the ISW multiplication in two cycles illustrated in Figure 7 for the case with 3 shares and security order 2. A generic description can be obtained by using the notations of Section 3.2 and adding one level of brackets around the $u_{j,i}$ and $u_{i,j}$ variables of Algorithm 1 (representing the operations performed in the first cycle) and a second level of brackets around the c_i variables (representing the operations performed in the second cycle).

In this respect, it is first important to recall that compared to the previous section, we now use the specific model for glitches, that exploits this particular circuit topology. Concretely, it means that for the implementation illustrated in Figure 7, the adversary can access the three types of probes that we describe next:

- Internal (3-extended) probes $p_{i,j}$ on the $u_{i,j}$'s giving access to three shares: namely a_i, b_j and the corresponding value of the randomness matrix.
- Internal (3-extended) probes p_i on the c_i's giving access to $u_{i,1}$, $u_{i,2}$, $u_{i,3}$.

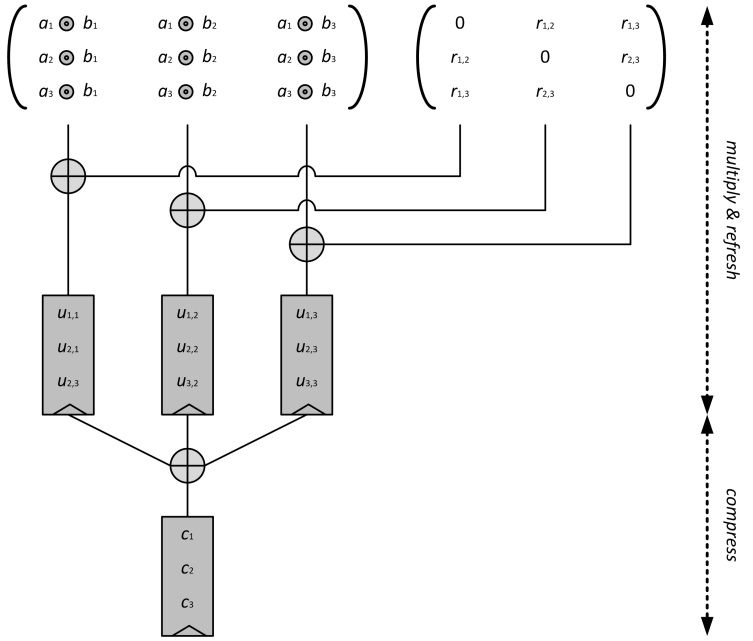

Figure 7: $(1, 0, 0)$-robust 2-SNI implementation of ISW in 2 cycles.

- Output (non-extended) probes on the c_i's giving only access to one share.

Note that in this model, an adversary willing to obtain a single internal value (e.g., an $r_{i,j}$) will simply use a (more informative) extended probe including this value. Note also that despite giving 3-extended probes to the adversary, we do not break the shares fan-in limit of 1. Besides, and quite importantly, the c_i shares appear twice in the list: either as internal probes which can be glitch-extended, or as external probes which are not glitchy since stored in an additional memory element. Despite not being necessary for probing security, the additional output memory elements storing the c_i shares are strictly necessary in order to obtain a robust and composable gadget, which we formalize as follows:

Proposition 3. *The multiplication gadget in Algorithm 1 implemented in two cycles (one for the $u_{i,j}$ values, a second one for the c_i values) is (1,0,0)–robust q–SNI.*

The proof of the Proposition is in Appendix D. In order to provide some intuition, we illustrate it with the 3-share implementation of Figure 7. An adversary attacking the internal values of this scheme can observe at most two of the following extended probes, each of them allowing him to see between 2 and 3 shares:

- *1st stage:*

$$\begin{array}{lll} p_{1,1} := (a_1, b_1, 0), & p_{1,2} := (a_1, b_2, r_{1,2}), & p_{1,3} := (a_1, b_3, r_{1,3}), \\ p_{2,1} := (a_2, b_1, r_{1,2}), & p_{2,2} := (a_2, b_2, 0), & p_{2,3} := (a_2, b_3, r_{2,3}), \\ p_{3,1} := (a_3, b_1, r_{1,3}), & p_{3,2} := (a_3, b_2, r_{2,3}), & p_{3,3} := (a_3, b_3, 0). \end{array}$$

- *2nd stage:*

$$p_1 := (u_{1,1}, u_{1,2}, u_{1,3}), \qquad p_2 := (u_{2,1}, u_{2,2}, u_{2,3}), \qquad p_3 := (u_{3,1}, u_{3,2}, u_{3,3}).$$

Alternatively, attacking output shares allows the observation of shares c_1, c_2, c_3. In the simulation, we therefore distinguish the following possible cases:

1. Both probes are on the internal shares (e.g., $p_{1,2}, p_1$).
2. One probe is internal, the other is on the output shares (e.g., $p_{1,2}, c_1$).
3. Both probes are on the output shares (e.g., c_1, c_2).

In the first case, and according to the proof, we construct the set of indices $I = \{1\}$ and $J = \{1, 2\}$. In the simulation phase we assign $r_{1,2}$ to a random value and we can perfectly compute $p_{1,2}$ by having access to a_1 and b_2. As for p_1, we perfectly simulate the first component $u_{1,1}$ by using a_1 and b_1; since $2 \in J$ and $2 \notin I$ we can use the components of the probed $p_{1,2}$ to simulate the second component $u_{1,2}$; and since $3 \notin J$ we can pick a uniform and random value for simulating the third component $u_{1,3}$.

In the second case, we have $I = \{1\}$ and $J = \{1, 2\}$. We simulate $p_{1,2}$ as before and we assign a uniform and random value to c_1, thank to the presence of the random bit $r_{1,3}$.

In the third case, since c_1 depends on the random bit $r_{1,3}$ which does not appear in the computation of c_2, and c_2 depends on the random bit $r_{2,3}$ which does not appear in the computation of c_1, we can simulate both shares as random and independent values.

We insist that despite our 2-cycle implementation is directly inspired by the ISW construction, its proof is not implied by the (previous) proofs of ISW-like multiplications. In particular, the extended probes actually give more information to the adversary than in the software setting analyzed by [RP10] and follow-up works. More precisely, the success of the simulation for the output values is due to the careful distribution of the random bits in the different registers. Indeed, each output share depends on a number of distinct random bits equal to the security order, and these random bits appear a second time in the computation of only one different output share each. This allows us to simulate the output probes with a random and independent value, and therefore to use the required number of input shares in order to satisfy the definition of SNI. The main overhead of this glitch-robust and composable multiplication is the need of $d^2 + d$ registers, to store the partial products in a first cycle and compress the output in a second cycle. It is an interesting open problem to determine whether robust and composable gadgets could be obtained in two cycles with less registers and/or randomness than in this section (e.g., by arranging the operations differently), or if such optimizations (in particular, the reduced number of registers) can only be obtained at the cost of an increased number of cycles.

5.3 Glitch locality principle

The previous proof highlighted that robust and composable implementations of the ISW multiplication require that their outputs c_i's are stored in memory gates, in order to stop the propagation of glitches in the circuit. This leads to the following formalization:

Proposition 4. *If a gadget* G *storing its outputs in registers is both* $(1, 0, 0)-$*robust* $q-$*NI and* $q-$*SNI (without glitches), then it is also* $(1, 0, 0)-$*robust* q*-SNI.*

Proof. By separating the probes between q_1 internal and q_2 output ones, we have that: (i) the internal probes can be simulated with q_1 shares per input since the gadget is $(1, 0, 0)-$robust q_1-probing secure (with $q_1 \leq q$), and (ii) the q_2 probes can be simulated with q_1 input shares since the gadget is $q-$SNI without glitches. □

This proposition shows that the glitch issue is in part "internal" to the masking gadgets. If registers are inserted after those gadgets, a designer can deal with glitch robustness (captured with the $(1, 0, 0)-$robust NI notion) and composability (captured with the $(0, 0, 0)-$robust SNI notion) separately. Glitches and composability are not independent issues though, since glitch–robust q-probing security is not enough for the lemma (i.e., some form of simulatability, captured by the glitch–robust NI notion, is needed) [MMSS18].

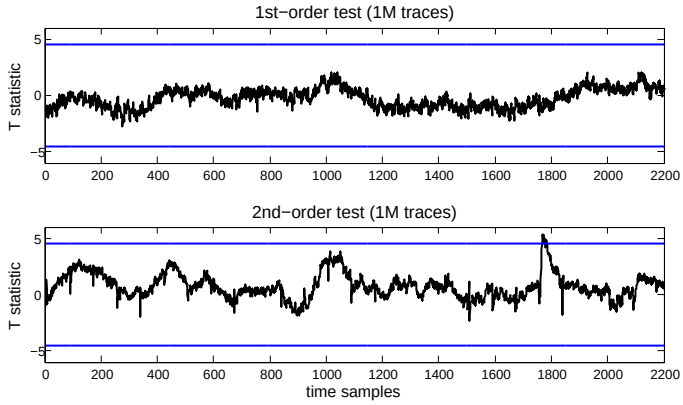

Figure 8: Non-specific T-test results for $d = 2$ shares.

6 Practical security evaluation

We implemented the 2-cycle architecture of Figure 7 in a Xilinx Spartan-6 FPGA for $d = 2$ and 3 shares, using exactly the same setup as in Section 4.4. Based on this setup, and since only interested in the security order of our designs, we again launched CRI's non-specific T-test to detect differences between the traces corresponding to fixed inputs and random inputs [GJJR11, CMG+]. In the $d = 2$ case, we were able to spot second-order leakages with 1 million measurements (see Figure 8). In the $d = 3$ case, we used 10 millions measurements and exploited the tweak proposed in [Sta17], Section 3.2, (i.e., we repeated 50 times the measurement of 250,000 traces and averaged them in order to mitigate the noise amplification due to masking and to speed up the detection). This allowed us to detect third order leakages (see Figure 9). None of our experiments suggested any lower-order leakage, confirming the results in [GMK17]. Thanks to the composability of our implementations, we can therefore claim for the first time that a combination of such higher-order hardware gadgets will remain robust against glitches and maintain their security for full (e.g., block cipher) implementations, as validated experimentally for the cipher SIMON in [RBG+15].

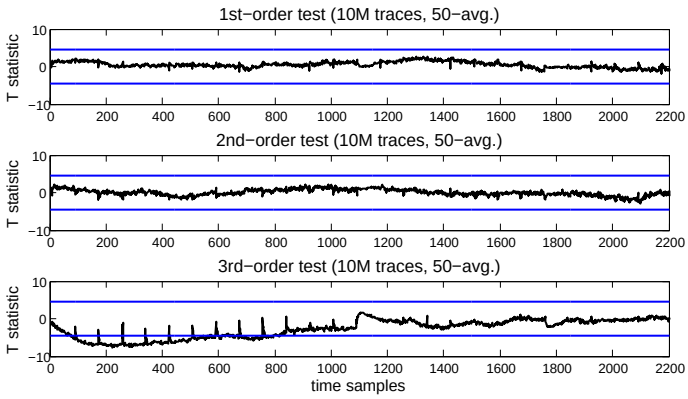

Figure 9: Tweaked non-specific T-test results for $d = 3$ shares.

7 Related work

Several recent works aimed at achieving higher-order side-channel security in the presence of glitches, including the Consolidated Masking Scheme (CMS) of Reparaz et al. [RBN+15a] instantiated in [CRB+16], the Domain Oriented Masking (DOM) in [GMK16, GMK17] and the Unified Masking Approach (UMA) in [GM17]. As recently discussed in [MMSS18], none of these schemes come with a security proof at arbitrary orders, making comparisons difficult. Furthermore, the same reference shows that this lack of proof is not only a theoretical concern and that probing security weaknesses (due to local or composability flaws) can be exhibited for all these references, as the number of shares in their masking schemes increases. Based on this state-of-the-art, and to the best of our knowledge, the implementation of Section 5.2 is the only published algorithm that is jointly robust against glitches and composable at arbitrary orders with $d + 1$ shares. We note that the parallel masking algorithm introduced by Barthe et al. in [BDF+17] exploits the same "compute partial products – refresh – compress" structure as our implementation in Figure 7. So despite more specialized to software implementations, it can lead to similar 2-cycle hardware implementations.

8 Composition rules

Before to conclude, we discuss how composable gadgets can be assembled to build complex circuits. First ignoring glitches for simplicity, we recall that there are two main approaches for this purpose. One is to select an appropriate combination of NI and SNI gadgets. The latter usually requires some further analysis / optimization [BBP+16]. The other is to go for the simpler but more expensive strategy proposed in [GR17] and proven in [CS18], which is to consider implementations where all multiplications are MIMO-SNI (i.e., Multiple-Input Multiple-Output SNI, which can be obtained by "refreshing" one input of a SNI multiplication), and all linear operations are simply performed share by share.[9]

Interestingly, our modeling implies that the same composition rules apply to glitchy implementations as long as the output shares of the (robust) multiplications are stored in registers. As previously mentioned, this prevents their glitch-extension (which also leads to the glitch locality principle of Section 5.3). So based on the glitch-robust implementation of a SNI multiplication in the Section 5.2, one can directly design complex circuits (e.g., S-boxes or full ciphers) following one of the aforementioned approaches.

9 Conclusions

While usually based on similar patterns (starting with the computation of partial products), higher-order masked multiplications can differ in the way they deal with composability and robustness thanks to refreshings and memory elements. Their design is subtle and error-prone, and generally benefits from formal proofs, especially when the number of shares increases (which makes exhaustive analysis impossible). We believe the robust probing model brings three interesting features in this respect. First, it allows formally guiding implementation choices related to physical defaults that so far required engineering intuition. Second it can lead to implementations providing robustness against physical defaults and composability jointly. Third, it is versatile since by tuning the g, t and c parameters, we can ask more or less to hardware designers, hence enabling to trade risks of implementation surprises and performance overheads. We insist that not being robust

[9] Refreshing gadgets that are robust against glitches can for example be instantiated with a SNI multiplication by one thanks to the algorithm in Section 5.2 (and can even be simplified to a 1-cycle implementation since each output share depends only on one input share in this case).

and composable does not imply that an implementation is insecure. It only implies that its security evaluation is more complex, since one cannot leverage the local security order analysis of simple gadgets, and rather has to deal directly with the complexity of full implementations. The introduction of the robust probing model is also beneficial in the latter case, since it enables the analysis of physical defaults in masking schemes with automated solvers, as recently undertaken in [BGI+18]. Its combination with the other formal tools such as [BBD+15], to analyze large circuits, is yet another interesting research direction.

Acknowledgments

The authors are grateful to François Dupressoir, Tobias Schneider and the CHES 2017/2018 reviewers for useful comments and feedback. Sebastian Faust is funded by the Emmy Noether Program FA 1320/1-1 of the German Research Foundation (DFG). François-Xavier Standaert is a senior associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded by the European Commission through the ERC project 724725 (acronym SWORD).

References

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In Oswald and Fischlin [OF15], pages 457–485.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 116–129. ACM, 2016.

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 616–648. Springer, 2016.

[BCPZ16] Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Gierlichs and Poschmann [GP16], pages 23–39.

[BDF+17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Coron and Nielsen [CN17], pages 535–566.

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *CARDIS 2014*, volume 8968 of *LNCS*, pages 64–81. Springer, 2014.

[BGG+16] Erik Boss, Vincent Grosso, Tim Güneysu, Gregor Leander, Amir Moradi, and Tobias Schneider. Strong 8-bit S-boxes with efficient masking in hardware. In Gierlichs and Poschmann [GP16], pages 171–193.

[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 Nielsen and Rijmen [NR18], pages 321–353.

[BGN+14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014 , Part II*, volume 8874 of *LNCS*, pages 326–343. Springer, 2014.

[BGNT15] Nicolas Bruneau, Sylvain Guilley, Zakaria Najm, and Yannick Teglia. Multivariate high-order attacks of shuffled tables recomputation. In Güneysu and Handschuh [GH15], pages 475–494.

[Bil15] Begül Bilgin. Threshold implementations: A countermeasure against higher-order differential power analysis. PhD Thesis, KU Leuven (Belgium) and U Twente (The Netherlands), May 2015.

[BM16] Guido Bertoni and Marco Martinoli. A methodology for the characterisation of leakages in combinatorial logic. In Carlet et al. [CHS16], pages 363–382.

[BNN+12] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold implementations of all 3 x 3 and 4 x 4 S-Boxes. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 76–91. Springer, 2012.

[CBG+17] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In Sylvain Guilley, editor, *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017*, volume 10348 of *LNCS*, pages 1–18. Springer, 2017.

[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 Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 819–843, 2016.

[CGP+12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In Werner Schindler and Sorin A. Huss, editors, *COSADE 2012*, volume 7275 of *LNCS*, pages 69–81. Springer, 2012.

[CHS16] Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors. *SPACE 2016*, volume 10076 of *LNCS*. Springer, 2016.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO 1999*, volume 1666, pages 398–412. Springer, 1999.

[CMG+] Jeremy Cooper, Elke De Mulder, Gilbert Goodwill, Josh Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test vector leakage assessment (TVLA) methodology in practice (extended abstract). ICMC 2013. http://icmc-2013.org/wp/wp-content/uploads/2013/09/goodwillkenworthtestvector.pdf.

[CN17] Jean-Sébastien Coron and Jesper Buus Nielsen, editors. *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *LNCS*, 2017.

[CPR07] Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side channel cryptanalysis of a higher order masking scheme. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 28–44. Springer, 2007.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Moriai [Mor14], pages 410–424.

[CRB+16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 shares in hardware. In Gierlichs and Poschmann [GP16], pages 194–212.

[CS18] Gaëtan Cassiers and François-Xavier Standaert. Improved bitslice masking: from optimized non-interference to probe isolation. Cryptology ePrint Archive, Report 2018/438, 2018. https://eprint.iacr.org/2018/438.

[Dae16] Joan Daemen. Spectral characterization of iterating lossy mappings. In Carlet et al. [CHS16], pages 159–178.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, 2014.

[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 Oswald and Fischlin [OF15], pages 401–429.

[DS16] François Durvaux and François-Xavier Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 240–262. Springer, 2016.

[FG05] Wieland Fischer and Berndt M. Gammel. Masking at gate level in the presence of glitches. In Rao and Sunar [RS05], pages 187–200.

[GH15] Tim Güneysu and Helena Handschuh, editors. *CHES 2015*, volume 9293 of *LNCS*. Springer, 2015.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side channel resistance validation. NIST non-invasive attack testing workshop, 2011. http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf.

[GM17] Hannes Gross and Stefan Mangard. Reconciling d+1 masking in hardware and software. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, 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. Cryptology ePrint Archive, Report 2016/486, 2016. http://eprint.iacr.org/2016/486.

[GMK17] Hannes Gross, Stefan Mangard, and Thomas Korak. An efficient side-channel protected AES implementation with arbitrary protection order. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 95–112. Springer, 2017.

[GP16] Benedikt Gierlichs and Axel Y. Poschmann, editors. *CHES 2016*, volume 9813 of *LNCS*. Springer, 2016.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In Coron and Nielsen [CN17], pages 567–597.

[GS18] Vincent Grosso and François-Xavier Standaert. Masking proofs are tight and how to exploit it in security evaluations. In Nielsen and Rijmen [NR18], pages 385–412.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.

[MMSS18] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. Glitch-resistant masking revisited - or why proofs in the robust probing model are needed. Cryptology ePrint Archive, Report 2018/490, 2018. https://eprint.iacr.org/2018/490.

[MOBW13] Luke Mather, Elisabeth Oswald, Joe Bandenburg, and Marcin Wójcik. Does my device leak information? An a priori statistical power analysis of leakage detection tests. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 486–505. Springer, 2013.

[Mor14] Shiho Moriai, editor. *FSE 2013*, volume 8424 of *LNCS*. Springer, 2014.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 351–365. Springer, 2005.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In Rao and Sunar [RS05], pages 157–171.

[MW15] Amir Moradi and Alexander Wild. Assessment of hiding the higher-order leakages in hardware - what are the achievements versus overheads? In Güneysu and Handschuh [GH15], pages 453–474.

[NR18] Jesper Buus Nielsen and Vincent Rijmen, editors. *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*. Springer, 2018.

[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.

[OF15] Elisabeth Oswald and Marc Fischlin, editors. *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*. Springer, 2015.

[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 Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 142–159. Springer, 2013.

[RBG+15] Debapriya Basu Roy, Shivam Bhasin, Sylvain Guilley, Jean-Luc Danger, and Debdeep Mukhopadhyay. From theory to practice of private circuit: A cautionary note. In *33rd IEEE International Conference on Computer Design, ICCD 2015, New York City, NY, USA, October 18-21, 2015*, pages 296–303. IEEE Computer Society, 2015.

[RBN+15a] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 764–783. Springer, 2015.

[RBN+15b] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. Cryptology ePrint Archive, Report 2015/719, 2015. http://eprint.iacr.org/2015/719.

[Rep15] Oscar Reparaz. A note on the security of higher-order threshold implementations. Cryptology ePrint Archive, Report 2015/001, 2015. http://eprint.iacr.org/2015/001.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 413–427. Springer, 2010.

[RS05] Josyula R. Rao and Berk Sunar, editors. *CHES 2005*, volume 3659 of *LNCS*. Springer, 2005.

[SM16] Tobias Schneider and Amir Moradi. Leakage assessment methodology - extended version. *J. Cryptographic Engineering*, 6(2):85–99, 2016.

[Sta17] François-Xavier Standaert. How (not) to use welch's t-test in side-channel security evaluations. Cryptology ePrint Archive, Report 2017/138, 2017. http://eprint.iacr.org/2017/138.

[TWO13] Michael Tunstall, Carolyn Whitnall, and Elisabeth Oswald. Masking tables - an underestimated security risk. In Moriai [Mor14], pages 425–444.

A Additional figures

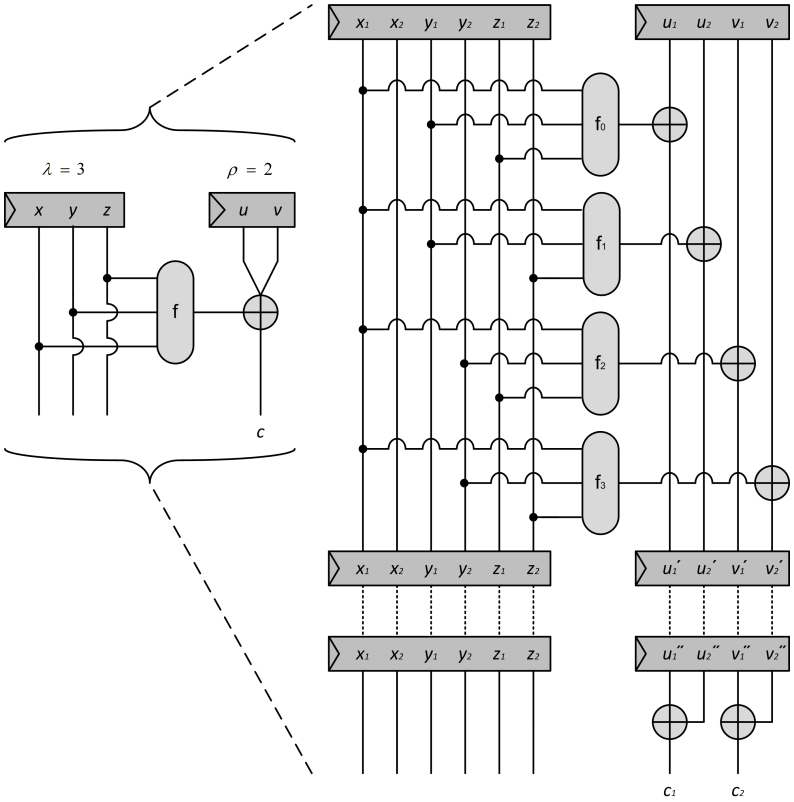

Figure 10: Unbalanced Feistel network (left) and its TI (right).

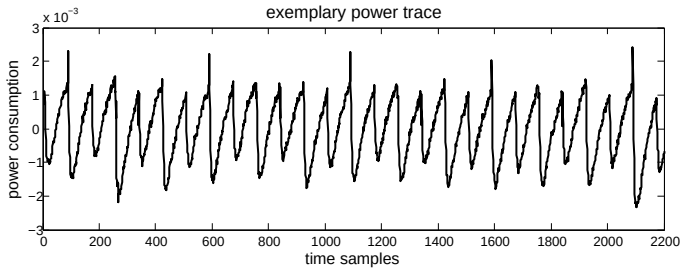

Figure 11: Exemplary power trace for the implementation of Figure 7.

B 4-bit functions with generalized Feistel representation

$$
\begin{aligned}
Q_4^4 &: (x, y, z, t) \mapsto (x \oplus zt, y, z, t) \\
Q_{12}^4 &: (x, y, z, t) \mapsto (x, y \oplus (y \oplus z)t, z \oplus yt, t) \\
Q_{293}^4 &: (x, y, z, t) \mapsto (x \oplus yz, y \oplus (y \oplus z)t, z \oplus yt, t) \\
Q_{294}^4 &: (x, y, z, t) \mapsto (x \oplus yt, y \oplus zt, z \oplus yt, t) \\
Q_{299}^4 &: (x, y, z, t) \mapsto (x \oplus (x \oplus z)t, y \oplus (x \oplus y \oplus z)t, z \oplus (y \oplus z)t, t) \\
C_1^4 &: (x, y, z, t) \mapsto (x \oplus yzt, y, z, t) \\
C_2^4 &: (x, y, z, t) \mapsto (x \oplus (x \oplus y)zt, y \oplus xzt, z, t) \\
C_3^4 &: (x, y, z, t) \mapsto (x \oplus zt, y \oplus xzt, z, t) \\
C_{13}^4 &: (x, y, z, t) \mapsto (x \oplus yzt, y \oplus (y \oplus z)t, z \oplus yt, t) \\
C_{243}^4 &: (x, y, z, t) \mapsto (x \oplus (x + z)(t + yt) \oplus yz, y \oplus (x \oplus y)t, z \oplus (y \oplus z)t, t)
\end{aligned}
$$

C 4-bit shared functions

C.1 Q_4^4

$$
\begin{array}{ll}
x_1 & \Big[\big[x_1 \oplus (z_1 \odot t_1)\big] \oplus (z_1 \odot t_2)\Big] \\
x_2 & \Big[\big[x_2 \oplus (z_2 \odot t_2)\big] \oplus (z_2 \odot t_1)\Big] \\
y_1 & y_1 \\
y_2 & y_2 \\
z_1 & z_1 \\
z_2 & z_2 \\
t_1 & t_1 \\
t_2 & t_2
\end{array}
$$

C.2 Q_{12}^4

$$
\begin{array}{ll}
x_1 & x_1 \\
x_2 & x_2 \\
y_1 & \Big[\big[y_1 \oplus ((z_1 \oplus y_1) \odot t_1)\big] \oplus ((z_1 \oplus y_1) \odot t_2)\Big] \\
y_2 & \Big[\big[y_2 \oplus ((z_2 \oplus y_2) \odot t_2)\big] \oplus ((z_2 \oplus y_2) \odot t_1)\Big] \\
z_1 & \Big[\big[z_1 \oplus (y_1 \odot t_1)\big] \oplus (y_1 \odot t_2)\Big] \\
z_2 & \Big[\big[z_2 \oplus (y_2 \odot t_2)\big] \oplus (y_2 \odot t_1)\Big] \\
t_1 & t_1 \\
t_2 & t_2
\end{array}
$$

C.3 Q_{293}^4

$$
\begin{array}{ll}
x_1 & \Big[\big[x_1 \oplus (z_1 \odot y_1)\big] \oplus (z_2 \odot y_1)\Big] \\
x_2 & \Big[\big[x_2 \oplus (z_2 \odot y_2)\big] \oplus (z_1 \odot y_2)\Big] \\
y_1 & \Big[\big[y_1 \oplus ((z_1 \oplus y_1) \odot t_1)\big] \oplus ((z_1 \oplus y_1) \odot t_2)\Big] \\
y_2 & \Big[\big[y_2 \oplus ((z_2 \oplus y_2) \odot t_2)\big] \oplus ((z_2 \oplus y_2) \odot t_1)\Big] \\
z_1 & \Big[\big[z_1 \oplus (y_1 \odot t_1)\big] \oplus (y_1 \odot t_2)\Big] \\
z_2 & \Big[\big[z_2 \oplus (y_2 \odot t_2)\big] \oplus (y_2 \odot t_1)\Big] \\
t_1 & t_1 \\
t_2 & t_2
\end{array}
$$

C.4 Q^4_{294}

$$
\begin{array}{ll}
x_1 & \Big[\big[x_1 \oplus (t_1 \odot y_1)\big] \oplus (t_2 \odot y_1)\Big] \\
x_2 & \Big[\big[x_2 \oplus (t_2 \odot y_2)\big] \oplus (t_1 \odot y_2)\Big] \\
y_1 & \Big[\big[y_1 \oplus (z_1 \odot t_1)\big] \oplus (z_2 \odot t_1)\Big] \\
y_2 & \Big[\big[y_2 \oplus (z_2 \odot t_2)\big] \oplus (z_1 \odot t_2)\Big] \\
z_1 & z_1 \\
z_2 & z_2 \\
t_1 & t_1 \\
t_2 & t_2
\end{array}
$$

C.5 Q^4_{299}

$$
\begin{array}{ll}
x_1 & \Big[\big[x_1 \oplus (t_1 \odot (x_1 \oplus z_1))\big] \oplus (t_2 \odot (x_1 \oplus z_1))\Big] \\
x_2 & \Big[\big[x_2 \oplus (t_2 \odot (x_2 \oplus z_2))\big] \oplus (t_1 \odot (x_2 \oplus z_2))\Big] \\
y_1 & \Big[\big[y_1 \oplus ((x_1 \oplus y_1 \oplus z_1) \odot t_1)\big] \oplus ((x_1 \oplus y_1 \oplus z_1) \odot t_2)\Big] \\
y_2 & \Big[\big[y_2 \oplus ((x_2 \oplus y_2 \oplus z_2) \odot t_2)\big] \oplus ((x_2 \oplus y_2 \oplus z_2) \odot t_1)\Big] \\
z_1 & \Big[\big[z_1 \oplus ((y_1 \oplus z_1) \odot t_1)\big] \oplus ((y_1 \oplus z_1) \odot t_2)\Big] \\
z_2 & \Big[\big[z_2 \oplus ((y_2 \oplus z_2) \odot t_2)\big] \oplus ((y_2 \oplus z_2) \odot t_1)\Big] \\
t_1 & t_1 \\
t_2 & t_2
\end{array}
$$

C.6 C^4_1

$$
\begin{array}{ll}
x_1 & \Big[\Big[\big[\big[x_1 \oplus (y_1 \odot z_1 \odot t_1)\big] \oplus (y_1 \odot z_1 \odot t_2)\big] \oplus (y_1 \odot z_2 \odot t_1)\Big] \oplus (y_2 \odot z_1 \odot t_1)\Big] \\
x_2 & \Big[\Big[\big[\big[x_2 \oplus (y_2 \odot z_2 \odot t_2)\big] \oplus (y_2 \odot z_2 \odot t_1)\big] \oplus (y_2 \odot z_1 \odot t_2)\Big] \oplus (y_1 \odot z_2 \odot t_2)\Big] \\
y_1 & y_1 \\
y_2 & y_2 \\
z_1 & z_1 \\
z_2 & z_2 \\
t_1 & t_1 \\
t_2 & t_2
\end{array}
$$

C.7 C_{13}^{4}

$$
\begin{aligned}
x_1 &\quad \Big[\Big[\Big[\big[x_1 \oplus (y_1 \odot z_1 \odot t_1)\big] \oplus (y_1 \odot z_1 \odot t_2)\Big] \oplus (y_1 \odot z_2 \odot t_1)\Big] \oplus (y_2 \odot z_1 \odot t_1)\Big] \\
x_2 &\quad \Big[\Big[\Big[\big[x_2 \oplus (y_2 \odot z_2 \odot t_2)\big] \oplus (y_2 \odot z_2 \odot t_1)\Big] \oplus (y_2 \odot z_1 \odot t_2)\Big] \oplus (y_1 \odot z_2 \odot t_2)\Big] \\
y_1 &\quad \Big[\big[y_1 \oplus ((y_1 \oplus z_1) \odot t_1)\big] \oplus ((y_1 \oplus z_1) \odot t_2)\Big] \\
y_2 &\quad \Big[\big[y_2 \oplus ((y_2 \oplus z_2) \odot t_2)\big] \oplus ((y_2 \oplus z_2) \odot t_1)\Big] \\
z_1 &\quad \Big[\big[z_1 \oplus (y_1 \odot t_1)\big] \oplus (y_1 \odot t_2)\Big] \\
z_2 &\quad \Big[\big[z_2 \oplus (y_2 \odot t_2)\big] \oplus (y_2 \odot t_1)\Big] \\
t_1 &\quad t_1 \\
t_2 &\quad t_2
\end{aligned}
$$

D Proof of Proposition 3

Let $\Omega = (\mathcal{I}, \mathcal{O})$ be a set of q adversary's observations respectively on the internal and on the output values, where $|\mathcal{I}| = q_1$ and in particular $q_1 + |\mathcal{O}| \leq q$. We construct a perfect simulator of the adversary's probes, which makes use of at most q_1 shares of the secrets x and y.

Let $w_1, \ldots, w_q$ be the probed values. According to the specific model for glitches presented in Section 4.1, the possible internal extended probes can be classified in the following groups:

(1) $p_{i,j} := (a_i, b_j, r_{i,j})$ with $i, j = 1, \ldots, q+1$

(2) $p_i := (u_{i,1}, \ldots, u_{i,q+1})$ with $i = 1, \ldots, q+1$

On the other hand, since the output shares are stored in registers, glitches do not affect them and so the possible probes on the output shares are, as in the non-robust probing model, the c_i with $i = 1, \ldots, q+1$, as in Algorithm 1.

We define two sets of indices I and J such that $|I| \leq q_1$, $|J| \leq q_1$ and the values of the probes can be perfectly simulated given only the knowledge of $(x_i)_{i \in I}$ and $(y_i)_{i \in J}$. The sets are constructed as follows.

- Initially I and J are empty.
- For every probe as in group (1) add i to I and j to J.
- For every probe as in group (2) add i to I and moreover for every probe of the form $p_{j,i}$ add j to J.

Since the adversary is allowed to make at most q_1 internal probes, it holds that $|I| \leq q_1$ and $|J| \leq q_1$.

We now show the simulation phase. First of all, the simulator assigns a random value to every $r_{i,j}$ entering in the computation of any probe. Then we consider an observed value w_h in group (1). In this case, by definition of I and J the simulator has access to a_i and b_j and we distinguish three cases:

- If $i = j$, the simulator assigns $r_{i,i}$ to 0 and then perfectly simulates w_h using a_i and b_i.
- If $j \in I$ and $i \in J$, then by definition the adversary has probed also $p_{j,i}$ or p_i and p_j. Therefore, in any case, the adversary has already probed a value containing in its computation the random bit $r_{i,j}$. The simulator then perfectly simulates w_h using a_i, b_j and the $r_{i,j}$ assigned previously.
- In all the other cases, $r_{i,j}$ does not enter in the computation of any other probe, and therefore the simulator can assign w_h to a random and independent value.

As for a probe w_h in group (2), by definition $i \in I, J$. So the simulator can perfectly compute the ith-component of the probe using a_i, b_i. For each of the remaining jth-components of p_i we distinguish the following cases.

- If $j \in J$ and $j \notin I$, then the adversary has already probed $p_{i,j}$, which can be simulated as in the first phase and entirely used as jth-component of w_h.
- If $j \in J$ and $j \in I$, then the adversary has already probed $p_{i,j}$ or p_j or $p_{j,i}$. In the first case the simulator follows the previous step. In both the latter cases, $r_{i,j}$ was assigned in the preliminary phase and can be used with a_i and b_j to simulate the jth-component of w_h.

- If $j \notin J$, the simulator assigns to the jth-component of p_i a random and independent value: indeed, the bit $r_{i,j}$ involved in the computation of such a component is not used in any other probe.

We conclude the proof by showing how to simulate a probe w_h in the output values. We notice that since in this case the probes are as in the traditional probing model, the proof is really similar to the one of Proposition 2 in [BBD+15]. We have to take into account the following two cases:

- If the attacker has observed also some of the internal values, then the partial sums previously probed are already simulated. As for the remaining terms, we note that by definition of the scheme there always exists one random bit $r_{k,l}$ in w_h, which does not appear in the computation of any other observed element. Therefore the simulator can assign to w_h a random and independent value.

- If the attacker has only observed output shares, then we point out that by definition each of them is composed by q random bits and at most one of them can enter in the computation of each other output variable c_i. Since the adversary may has previously probed at most $q-1$ of them, there exist one random bit $r_{k,l}$ in w_h, which does not appear in the computation of any other observed value. Thus the simulator can assign to w_h a random and independent element, completing the proof.