Glitch-Resistant Masking Revisited
or Why Proofs in the Robust Probing Model are Needed

Thorben Moos[1], Amir Moradi[1],
Tobias Schneider[2], and François-Xavier Standaert[2]

[1] Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
[2] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium
[1]{firstname.lastname}@rub.de [2]{firstname.lastname}@uclouvain.be

Abstract. Implementing the masking countermeasure in hardware is a delicate task. Various solutions have been proposed for this purpose over the last years: we focus on Threshold Implementations (TIs), Domain-Oriented Masking (DOM), the Unified Masking Approach (UMA) and Generic Low Latency Masking (GLM). The latter generally come with innovative ideas to prevent physical defaults such as glitches. Yet, and in contrast to the situation in software-oriented masking, these schemes have not been formally proven at arbitrary security orders and their composability properties were left unclear. So far, only a 2-cycle implementation of the seminal masking scheme by Ishai, Sahai and Wagner has been shown secure and composable in the robust probing model – a variation of the probing model aimed to capture physical defaults such as glitches – for any number of shares. In this paper, we argue that this lack of proofs for TIs, DOM, UMA and GLM makes the interpretation of their security guarantees difficult as the number of shares increases. For this purpose, we first put forward that the higher-order variants of all these schemes are affected by (local or composability) security flaws in the (robust) probing model, due to insufficient refreshing. We then show that composability and robustness against glitches cannot be analyzed independently. We finally detail how these abstract flaws translate into concrete (experimental) attacks, and discuss the additional constraints robust probing security implies on the need of registers. Despite not systematically leading to improved complexities at low security orders, e.g., with respect to the required number of measurements for a successful attack, we argue that these weaknesses provide a case for the need of security proofs in the robust probing model at higher security orders.

1 Introduction

Masking (aka secret sharing) is one of the most popular countermeasures against side-channel attacks [11]. Evaluating its security guarantees is known to be non-trivial, especially as the number of shares and claimed security order increase. The latter is confirmed by various security flaws that have been exhibited in early proposals of higher-order masking schemes, which we organize in two categories.

First, *local flaws* correspond to cases where a masked gadget (e.g., a multiplication algorithm, a masked S-box, ...) does not deliver its security guarantees. A typical example of a local flaw is the attack against the higher-order masking scheme of Schramm and Paar [46], put forward by Coron et al. [15]. Second, *composability flaws* correspond to cases where the combination of locally secure gadgets leads to additional weaknesses. A typical example of a composability flaw is the attack against the higher-order masking scheme of Rivain and Prouff [42] (which describes locally secure gadgets), put forward by Coron et al. [16].

In order to avoid such security flaws, two main theoretical advances have been introduced in the literature. First, security proofs in the *probing model* of Ishai et al. [31] can be used to analyze the local security of a masked gadget. Second, the notions of *Non-Interference* (NI) and *Strong Non-Interference* (SNI) can be used to capture the compositional security of masked gadgets [5].

Those theoretical advances are complemented by practical ones exploiting program verification techniques. For example, the work by Barthe et al. describes a tool able to verify the security of a masked implementation up to a certain order [4]. Other works propose similar but more specialized ideas [20, 40].

Furthermore, under some assumptions of sufficiently noisy and independent leakages, security in the (abstract) probing model implies security in the (more concrete) noisy leakage model [38], as shown by Duc et al. [18]. Since (under the independence condition only), probing security also implies security in the bounded moment leakage model [6], which is frequently used to assess the concrete security order of actual implementations [45], these results suggest probing security as a useful first step to verify for any masked implementation.

Concretely, this first (abstract) evaluation step of masked implementations can typically rely on two approaches. Either security is claimed for arbitrary orders. In this case, a hand-made proof is required for the masked gadgets considered (and this proof has to guarantee composability in case the target implementation is a full cipher mixing many gadgets). Or security is claimed up to a given order that can be exhaustively analyzed thanks to program verification techniques. To a large extent, all recent results in (what we will denote as) software-oriented masking (to be understood as the masking schemes primarily designed for software implementations) follow one of these approaches, leading to easy-to-interpret guarantees. We cite [14, 7] and [6] as recent examples.

Hardware-oriented masking. In parallel to software-oriented masking, significant efforts have also been devoted to the design of masking gadgets for hardware implementations. In this context, one important additional issue is that physical defaults such as glitches can easily contradict the independence assumption required for secure masking [32]. Since this break of the independence assumption directly leads to devastating attacks [33], the literature then focused on the design of gadgets with better resistance to glitches. A popular illustration of such progresses is the introduction of *Threshold Implementations* (TIs), which showed that a simple algorithmic property (namely, the *non-completeness* prop-

erty) is sufficient to mitigate the glitch issue [36]. The latter was then successfully applied to many first-order threshold implementations (e.g., [37, 34, 9]).

Yet, as in the software case, the generalization from first-order TIs to higher-order TIs proved to be challenging. For example, the first attempt to build a higher-order TI in [8] was not successful because of a lack of refreshing leading to a composability flaw [39, 41]. Since then, various papers proposed innovative ways to implement higher-order masking in hardware, mixing engineering intuitions and elements borrowed from the software-oriented masking literature. We mention for example the Consolidated Masking (CMS) scheme in [41, 12], the Domain-Oriented Masking (DOM) in [29, 29, 30], the Unified Masking Approach (UMA) in [27, 28] and the Generic Low Latency Masking (GLM) in [26].

An interpretation issue. Reading these papers, it is tempting to conclude that they provide solutions for higher-order secure (glitch-resistant) masking gadgets, with a certain degree of composability. Indeed, most of them use the number of shares as a parameter of their designs and provide performance evaluations for full ciphers (which suggests composability is part of the authors' concerns). Yet, contrary to the usual situation in software-oriented masking, none of these proposals comes with a probing security proof at arbitrary order. For example, the CMS implementation in [12] investigates the concrete security of a second-order masked AES design (using the tools of [45]), the DOM implementations in [30] investigate the concrete security of first- and second-order masked AES designs, the UMA implementations in [27] investigate the concrete security of dth-order masked `Ascon` designs for $d = 1, 2, 3$, and, analogous to the GLM scheme, analyze the side-channel resistance of the `Ascon` S-box for $d = 1, 2, 3$ using the formal verification tool introduced in [10]. Hence, these examples raise the question whether the CMS, DOM, UMA and GLM algorithms (or their generalization) directly lead to higher-order secure implementations, or whether the lack of proofs for these designs leaves room for weaknesses in the higher-order cases, that require attention/tweaks? We show the second statement is correct by:

- exhibiting a local flaw in the (generalized) CMS multiplication of [12],
- exhibiting a local flaw in the DOM-*dep* multiplication of [29],
- exhibiting a composability flaw in the UMA of [27],
- showing that these flaws are reproduced in the GLM of [26].

We note that these flaws do not invalidate the innovative ideas in these schemes: they only show that when moving to higher security orders, the engineering intuition that led to the successful design of gadgets secure at low orders benefits from a more formal analysis. In this regard, our only claim is that this collection of examples illustrates the difficulty to interpret the higher-order security guarantees provided by CMS, DOM, UMA and GLM, and that, without the appropriate tweaks, these schemes cannot be extended beyond the contexts in which they were exhaustively analyzed. The latter leads to an error-prone situation for engineers willing to implement higher-order (glitch-resistant) masking in hardware. We use it to argue that as in the software case, *harware-oriented*

masking schemes should either restrict claims to the specific orders that have been exhaustively investigated, or provide a hand-made proof for arbitrary orders.

The need of robust probing security. The previous issues can be solved by integrating the additional information provided by physical defaults such as glitches in the probing model, as recently proposed by Faust et al. [21]. This reference describes a variant of the multiplication algorithm of Ishai et al. in [31] and proved its security in the *robust probing model* for this purpose. To the best of our knowledge, this is the first (and so far only) multiplication algorithm proven secure and composable at arbitrary orders in the presence of glitches.

In this respect, a final question is whether dealing with and analyzing physical defaults and composability issues jointly is strictly needed? For example, is it enough to combine a glitch-resistant (probing) secure TI gadget with a strong (e.g., SNI) refresh and well-placed registers to obtain a gadget that is composable in the presence of glitches? We answer the question negatively by providing a counterexample to this approach, hence proving that analyzing the glitch-resistance and composability of masked gadgets independently is not enough, which provides a strong case for the need of the robust SNI abstraction.

Experimental confirmation. We finally investigate the concrete exploitability of the (local and composability) flaws exhibited in higher-order TIs, DOM, UMA and GLM based on an FPGA case study. We conclude that these flaws can be observed in practice (sometimes leading to lower attack complexities than the generic attack at order $(d + 1)$, sometimes not for the – low – security orders we consider experimentally). In any case, we argue that the presence of these flaws is problematic since it prevents the extrapolation of the security guarantees of these schemes to higher orders. We also use our experiments to discuss the additional constraints that the robust probing security abstraction implies on the placement of registers within masked hardware implementations.

2 Background

We first recall security definitions that are relevant to our discussions.

The t-probing model was introduced by Ishai et al. in [31] in order to prove the security of masked implementations. It assumes an adversary who can probe a limited number t of wires inside the target implementation. Probing security requires that the observation of these wires does not allow the adversary to learn sensitive information. Formally, this implies to define the target implementation as a circuit C (e.g., modeled as a graph) or as a sequence of leaking operations. Due to its simplicity, probing security was popular to analyze the first proposals of higher-order masking schemes. We next use the following definition:

Definition 1 (t-probing security [31, 42]). *A circuit C is t-probing secure iff every t-tuple of its intermediate variables is independent of any sensitive variable.*

In the case of block ciphers, sensitive variables typically correspond to partial computation results depending on the plaintext and key [15]. Concretely, probing security can be achieved by splitting every sensitive variable k in at least $t+1$ values (usually called *shares*) so that their sum gives k, performing all computations on these shares, and re-combining the final result only.

One limitation of this definition of probing security is that it does not provide any guarantee of composability. Thus, while it is sufficient for the direct analysis of a complete circuit C, it does not allow the separate analysis of smaller circuit gadgets G. The latter typically comes in handy as the size of the circuits and the number of shares grows, making the direct analysis unpractical. More precisely, when gadgets are composed to produce 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 adversary. The following definitions of NI and SNI have been introduced by Barthe et al. for this purpose:

Definition 2 ($t-$Non-Interference [5]). *A circuit gadget G is* $t-$Non-Interfering *($t-$NI) iff for any set of t_1 probes on its intermediate values and every set of t_2 probes on its output shares with $t_1 + t_2 \leq t$, the totality of the probes can be simulated with only $t_1 + t_2$ shares of each input.*

Definition 3 ($t-$Strong Non-Interference [5]). *A circuit gadget G is* $t-$Strong Non-Interfering *(t-SNI) iff for any set of t_1 probes on its intermediate values and every set of t_2 probes on its output shares with $t_1+t_2 \leq t$, the totality of the probes can be simulated with t_1 shares of each input.*

As illustrated in [7] for the case of the AES S-box, combining NI and SNI gadgets enables compositional reasoning for arbitrary circuits. In order to satisfy these definitions, one has to build 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 $t_1 + t_2$ shares (in the NI case) or t_1 shares (in the SNI case) of each secret input). The simulation is successful if no distinguisher can tell apart the simulation from the adversary's view. In this respect, one important technical clarification is that in the definitions of Barthe et al., the distinguisher can access the joint distribution of the (simulated) probes and input shares (which is strictly necessary for the compositional proofs). As a result, SNI is a stronger notion than NI, which it itself a stronger notion than probing security.

We finally introduce the robust probing model with the following example of a TI gadget implementing a Toffoli gate (i.e., $c = (x \odot y) \oplus z$):

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

with the subscripts of the x, y, z, c variables indicating the shares' indices.

Based on this example, first assume that the gadget is implemented in a single cycle and in a glitch-free manner. In this case, the adversary can only probe the

input shares x_i, y_i, z_i and output shares c_i, but not the intermediate values. That is, thanks to the glitch-free hardware, the output shares are produced from the input shares without any transient state that would leak additional information. It is easy to see that such an (ideal) gadget is 2-probing secure.

In practice though, most hardware implementations are not glitch-free and transient values leak additional information about the internal values [32, 33]. The latter can be captured by the robust probing model which assumes that probes are "*extended*" so that when applied to any wire of a combinatorial circuit, the adversary can observe all the inputs this wire depends on [21].[3] In this case, the adversary can choose between probing output values stored in registers (which cannot be extended) and internal values before they are stored in registers (which can be extended). For example, in the gadget of Equation 1 implemented in a single cycle, an extended probe on the internal value c_1 would give access to x_2, x_3, y_2, y_3 and z_2 to the adversary. Interestingly, thanks to the non-completeness property (which requires that every combinatorial gadget excludes at least one share of any sensitive variable), this TI gadget remains 1-probing secure. We will refer to such implementations as *glitch-resistant*, reflecting the fact that they can cope with glitches by design (in contrast to *glitch-free* hardware which requires the problem to be solved at the micro-electronic level).

Additional remarks. As discussed in [21], the gadget of Equation 1 is neither NI nor SNI, even if it is implemented in glitch-free hardware. This is because it does not use any fresh internal randomness that can help the simulation. Remember that the distinguisher has access to the joint distribution of the (simulated) probes and input shares, so the simulator cannot leverage the shares of z for refreshing the shares of the $x \odot y$ product, as TIs typically exploit.

Note also that the possibility for the adversary to choose between an output probe and an internal probe for the output values stored in a register (e.g., the c_i's in Equation 1) is essential to capture composability with glitches. Indeed, only the (stable, non-extended) output probes are included in the t_2 probes that are excluded from the input shares' count in the SNI definition.

3 Consolidated Masking Scheme (CMS)

At CRYPTO 2015, Reparaz et al. presented links between the established ISW multiplication and the concept of TIs [41]. They introduced an approach to realize masking in hardware with only $d + 1$ shares, which we denote in the following as CMS.[4] This scheme was later applied in [12] to implement a masked AES with only $d+1$ shares. In this section, we first recall the CMS multiplication as introduced in [41] and substantiated in [12]. Then we present a third-order flaw based on the particular (ring) refreshing strategy of the scheme.

[3] We only describe the glitch-extended probes that will be relevant to our discussions. Extensions corresponding to other physical defaults are discussed in [21].

[4] Earlier proposals of higher-order TIs usually needed more shares [8].

3.1 Multiplication with Independent Inputs

While CMS can be applied to many different operations, we restrict our analysis to the common multiplication of two inputs. To this end, we rely on the description given in [12] for a two-input AND gate. Their approach is based on the consecutive application of multiple layers to the input shares: non-linear layer $\mathcal{N}$, linear layer $\mathcal{L}$, refresh layer $\mathcal{R}$, synchronization layer $\mathcal{S}$ (i.e., register stage), compression layer $\mathcal{C}$. In the case of $d+1$ shares masking, the linear layer $\mathcal{L}$ is skipped for the multiplication to ensure that each term given to the refresh layer $\mathcal{R}$ contains only one share of each input variable. This refresh is done in a circular manner (cf. Figure 1) requiring $(d+1)^2$ random elements. In the compression phase, the refreshed values are summed up in order to achieve $d+1$ shares for the output. The authors of [12] provide concrete installations only up to order $d = 2$. Based on these descriptions, we generalize their approach for an arbitrary number of shares with the algorithmic description in Algorithm 1.

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

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

```
for i = 1 to n do
  c_i = 0
  for j = 1 to n do
    c_i = c_i + (a_i · b_j + r_{(i−1)·d+j mod n²} + r_{(i−1)·d+j+1 mod n²});
  end for
end for
```

Note that this algorithm is only a functional representation of the scheme and lacks the concrete distinction into layers which is the basis of the CMS concept. Therefore, Figure 1 depicts the layer-wise architecture for the $d = 3$ case based on the notations of Algorithm 1. Note also that the implementation of Figure 1 does not satisfy the non-completeness property of standard TIs (which is similar to Figure 1 in [12] that we extend in the natural manner). We will discuss the impact of tweaking the design to make it non-complete later in the section.

3.2 A Third-Order Flaw

In the following, we demonstrate that the CMS multiplication as given in Algorithm 1 and Figure 1 does not provide the claimed security guarantees for arbitrary d. Our flaw stems from the combination of the circular refresh strategy $\mathcal{R}$ with the specific compression layer $\mathcal{C}$. In particular, summing up all terms $a_i \cdot b_j + r_k$ for a specific value of i cancels out many of the random terms from the refresh layer. While this is not problematic for the orders $d = 1, 2$ considered in [12], it leads to a trivial attack with only three probes for $d \geq 3$.

Concretely, after compression each output share c_i can be written as:

$$c_i = a_i \cdot b + r_{(i-1)\cdot d+1} + r_{i \cdot d+1}. \tag{2}$$

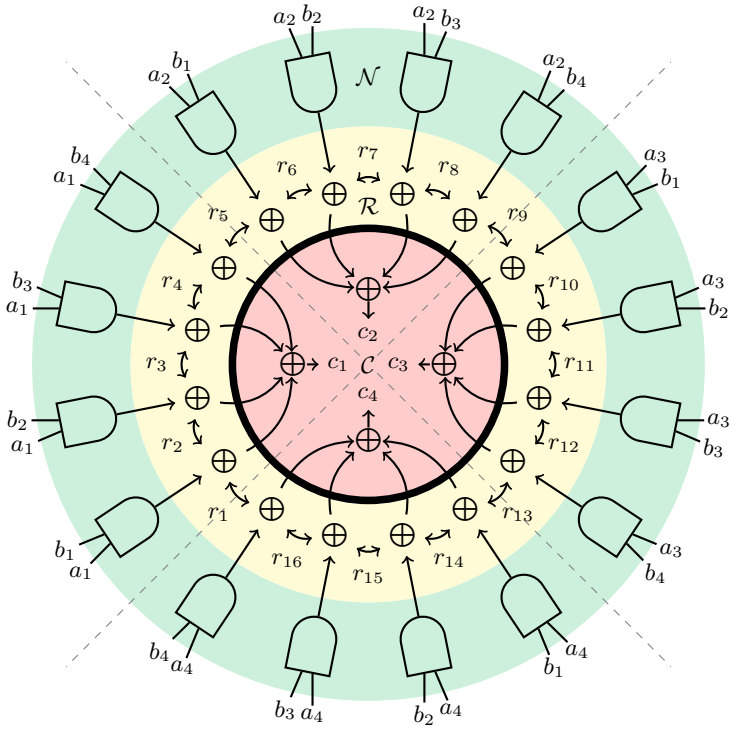


Fig. 1. Architecture of CMS multiplication extending the proposal in [12] to $d = 3$, consisting of the (green) non-linear layer $\mathcal{N}$, the (yellow) refresh layer $\mathcal{R}$, the (black) synchronization (registers) layer $\mathcal{S}$, and the (red) compression layer $\mathcal{C}$.

Therefore, by probing:

$$P_1 = c_i, \tag{3}$$
$$P_2 = r_{(i-1)\cdot d+1}, \tag{4}$$
$$P_3 = r_{i\cdot d+1}, \tag{5}$$

the adversary can observe a joint distribution (P_1, P_2, P_3) which depends on $a_i \cdot b$. While a_i is still a random value independent of a, it does not suffice as a mask for b given the "zero bias" of multiplicative masking schemes [23] (e.g., for the binary case, $a_i \cdot b = 1$ implies $b = 1$). Thus, the joint distribution leaks about the sensitive value b invalidating the security of the multiplication scheme.

Example 1 ($d = 3$). For better understanding, we demonstrate an attack on the simplest case with $d = 3$ which is shown in Figure 1. The probes are placed according to the aforementioned guidelines as follows:

$$P_1 = c_1 = a_1 \cdot b + r_1 + r_5, \tag{6}$$
$$P_2 = r_1, \tag{7}$$
$$P_3 = r_5. \tag{8}$$

The histograms of the joint distribution of (P_1, P_2, P_3) for fixed $b \in \mathbb{F}_2$ are given in Table 1. It is noticeable that they differ based on the value of b. Therefore, an adversary could distinguish the value of b with only three regular probes for any order $d \geq 3$ which is less than the claimed order of security.

Table 1. Histogram of the joint distribution of (P_1, P_2, P_3) for $b = 0$ and $b = 1$.

(P_1, P_2, P_3)	0	1	2	3	4	5	6	7
$b = 0$	2	0	0	2	0	2	2	0
$b = 1$	1	1	1	1	1	1	1	1

3.3 Discussion

We first insist that the previous attack does not contradict the claims in [12] since (in the core of the paper) the authors make clear that their analysis is limited to the case $d = 2$, i.e., with 3 shares. Thus, our only claim is again that the title of the paper can be misleading, since the natural extension of the proposed algorithms does not lead to higher-order secure gadgets as one could expect. It leaves as an open problem to find efficient solutions to fix this flaw (some proposals can be found in Thomas De Cnudde's PhD dissertation).

We also observe that considering a non-complete compression layer, despite not necessary from the glitches viewpoint (since a register stage prevents the propagation of the glitches before the compression in Figure 1), would actually make the attack slightly more difficult. For example, imagine that no c_i value in Figure 1 depends on the 4 shares of a_i or b_i: then an attack would only succeed by probing multiple c_i's together with the r_i's at their "borders" (postponing the apparition of the security flaw to higher orders). Denoting the number of c_i's to probe with n, the generalized attack will work with at most $3n$ probes, and possibly less if the probed c_i's share a border. It is however interesting that the non-completeness property turns out to be useful for composability purposes. We leave the exploitation of this observation (e.g., to design secure and efficient implementations at low orders) as an interesting scope for further research.

We finally note that, as the ring refreshing in [12] is not SNI, the generalization of the S-box design in this reference to higher-orders also suffers from composability flaws similar to the ones of the UMA and GLM schemes.

4 Domain-Oriented Masking (DOM)

Domain-Oriented Masking (DOM) was proposed in 2016 by Gross et al. with the goal to enable secure masking in hardware with only $d + 1$ shares [29, 30]. The main contribution is a masked multiplier initially denoted as DOM-*indep*. Its randomness distribution is closely related to the ISW multiplication and it is therefore probing secure given independently shared inputs. However, for

the multiplication of dependently shared inputs, Gross et al. include another alternative multiplication scheme called DOM-*dep* in their eprint version [29]. It is used in some of their proposed designs to improve efficiency. In this section, we first recall the specification of DOM-*dep* and then demonstrate a $(\lceil \frac{d}{2} \rceil + 1)$th-order flaw for $d \geq 2$, contradicting the DOM security claims.

4.1 Multiplication with Dependent Inputs

A straightforward way to extend DOM-*indep* for allowing dependently shared inputs is to SNI refresh one of the inputs [25]. This provides security but comes with significant costs in randomness, area, and latency. DOM-*dep* was proposed as a more efficient alternative which does not require re-sharing. Instead, a blinding value z is introduced to multiply the inputs a and b as:

$$c = a \cdot b = a \cdot (b + z) + (a \cdot z). \tag{9}$$

Since z is a random value, the authors proposed an efficient way to compute $a \cdot (b + z)$ by first decoding $(b + z)$ and then multiplying the result with each share of a. Therefore, DOM-*dep* requires only one full DOM-*indep* multiplication compared to two for the previously outlined (straightforward) approach. The generic scheme for any order is given in Algorithm 2 based on the descriptions provided in [29], where the $\mathbf{x}$ notation is used to represent vectors of shares.

Algorithm 2 DOM-*dep* multiplication algorithm with $n \geq 2$ shares.

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

```
for i = 1 to n do
    z_i ←$ F_q
    x_i ← (b_i + z_i)
end for
x = Decode(x)
c =DOM-indep (a, z)
for i = 1 to n do
    c_i ← c_i + (a_i · x)
end for
```

We want to note that (as for the CMS multiplication), Algorithm 2 is only a functional representation of DOM-*dep* and does not show the concurrent operations and register stages required for a hardware design. Instead, these are depicted in Figure 2 (based on Figure 4 of [29]) for the special case of $d = 2$.

4.2 A $(\lceil \frac{d}{2} \rceil + 1)$th-Order Flaw

In the following, we demonstrate that DOM-*dep* as given in Algorithm 2 and Figure 2 does not provide the claimed security guarantees for arbitrary d. For simplicity, we assume that the input encodings of a and b are identical (i.e., $a_i = b_i,\ 1 \leq i \leq n$). The main problem of DOM-*dep* stems from the $\mathsf{Decode}(\mathbf{x})$

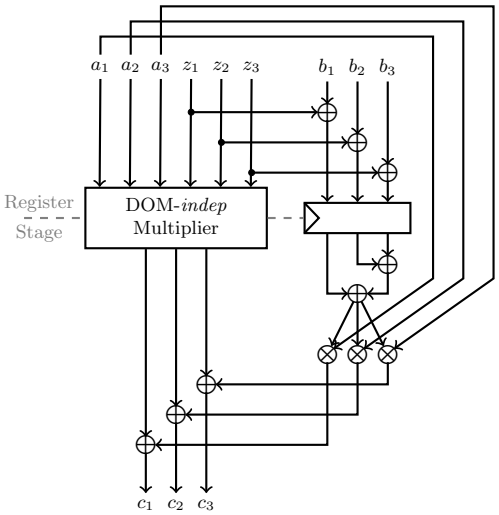


Fig. 2. Architecture of DOM-*dep* for $d = 2$.

operation. In an idealized world (corresponding to unrealistic glitch-free hardware discussed at the end of Section 2), this operation would be performed without leaking information on intermediate values, and an adversary would not be able to probe any intermediate sum of the decoding. Therefore, with one probe on $\mathsf{Decode}(\mathbf{x})$ the adversary either receives (a) one of the input shares $b_i + z_i$, or (b) the output value $b + z$. Both cases cannot be used to construct an attack, since for (a) it is similar as probing an intermediate value in the secure DOM-*indep* multiplier (assuming $a_i = b_i$), and for (b) z is a true random value which cannot be probed directly (only its shares z_i). Hence, DOM-*dep* might be secure in this idealized model (we leave the proof as an open problem).

However, as a hardware-oriented masking scheme, DOM is aimed to be glitch-resistant and therefore to maintain security even in the more practical robust probing model. In this case, the adversary has access to more powerful probes which enable her to extract sensitive information from DOM-*dep*. For the operation $\mathsf{Decode}(\mathbf{x})$, probing the output value $b + z$ provides information about all input sums $b_i + z_i$, since there are no registers to prevent glitches. This alone does not suffice for an attack, because the shares b_i are still masked by the z_i's. Nevertheless, by also probing in the DOM-*indep* multiplication of a and z, it is possible to break the scheme. In particular, the adversary first accesses:

$$\{b_1 + z_1, b_2 + z_2, \ldots, b_{n/2} + z_{n/2}\}, \tag{10}$$

with only one probe on the output of $\mathsf{Decode}(\mathbf{x})$. Then $\frac{n}{2}$ probes are placed in the cross-product terms of DOM-*indep* which consist of some of the already probed

random terms z_i and the remaining unprobed input shares a_i:

$$\{a_{n/2+1} \cdot z_1, a_{n/2+2} \cdot z_2, \ldots, a_n \cdot z_{n/2}\}. \tag{11}$$

The distribution of these $(\lceil \frac{d}{2} \rceil + 1)$ variables depends on the value of a. For odd values of n, another probe might be necessary to probe a_n when considering $\lfloor \frac{n}{2} \rfloor$ cross-product terms. However, since there is no register between $\mathsf{Decode}(\mathbf{x})$ and the subsequent share-wise multiplication, the adversary can simply place the extended probe on the computation $x \cdot a_n$. This provides the same input sums as before with the added benefit of leaking a_n. Therefore, DOM-*dep* does not provide the desired robust probing security for $d \geq 2$.

Example 2 ($d = 2$). We demonstrate an attack on the simplest case with $d = 2$ as shown in Figure 2. The probes are placed on the output of the computation of $x \cdot a_3$ and on one term of the cross-product according to the aforementioned guidelines. We choose to target the intermediate variable $(a_1 + z_1) \cdot a_3$ accessed by the extended probe. This leads to following probed variables:

$$P_1 = (a_1 + z_1) \cdot a_3, \tag{12}$$

$$P_2 = a_2 \cdot z_1, \tag{13}$$

The histograms of the joint distribution of (P_1, P_2) for fixed $a \in \mathbb{F}_2$ are given in Table 2. It is noticeable that they differ based on the value of a. Therefore, an adversary could distinguish the value of a with only one extended and one regular probe, which is less than the claimed order of security.

Table 2. Histogram of the joint distribution of (P_1, P_2) for $a = 0$ and $a = 1$.

(P_1, P_2)	0	1	2	3
$a = 0$	5	1	1	1
$a = 1$	4	2	2	0

4.3 Discussion

As previously mentioned, there is an easy (but costly) fix to this attack by using an SNI refreshing gadget before each multiplication of dependently shared values. By contrast, the introduction of register stages in $\mathsf{Decode}(\mathbf{x})$ does not solve the problem since any intermediate sum containing more than one share of b could be used for an attack with less than $d+1$ probes for $d \geq 3$. This for example implies that even a software implementation of DOM-*dep* without glitches would be vulnerable to the presented flaw. It recalls the gradation between the (minimum) amount of information leaked by an ideal (glitch-free) hardware implementation, the (intermediate) amount of information leaked by a standard software implementation (where some intermediate variables are leaked) and the worst-case amount of information leaked by a glitchy hardware one.

5 Unified Masking Approach (UMA)

Following the concept of DOM, Gross and Mangard proposed a more randomness-efficient hardware multiplication scheme denoted as Unified Masking Approach (UMA) in [27, 28]. It essentially combines the software-oriented parallel masking algorithm of Barthe et al. [6] with the randomness optimizations of Belaid et al. [7] in order to achieve (so far the most) randomness-efficient masked multiplication in hardware. For certain orders d, UMA even outperforms known software solutions. In contrast to [6, 7], the authors of UMA do not state any limitation regarding the composability of their multiplication scheme. In the following, we first shortly recall the UMA concept and then highlight composability issues.

5.1 A (not so) Universal Multiplication

The basis of UMA is the multiplication algorithm from Barthe et al. [6]. It is extended with optimizations from Belaïd et al. [7] and DOM [30] for certain values of d to reduce the randomness complexity even further. Therefore, the generic solution given in Algorithm 3 ($\mathbf{a}_{+i}$ denotes a rotation of the share vector $\mathbf{a}$ by i positions) includes a distinction of different cases for d to account for these optimizations. The multiplication is split into five blocks: *Inner-Domain*, *Complete*, *Pseudo-Complete*, *Half-Complete*, and *Incomplete*.

- *Inner-Domain*: In this block, the inputs are multiplied share-wise. Since this operation is implemented without mixing the input domains (assuming independent inputs), it does not require the inclusion of register stages.
- *Complete*: With the *Pseudo-Complete* block, the *Complete* block implements the masked multiplication according to Barthe et al.'s algorithm. Each loop iteration is performed in parallel to each other, but a register stage is required after every addition to ensure security, resulting in a delay of five cycles.
- *Pseudo-Complete*: This block processes the remaining terms of Barthe et al.'s algorithm. It requires register stages after every addition, but the delay is four cycles since it contains one less addition than the *Complete* block.
- *Half-Complete*: This block contains a further case distinction for $d = 2$. In this scenario, the multiplication is implemented according to Belaïd et al.'s optimal algorithm and requires three register stages. For the other cases, the authors rely on DOM which only adds a delay of one cycle, because the terms $\mathbf{r}^l + \mathbf{a} \cdot \mathbf{b}_{+2l+1}$ and $\mathbf{r}^l_{+2l+2} + \mathbf{a} \cdot \mathbf{b}_{+2l+2}$ are computed in parallel.
- *Incomplete*: Similar to the previous block, the *Incomplete* terms are computed according to DOM and require the inclusion of one register stage.

Depending on the order d, these blocks are instantiated and their outputs are combined as depicted in Figure 5 from [27] (cf. Figure 3). *Inner-Domain* is always implemented and connected to $\lfloor \frac{d}{4} \rfloor$ *Complete* blocks, and optionally to one *Pseudo-Complete*, *Half-Complete*, or *Incomplete* block. Additional registers or control logic might be necessary to ensure synchronization between the different blocks given the difference in delay (which we will discuss in Section 8.2).

Algorithm 3 UMA multiplication algorithm with $n \geq 1$ shares. [27]

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

$l = \lfloor \frac{d}{4} \rfloor$

$\mathbf{c} = \mathbf{a} \cdot \mathbf{b}$ — *Inner-Domain*

for $i = 0 < \lfloor \frac{d}{4} \rfloor$ **do**
  $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{r}^i + \mathbf{a} \cdot \mathbf{b}_{+2i+1} + \mathbf{a}_{+2i+1} \cdot \mathbf{b} + \mathbf{r}^i_{+1} + \mathbf{a} \cdot \mathbf{b}_{+2i+2} + \mathbf{a}_{+2i+2} \cdot \mathbf{b}$ — *Complete*
end for

if $d \equiv 3 \mod 4$ **then**
  $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{r}^l + \mathbf{a} \cdot \mathbf{b}_{+2l+1} + \mathbf{a}_{+2l+1} \cdot \mathbf{b} + \mathbf{r}^l_{+1} + \mathbf{a} \cdot \mathbf{b}_{+2l+2}$ — *Pseudo-Complete*
end if

if $d \equiv 2 \mod 4$ **then**
  if $d = 2$ **then**
    $\mathbf{z} = \{r^l_1, r^l_2, r^l_1 + r^l_2\}$
    $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{z} + \mathbf{a} \cdot \mathbf{b}_{+2l+1} + \mathbf{a}_{+2l+1} \cdot \mathbf{b}$
  else
    $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{r}^l + \mathbf{a} \cdot \mathbf{b}_{+2l+1} + \mathbf{r}^l_{+2l+2} + \mathbf{a} \cdot \mathbf{b}_{+2l+2}$
  end if — *Half-Complete*
end if

if $d \equiv 1 \mod 4$ **then**
  $\mathbf{z} = \{r^l, r^l\}$
  $\mathbf{c} \leftarrow \mathbf{c} + \mathbf{z} + \mathbf{a} \cdot \mathbf{b}_{+2l+1}$ — *Incomplete*
end if

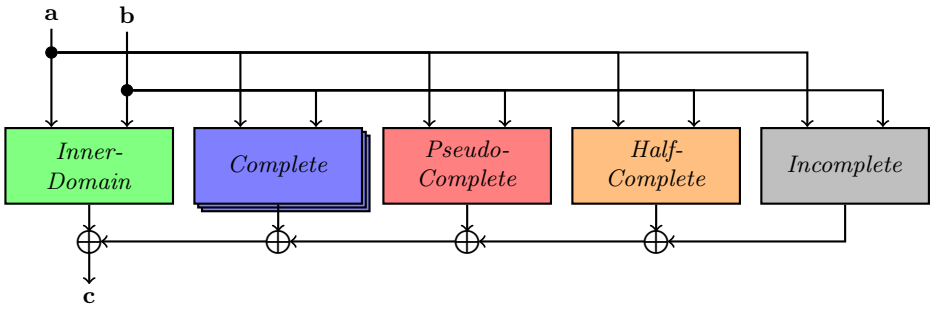


Fig. 3. Connection of the UMA blocks [27].

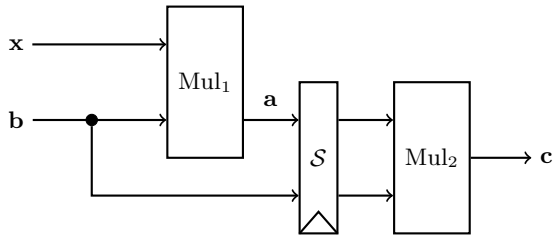


Fig. 4. Composition of two UMA multiplications.

5.2 A Systematic Composability Flaw

Belaïd et al. and Barthe et al. analyzed the security of their multiplication algorithms with formal proofs and verification in regard to both probing security and SNI. Therefore, they were able to provide concrete assertions regarding the composability of their schemes. In particular, it was found that the randomness-optimized multiplications in [7] are not SNI and that the parallel multiplications in [6] are only d-SNI until $d = 2$ (its composition with simple refreshing gadgets is d-SNI for larger d's). Therefore, a designer has to take great care where to instantiate them without violating the security of the whole design.

By contrast, for UMA the authors do not examine their multiplication regarding this criterion. Their case study uses the UMA multiplication without discussing composability explicitly. Therefore, a non-expert reader might be compelled to believe that the unified masking approach is indeed universal and can be used at any point of any masked design. In the following, we show that an exemplary composition of two UMA multiplications does not compose well.

Following the typical pattern of composability flaws put forward by Coron et al. in [16], our example is depicted in Figure 4. The input encoding $\mathbf{b}$ is initially refreshed by multiplying it with a random encoding $\mathbf{x}$. Then the refreshed output $\mathbf{a}$ is multiplied with the original $\mathbf{b}$ resulting in $\mathbf{c}$. This structure is commonly used when an input is multiplied with a linear transformation of itself, e.g., for the inversion in $GF(2^8)$ [7]. For simplicity, we omitted the linear transformation from our construction. In addition to the register stage between the multiplications, there are multiple registers inside Mul_1 and Mul_2 depending on the order. For now, we assume that the registers are enabled in a sequential fashion, e.g., the second stage is enabled only after the first one. Given a freely-composable multiplication, e.g., ISW [31] or DOM [30], this structure should provide d-probing security. However, for UMA this is not true for orders $d > 1$ as we demonstrate by attacking the composition with d probes. Since the UMA multiplication differs in structure depending on the order, we look at multiple cases separately.

Example 3 ($d = 2$). Firstly, we consider Belaïd et al.'s optimized multiplication for $d = 2$. In our structure, the second multiplication Mul_2 can be written as:

$$c_1 = a_1 \cdot b_1 + r_1^2 + a_1 \cdot b_2 + a_2 \cdot b_1, \tag{14}$$

$$c_2 = a_2 \cdot b_2 + r_2^2 + a_2 \cdot b_3 + a_3 \cdot b_2, \tag{15}$$
$$c_3 = a_3 \cdot b_3 + r_1^2 + r_2^2 + a_3 \cdot b_1 + a_1 \cdot b_3, \tag{16}$$

where $\{r_1^2, r_2^2\}$ denotes the randomness used for this multiplication (resp., $\{r_1^1, r_2^1\}$ for Mul_1). One possibility to attack b consists in probing a random element in Mul_1 and a cross-product term in Mul_2 as:

$$P_1 = r_1^1, \tag{17}$$
$$P_2 = a_1 \cdot b_3 = (x_1 \cdot b_1 + r_1^1 + x_1 \cdot b_2 + x_2 \cdot b_1) \cdot b_3. \tag{18}$$

Since the joint distribution of (P_1, P_2) (reproduced in Table 3) depends on the value of b, it can be used to distinguish the sensitive variable with only two probes which contradicts the security claims of UMA.

Table 3. Histogram of the joint distribution of (P_1, P_2) for $b = 0$ and $b = 1$.

(P_1, P_2)	0	1	2	3
$b = 0$	12	12	4	4
$b = 1$	14	10	2	6

This attack generalizes to higher orders. For simplicity, we first discuss the flaw for $d \equiv 0 \mod 4$, i.e., when the multiplication consists of only the *Inner-Domain* and *Complete* blocks. The first output share of Mul_1 is of the form:

$$a_1 = x_1 \cdot b_1 + r_1^1 + x_1 \cdot b_2 + x_2 \cdot b_1 + r_2^1 + x_1 \cdot b_3 + x_3 \cdot b_1, \tag{19}$$
$$+ r_3^1 + x_1 \cdot b_4 + x_4 \cdot b_1 + r_4^1 + x_1 \cdot b_5 + x_5 \cdot b_1, \tag{20}$$
$$+ \cdots \tag{21}$$
$$+ r_{\frac{d}{2}-1}^1 + x_1 \cdot b_{\frac{d}{2}} + x_{\frac{d}{2}} \cdot b_1 + r_{\frac{d}{2}}^1 + x_1 \cdot b_{\frac{d}{2}+1} + x_{\frac{d}{2}+1} \cdot b_1. \tag{22}$$

It contains $\frac{d}{2} + 1$ shares of each input encoding which are masked by $\frac{d}{2}$ random elements. Given that this output share is one of the inputs of Mul_2, it is multiplied with every share of $\mathbf{b}$. In particular, with $b_{\frac{d}{2}+2}$, i.e., a share that is not contained in a_1. By putting $\frac{d}{2}$ probes in Mul_1 and one probe in Mul_2 as:

$$P_1 = r_1^1, \tag{23}$$
$$P_2 = r_2^1, \tag{24}$$
$$\cdots \tag{25}$$
$$P_{\frac{d}{2}} = r_{\frac{d}{2}}^1, \tag{26}$$
$$P_{\frac{d}{2}+1} = a_1 \cdot b_{\frac{d}{2}+2}, \tag{27}$$

the adversary can access a joint distribution $(P_1, \ldots, P_{\frac{d}{2}+1})$ which depends on $\frac{d}{2} + 2$ shares of $\mathbf{b}$. Eventually, with the remaining $\frac{d}{2} - 1$ probes, the adversary

can now access the still unknown shares by observing:

$$P_{\frac{d}{2}+2} = b_{\frac{d}{2}+3}, \tag{28}$$

$$\cdots \tag{29}$$

$$P_d = b_{d+1}, \tag{30}$$

which results in a joint distribution depending on $d+1$ shares of $\mathbf{b}$ (i.e., all n shares) with only d probes which is against the universal security claim of UMA. This attack can be trivially applied to any order $d \equiv 0 \mod 4$.

For $d \equiv 3 \mod 4$ (resp., the DOM optimization for $d \equiv 2 \mod 4$), the *Pseudo-Complete* block (resp., *Half-Complete* block) adds two further shares of $\mathbf{b}$ and two random elements to the output share a_1 of Mul_1. Therefore, a similar attack can be repeated with $\frac{d}{2}+2$ probes in Mul_1. In the incomplete case (i.e, $d \equiv 1 \mod 4$), only one further share of $\mathbf{b}$ and one random element is added to a_1 and therefore a similar attack requires $\frac{d}{2}+1$ probes in Mul_1.

5.3 Discussion

Since the algorithms [6, 7] which serve as a basis for UMA are not composable at every order, the fact that composability flaws pop up in UMA is not surprising. Interestingly, such composability flaws do not as directly appear in the specific application to the `Ascon` cipher chosen by Gross and Mangard. The main reason is that the `Ascon` S-box does not directly lead to simple dependent multiplications as in Figure 4 and composability flaws may only appear for larger d's and require to combine the shares of several rounds. So as mentioned in introduction, the main problem of [27] is its interpretation. On the one hand, the gadgets used in UMA are clearly not universally composable. On the other hand, exhaustive analysis for full circuits at high security orders is rapidly computationally hard [20, 4, 40, 26]. Admittedly, it may very well be that using SNI gadgets in this context is an overkill and that the biases caused by the lack of composability remain hard to exploit given the noise levels considered in concrete implementations until quite large security orders (as per an argument in the lines of [19], Section 4.2), or even that additional refreshings are not needed for this particular circuit. The tools introduced in [17] could be one option for the evaluation of this issue, which we leave as an interesting scope for further research.

6 Generic Low-Latency Masking (GLM)

Low latency is an optimization goal which has only been recently examined in the context of masking and side-channel analysis. Some specific investigations have been targeting the block ciphers `Prince` and `Midori` [35], KECCAK [2] and the AES S-box [22]. However, the latter investigations do not provide generic solutions for arbitrary functions at arbitrary orders. In this respect, an important observation is that all the algorithms discussed in the previous sections require

a fixed delay of one or multiple register stages per multiplication. Therefore, Gross et al. proposed a Generic Low-Latency Masking (GLM) scheme in [26]. They essentially trade randomness and area for a lower latency by skipping the compression of the shares as much as possible in their designs. In the following, we first recall the concept of GLM and then briefly show the problems arising from the proposed refreshing and compression strategies.

6.1 Low-Latency Masking and Compression

The main idea of GLM is to skip the compression function inherent to the other masked multiplications. That is, instead of summing the cross-product terms in order to obtain $d + 1$ output shares, Gross et al. propose to continue the computations with the $(d + 1)^2$ uncompressed shares:

$$(a_1 \cdot b_1) \quad (a_1 \cdot b_2) \quad \dots \quad (a_n \cdot b_n). \tag{31}$$

To avoid collisions between shares (e.g., for the computation of $(a \cdot b) \cdot b$), certain inputs – and even parts of the circuit – are duplicated and independently encoded ensuring that the inputs to every non-linear function are independent. While this methodology can be applied to arbitrary functions, every non-linear operation increases the number of shares. When this number becomes prohibitive, the authors of GLM propose to use a refresh operation followed by a register stage and a compression function in order to reduce the number of shares again to $d + 1$. They recommend using the CMS refresh from [41] for this purpose.

6.2 Combining Previous Attacks

As noted in Section 3.2, the CMS refresh from [41] does not generalize to arbitrary orders. Furthermore, even with a different distribution of the cross-product terms (i.e., with non-completeness after the compression), the CMS refreshing is not SNI for $d > 2$ and it opens the door to composability flaws as discussed in Section 5. Therefore, any GLM architecture which relies on this refresh might be vulnerable to these previous attacks. Fixing this issue is not trivial as the compression layer $\mathcal{C}$ does not include a dedicated register stage which leads to further composability problems as discussed in more detail in Section 8.

6.3 Discussion

While the results in this section do not bring new technical elements, they illustrate that the interpretation issues that we mention in the introduction can easily lead to propagation of errors from one design to another, which can be avoided by formulating the algorithms and their security claims accurately. As in the previous section, we re-insist that the exploitation of a flaw may not be obvious for all designs (e.g., in the case of the `Ascon` cipher). So our only statement is that these limitations are not clearly stated in the original GLM paper and limit its claims for generality. Finding updated refresh $\mathcal{R}$ and compression $\mathcal{C}$ algorithms, which take these issues into account and enable true generality, is an interesting topic for future work as also noted by the authors of [26].

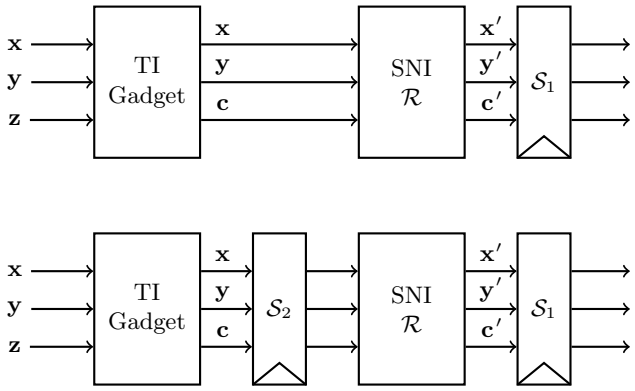


Fig. 5. Examples of non-complete and SNI gadgets that do not compose in the robust probing model (independent of the synchronization stages / registers).

7 On the Need of the Robust Probing Model

The previous (and next) sections show that probing security and composability are the result of a delicate trade-off between combinatorial computations, refreshing layers and register stages. In this respect, one natural question is whether solving these problems separately is (formally) sufficient to solve them jointly. In this section, we show that combining a glitch-resistant (non-complete and probing-secure) gadget with SNI refreshes and registers is in fact not sufficient, providing a case for the need of the robust SNI abstraction in [21].

For this purpose, we use the simple examples of Figure 5 where the TI gadget is the one given in Equation 1 (Section 2) and the SNI refresh is a 3-bit ISW refresh. First consider the top design with only one synchronization (register) stage $\mathcal{S}_1$. In this case, it is easy to see that a "glitch-extended probe" on one share of $\mathbf{c}'$ reveals all the intermediate randomness (coming from the SNI refresh) needed to compute this share of $\mathbf{c}'$ from the input shares of $\mathbf{x}, \mathbf{y}, \mathbf{z}$. Hence, this randomness cannot be used to simulate this single (extended) adversarial probe. Furthermore, adding a second register layer does not solve the problem. In this case, the adversary can directly probe $\mathbf{c}$, which cannot be simulated (since the first TI gadget only leverages the input shares to ensure probing security).

As for the previous sections, the latter examples do not imply that there are no combinations of TI gadgets, SNI refreshes and registers that are robust against glitches and composable (e.g., by using more than $d + 1$ shares). They just show that formally, the definitions of the non-completeness property and of SNI (without glitches) do not compose. As suggested in [21], Lemma 5, some form of simulatability (e.g., captured by the robust-NI property) is needed for the first (combinatorial) gadget of Figure 5. We believe such a composability is increasingly needed as the claimed security orders in hardware masking increase, making exhaustive analysis impossible for full implementations.

8 Experimental validation

In Sections 3 to 6 we have analyzed the local and compositional security of multiplication gadgets which have been proposed for glitch-resistant hardware masking and revealed that the higher-order versions of all these schemes are affected by flaws in the robust probing model. In this section we answer the question whether these flaws actually lead to exploitable leakage in real-world power measurements from hardware implementations of the corresponding schemes. After concluding positively in this regard, we discuss the severity of these leakages with respect to the practical security level of the investigated circuits. Whether or not the detected weaknesses invalidate the claims of the respective authors is open to interpretation (it in part depends on whether claims are stated in terms of security order or number of measurements to disclose the key). Yet, they effectively limit the generality of those proposed gadgets, which is an important cautionary note to designers willing to implement them. This result confirms the necessity for proofs in the robust probing model when claiming security for arbitrary orders and when aiming to protect larger non-linear functions (like substitution boxes of block ciphers) or full cryptographic primitives. Besides, while all of the exhibited flaws up to this part of the paper originate from a lack of fresh randomness, compositional security in hardware also highly depends on the correct instantiation of register stages. Additional concerns regarding DOM, GLM and UMA in this respect, and their connection to the robust probing model, are discussed in the second part of this section and in Appendix A.

Setup. In order to examine the detectability of the aforementioned flaws in practice, we conducted common fixed-versus-random t-test evaluations [24, 13] using power traces measured from an FPGA. We have used a SAKURA-G board [1] and implemented the designs explained below on its Spartan-6 FPGA operated at a clock frequency of 6 MHz. The power traces have been measured by means of a digital sampling oscilloscope at the sampling rate of 500 MS/s by monitoring the output of the embedded AC amplifier of the SAKURA-G, which amplifies the voltage drop over the resistor placed in the Vdd path of the target FPGA.

We have followed the procedure explained in [44] to collect the corresponding traces suitable for fixed-versus-random t-test analysis. In this scenario the shared input and the required fresh randomness are generated by the control FPGA. Hence, the target FPGA, whose leakage is measured, just operates on the given input and does not generate any true- or pseudo-randomness. It is noteworthy that no masking or unmasking is performed in either the control or the target FPGA. The whole communication between the PC and the measurement board as well as between both FPGAs on the board is performed in a shared manner. Using the resulting traces we conducted first- and higher-order univariate and multivariate analyses, by using the incremental formulas introduced in [44].

8.1 Exploiting the Flaws

In order to keep the following results comparable we used ordinary $GF(2^4)$ multipliers as a basis to construct each of the masked multiplication gadgets. Thus, in all designs which we analyzed the unshared operands are of 4-bit size. We present results for CMS, DOM and UMA and omit the GLM scheme to avoid redundancy, since it simply adopts the flaws from CMS. We demonstrate that in all three cases the leakage that is predicted by the exhibited flaws can be observed as multivariate leakage in the corresponding statistical moments.

CMS. As detailed in Section 3, CMS is neither probing secure nor SNI in the presence of glitches for $d > 2$, since its randomness distribution inherited from the ring structure is insufficient to deliver security for arbitrary protection orders. We analyze the construction for $d = 3$, as it is the simplest case that suffers from the third-order flaw. To be more precise we have implemented the design shown in Figure 1 and replaced all AND gates by $GF(2^4)$ multipliers. Figure 6 shows a sample trace and the results of a non-specific t-test up to the fourth statistical moment with 300 million traces. It is obvious that the design only exhibits univariate leakage in the fourth order, as it would be expected from a securely $(d+1)$-masked multiplication gadget with four shares. When moving to the multivariate analysis, however, third-order leakage can be observed with less than 100 million traces, as illustrated by Figure 7. The t-statistics curve in Figure 7(a) is obtained by calculating the second-order centralized moment of the joint distribution of each time sample together with the corresponding time sample from the consecutive clock cycle (i.e., shifted by an offset of 1 clock cycle - or 83 time samples), starting from time sample 500. Similarly, the t-statistics curve in Figure 7(c) is calculated with the third-order centralized moment of the joint distribution of each time sample with itself and the corresponding time sample in the consecutive clock cycle (again, starting from time sample 500 with 83 time samples per clock cycle). For instance, time sample 250 in Figure 7(c) corresponds to the third-order centralized statistical moment of the joint distribution of time samples 750, 750 and 833 in Figure 6(a).

It is noteworthy that in our first attempt of measuring this implementation we did not observe any (univariate or multivariate) leakage up to the third-order with up to 500 million traces, since the signal-to-noise ratio (SNR) was too small to detect the bias in the measurements associated with the joint distribution of the three probes given in Equations (6) to (8). Thus, for the experiments that led to the t-test results in Figures 6 and 7, we had to make sure that the manipulation of the probed values consumes enough power to overcome the small signal-to-noise ratio. In this regard we instantiated three extra modules connected to the 4-bit values c_1, r_1 and r_5 to amplify their corresponding leakage. Each of such extra modules (so-called leakage amplifiers) is formed by 6 times cascading a MIX module, which is a linear operation multiplying its 4-bit input to the

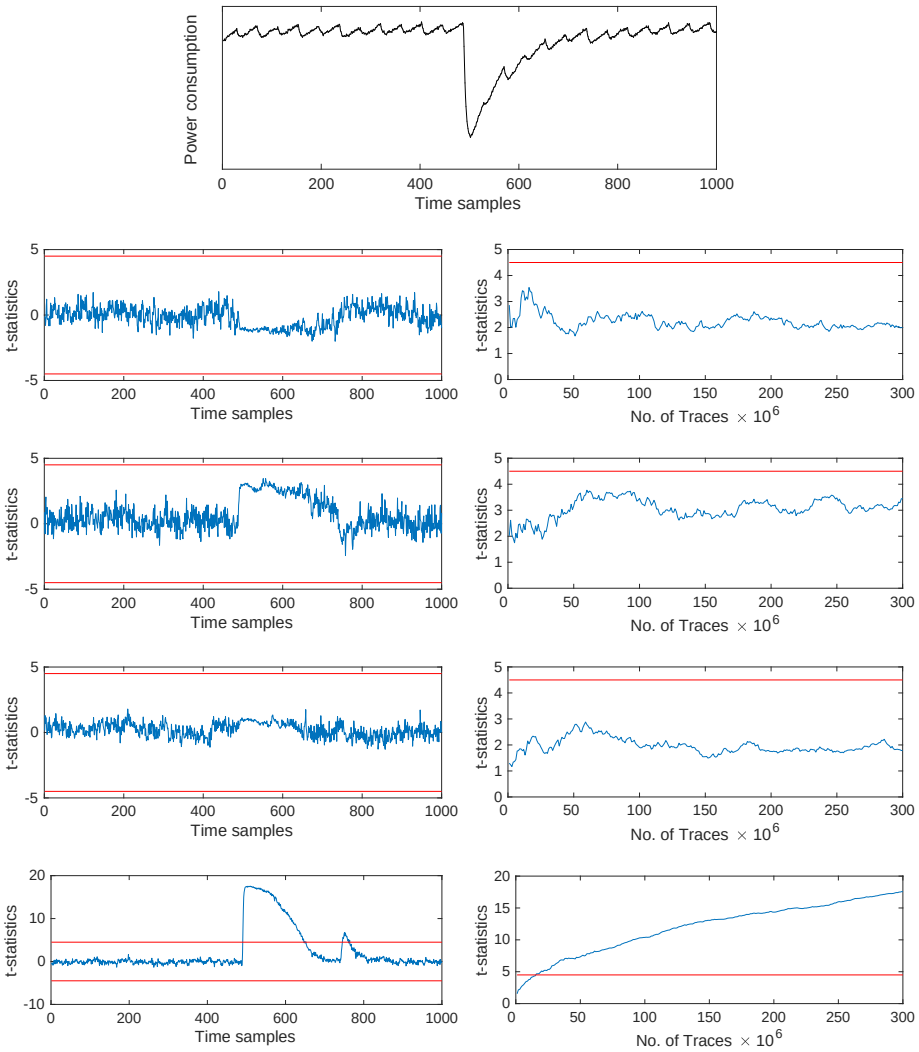
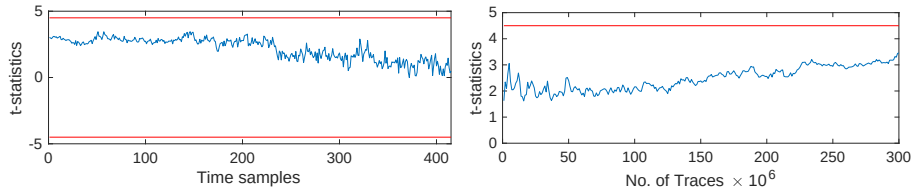


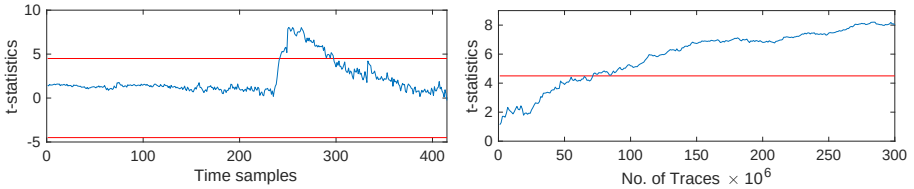
Fig. 6. Sample power trace and univariate non-specific t-test results with 300 million measurements for a single $GF(2^4)$ multiplier masked by means of CMS with $d = 3$. The second to fifth rows show the t-statistics for the statistical moments 1 to 4, respectively. The left column depicts the t-values over time, the right column illustrates the evolution of the absolute maximum t-value over the number of traces.

following binary matrix (i.e., Midori's MixColumns matrix [3]):

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$



(a) 2nd order, clocks $(t, t+1)$



(c) 3rd order, clocks $(t, t, t+1)$

Fig. 7. Multivariate non-specific t-test results with 300 million measurements for a single $GF(2^4)$ multiplier masked by means of CMS with $d = 3$. The left column depicts the t-values over time, the right column illustrates the evolution of the absolute maximum t-value over the number of traces.

Note that such leakage amplifier modules are completely separated and never mixed with each other, which could potentially violate the independence assumption of the masking scheme. They simply lead to a higher energy consumption depending on their corresponding input, which helps to achieve a higher SNR when the signal is much smaller than the noise level. As a result, the leakage corresponding to the third-order flaw becomes detectable by the t-test.

DOM. Similar to the CMS experiments, we implemented the DOM-*dep* multiplier (shown in Figure 2 for the $d = 2$ case) by instantiating all multiplications as $GF(2^4)$ multipliers. We chose to perform the experimental verification for the $d = 3$ case, since the exploitation of this flaw imposes less restrictive constraints on the timing of the signals at the output of the register in the construction. Like before we could only detect the leakage by amplifying the power consumption of the probed values that are detailed in Section 4. Accordingly, one leakage amplifier is connected to $a_3 \cdot z_1$, another one to $a_4 \cdot z_2$. For the last one we implemented the XOR of the Decode operation in such a way that the XOR between the first two elements, i.e., $(a_1 + z_1) + (a_2 + z_2)$, is calculated before the third term is added. The output of this earlier evaluated XOR then supplies the third leakage amplifier module. Note that such a particular order of the aforementioned XORs does not violate the claims of the DOM-*dep* multiplier [29] and could indeed occur in reality when synthesizing the construction.[5]

[5] Overall, we believe it is desirable that the security of a glitch-resistant gadget does not rely on the assumption that specific signal timings in the combinational paths are unlikely to occur, since this leads to security guarantees which can be falsified by physical defaults. The same is true for the example with $d = 2$ and two probes.

Analogous to the CMS case we report univariate non-specific t-test results up to the fourth order in Figure 8 and multivariate analyses up to the third-order in Figure 9. It can be seen that the univariate fourth-order t-test and the

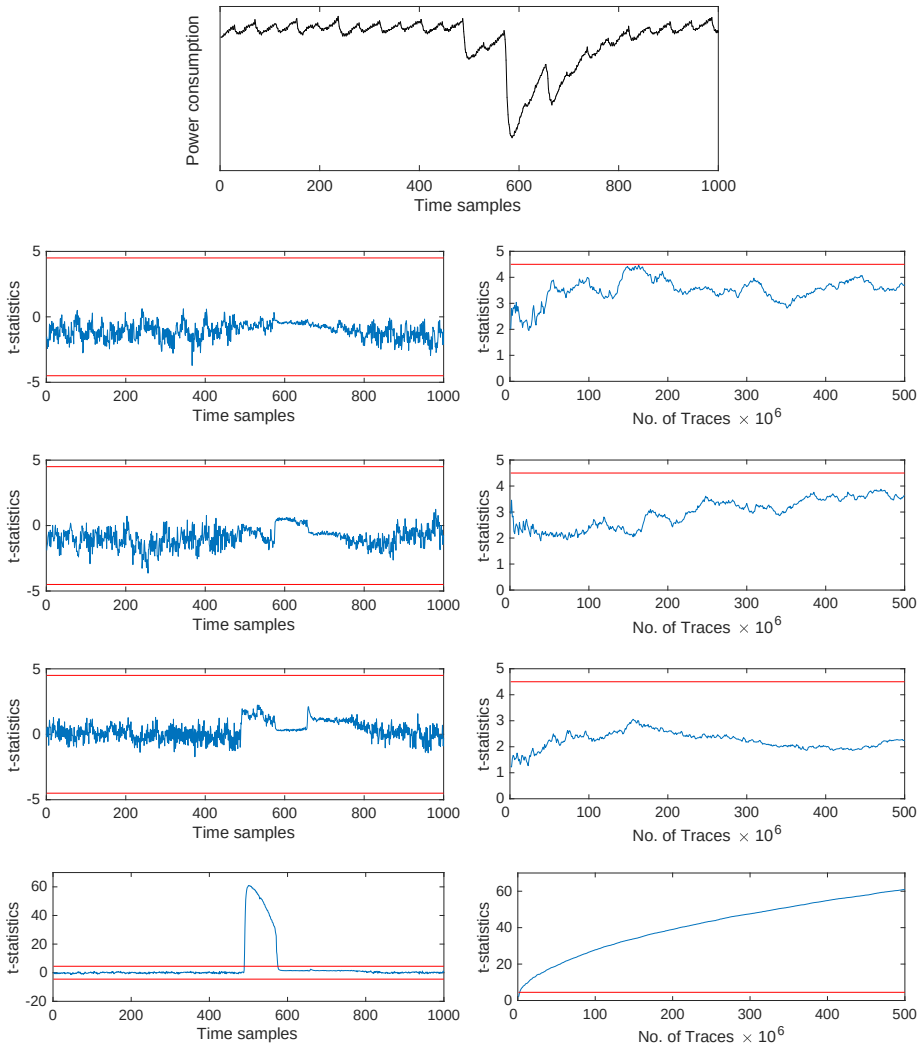
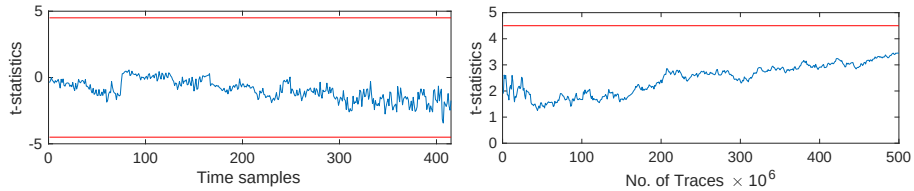
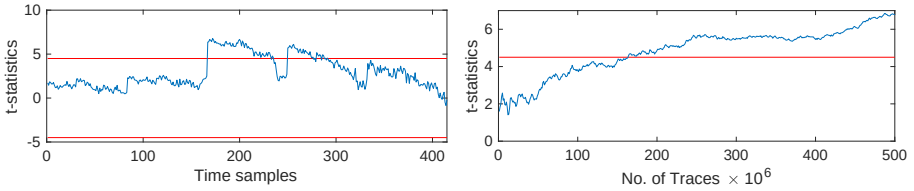


Fig. 8. Sample power trace and univariate non-specific t-test results with 500 million measurements for a single DOM-*dep* multiplier with $d = 3$, based on $GF(2^4)$ multiplications. The second to fifth rows show the t-statistics for statistical moments 1 to 4.

multivariate third-order t-test indicate leakage with high-confidence ($t > 4.5$) when considering 500 million traces. Note that the corresponding offsets for the multivariate tests are identical to the ones previously outlined in the CMS case.



(a) 2nd order, clocks $(t, t+1)$



(c) 3rd order, clocks $(t, t, t+1)$

Fig. 9. Multivariate non-specific t-test results with 500 million measurements for a single DOM-*dep* multiplier with $d = 3$, based on $GF(2^4)$ multiplications.

Thus, the smallest data-dependent statistical moment is indeed the third one, which confirms the existence of the theoretically exhibited flaw.

Related work. Recently a first practical side-channel evaluation of a full block cipher (triple-DES) protected by domain-oriented masking has been published at COSADE 2018 [43]. This work makes extensive use of the DOM-*dep* multiplier to construct the DES substitution box and provides univariate t-test results with 50 million power traces taken from an FPGA implementation of the full cipher in the $d = 1$ case and 2 billion power traces in the $d = 2$ case. They come to the conclusion that their masked S-box indeed delivers the corresponding protection order promised by DOM. However, in view of our new results we assume that a multivariate analysis could have revealed a second-order leakage in the $d = 2$ case, which is an interesting scope for further investigations.

UMA. In the case of UMA we do not evaluate a single instance to reveal the existence of a local flaw, but compose two multiplications (with $d = 2$) as depicted in Figure 10 to show that UMA suffers from a lack of composability. Each of the multiplications (upper and lower half of the figure) consists of one half-complete block and the corresponding inner-domain terms (as detailed in [27]). Furthermore, the randomness optimizations by Belaïd et al. are in place. The registers which are depicted in black solid lines are mandatory by design. According to the authors of UMA, the green dashed registers are optional pipeline registers; the red dashed registers are output synchronization stages to separate the multiplications from each other; the blue registers are optional pipelining registers due to the composition; and the purple registers are optional pipeline registers for the inner-domain terms (which are not considered by the authors of UMA). Each multiplication in Figure 10 depicts the instantiation of one $GF(2^4)$ multiplier. To show that UMA does not satisfy composability as ensured by the

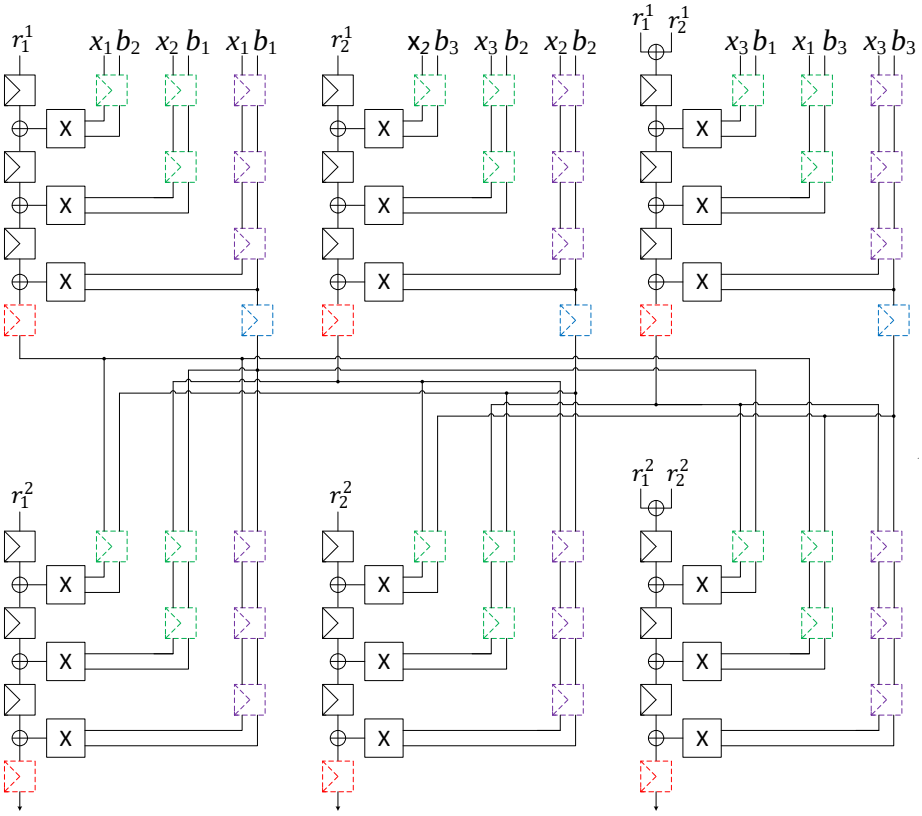


Fig. 10. Composition of two UMA multiplications ($d = 2$) with several kinds of (mandatory and optional) pipelining and synchronization register stages.

definition of strong non-interference (SNI) we consider the following scenario. If an adversary places one output probe on the leftmost output register of the first multiplication and one internal probe on the fresh randomness r_1^1 he can observe a joint distribution that depends on two shares of b and two shares of x and thus can not be simulated with only one share of each input. The latter leads to the attack with only two probes on the composed multiplications detailed in Section 5. Probably caused by the lower number of shares compared to the previous experiments, or just a stronger bias that is imposed by the flaw, we were not forced to use any leakage amplifiers or other particular considerations to detect the corresponding leakage. When applying all (mandatory and optional) registers that are included in Figure 10 the leakage corresponding to the composability flaw can directly be observed as multivariate second-order leakage. A sample trace and the univariate t-test results up to the third order are depicted in Figure 11, while the multivariate second-order result can be seen in Figure 12.

In this last experiment the multivariate leakage could not be observed in two

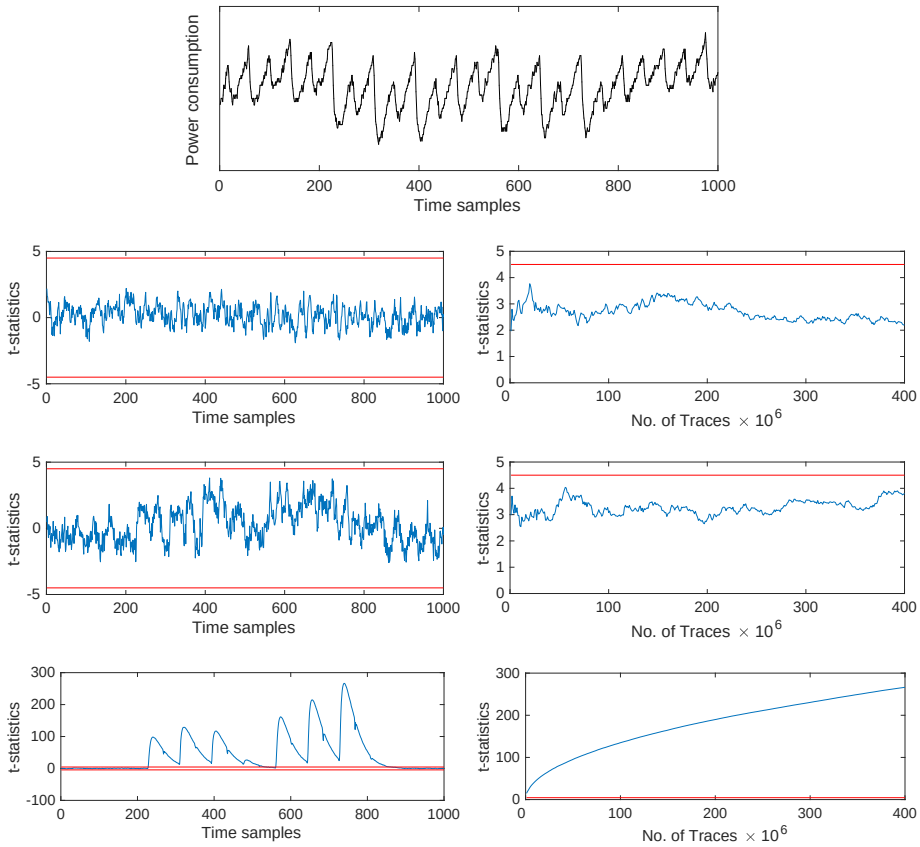


Fig. 11. Sample power trace and univariate non-specific t-test results with 400 million measurements for two composed UMA multiplications with $d = 2$, based on $GF(2^4)$ multipliers. The second to fourth rows show the t-statistics for the statistical moments 1 to 3, respectively, arranged like before.

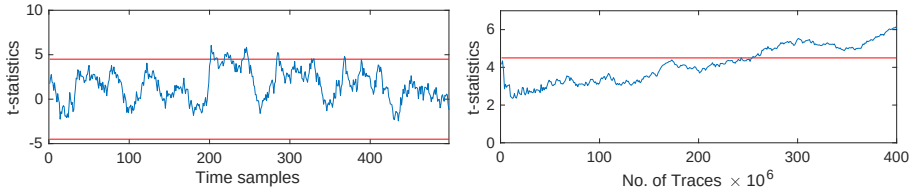


Fig. 12. Multivariate second-order non-specific t-test results, clocks $(t, t+5)$, with 400 million measurements for two composed UMA multiplications with $d = 2$, based on $GF(2^4)$ multipliers, arranged like before.

consecutive clock cycles, but in sample points with an offset of 5 clock cycles. Figure 12(a) shows the resulting t-statistics curve when shifting this offset of 5 clock cycles over the whole 1000 time samples.

Discussion. In this section we have demonstrated that all of the exhibited flaws from Sections 3 to 6 are practically detectable in real-world power measurements, which effectively reduces the protection order of the corresponding schemes. However, our results do not imply that these flaws necessarily reduce the practical security level of full implementations instantiating these schemes. Admittedly, the biases caused by the flaws have a low amplitude and therefore may be hard to exploit in some cases. For example, for the concrete signal-to-noise ratio and number of shares in our experiments, an exploitation of the univariate leakage in the $(d+1)$-th order will generally succeed with less traces than considering the multivariate d-th order leakage for an analysis. Yet, the reduction of the protection order raises doubts about higher noise levels and a larger number of shares (especially in the case of CMS, where the exploitation effort due to the flaw does not scale with the number of shares when $d > 2$).

We note that our findings do not imply that it is impossible to construct d-probing secure circuits with the investigated gadgets. In case of UMA for example the authors build a substitution box using the locally secure gadgets and verify the probing security of the composition by exhaustively analyzing the resulting circuit for small protection orders [28]. In this regard they make use of the recently introduced tool in [10]. Such an approach is generally valid and can potentially lead to more efficient constructions than composing only SNI gadgets. However, the (smart) exhaustive analysis it performs still does not scale well for full implementations protected with a large numbers of shares.

8.2 Composability in Hardware - A Matter of Registers

As already mentioned, compositional security does not only depend on the amount of fresh randomness that is applied, but also on the correct instantiation of register stages in the composed circuits. While this is usually not an issue for software implementations, where all operations are inherently processed in a sequential manner, hardware implementations offer a lot more freedom in terms of parallelization and order of operations. Thus, special care needs to be taken in order to not degrade the security of the whole implementation by an incorrect placement of memory elements. In the robust probing model, this is formalized by the fact that an adversary can always choose to probe an internal (glitchy) computation or a stable output, and only the latter ones are excluded from the probe count in the SNI definition. Combined with the fact that the "share fan-in" of a glitch-robust and composable multiplication should be minimum, it guided the design of the (so far only) robust and composable multiplication algorithm and implementation proposed in [21], which requires $(d+1)^2 + (d+1)$ registers to store all the (refreshed) partial products and the final output.

DOM. As a case study, we take a look at the DOM-*indep* multiplier of the domain-oriented masking scheme, initially proposed in [29]. The refresh layer (called resharing step in [29]) of the DOM-*indep* multiplier is d-SNI. Furthermore, the full multiplier is d-probing secure in the presence of glitches. However,

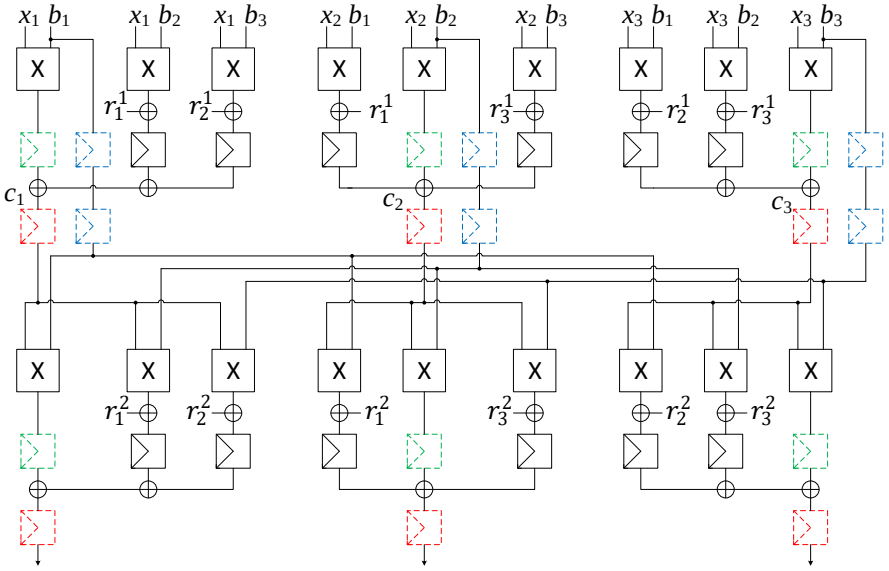


Fig. 13. Composition of two DOM-*indep* multiplications (d = 2) with several kinds of (mandatory and optional) pipelining and synchronization register stages.

this is not sufficient to guarantee that any composition of DOM-*indep* multipliers leads to a d-probing secure (or d-SNI) circuit. In Figure 13 we have depicted such a composition of two DOM-*indep* multipliers for the $d = 2$ case where different possibilities for the inclusion of register stages are illustrated. Only the black solid registers are mandatory by design. In particular the green and red dashed registers are claimed to be optional (and not relevant for the security of the gadget) [29]. We show in the following that especially the red dashed output registers which separate both multipliers from each other are in fact crucial for the compositional security. For this purpose, we have implemented the design in Figure 13, but left out the red output registers as well as the neighboring blue ones to ensure correct pipelining. With respect to the robust probing model such an implementation violates the requirement that any composition of two gadgets with a limited share fan-in should be separated by memory elements [21]. Like before the construction has been implemented based on $GF(2^4)$ multipliers. We acquired 500 million power traces suitable for a non-specific t-test evaluation. The results for the univariate case are shown in Figure 14. As illustrated in the figure, a significant univariate second-order leakage can be observed. To explain the source of this leakage we consider one extended probe on the computation of the cross-product $c_1 \cdot b_2$, where c is the output of the upper DOM-*indep* multiplier. This probe gives access to the following input variables:

$$P_{1,1} = b_2, \tag{32}$$

$$P_{1,2} = x_1 \cdot b_1, \tag{33}$$

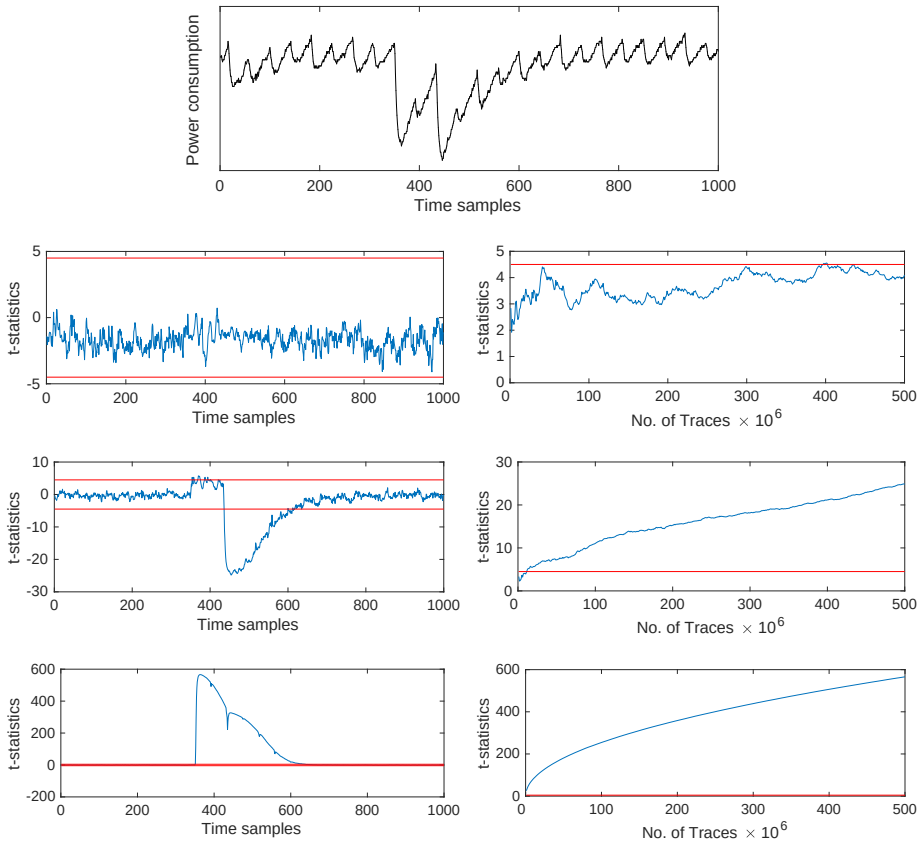


Fig. 14. Sample power trace and univariate non-specific t-test results with 500 million measurements for two composed DOM-*indep* multiplications ($d = 2$) based on $GF(2^4)$ multipliers without output registers but with pipeline registers applied. The second to fourth row show the t-statistics for the statistical moments 1 to 3, respectively.

$$P_{1,3} = x_1 \cdot b_2 + r_1^1, \tag{34}$$

$$P_{1,4} = x_1 \cdot b_3 + r_2^1. \tag{35}$$

Combining $P_{1,1}, P_{1,2}$ as:

$$P_1' = P_{1,1} + P_{1,2} = b_2 + x_1 \cdot b_1. \tag{36}$$

results in a distribution that depends on two shares of b already. Placing a second (regular) probe on b_3 (or an extended probe on some cross-product involving b_3) leads to a distribution depending on b, which results in (univariate) second-order leakage. An analogous attack with one probe exists for the $d = 1$ case.

Related work. The same article that already instantiated DOM-*dep* multipliers to protect a full triple-DES circuit also proposes to use DOM-*indep* multipliers in their construction [43]. Unfortunately, the authors make the mistake of

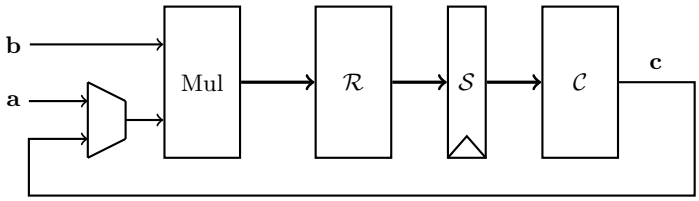


Fig. 15. Iterative multiplication following the concept of GLM with refresh layer $\mathcal{R}$, synchronization layer $\mathcal{S}$, and compression layer $\mathcal{C}$.

composing DOM-*indep* multipliers without any output register in between (as apparent for example in Figure 3 of [43]). No practical side-channel analysis is presented for the implementations that make use of the DOM-*indep* multipliers. It is an interesting open question whether a univariate leakage in the d-th moment would show up in such a case, as in our experiments.

GLM. After having seen that the GLM scheme, explained in Section 6, is insecure for higher orders due to its instantiation of the CMS refresh layer, it might be tempting to simply replace the insufficient refreshing step by an SNI one, for example the DOM-*indep* refresh layer (especially since the authors of GLM specifically leave the search for a more suitable alternative open to future work [26]). A simplified schematic of the GLM hardware design is shown in Figure 15. One can see that no register stage is placed after the compression layer. Due to the absence of this register stage, the just presented composability issues of the DOM-*indep* multiplication would arise, rendering the whole construction insecure. Adding such a register stage would on the one hand fix the security issue, but on the other hand also add an additional delay of one clock cycle per cross-product, which is not ideal for a low-latency construction. To confirm that including the output register indeed fixes the security problems we have implemented the design in Figure 13 with all of the registers being present and measured another 500 million traces. A sample trace and the corresponding results can be seen in Figure 16 for the univariate case and in Figure 17 for the multivariate case. As expected no leakage in the first two statistical moments can be observed, although admittedly the t-values in the multivariate second-order analysis come close to the 4.5 threshold. We observed those large t-values for an offset of 4 clock cycles and assume them to be a random occurrence.

Pipelining Registers We further detail the relevance of pipelining registers for the security of multiplication gadgets in Appendix A and show that they are not optional with case studies based on DOM and UMA. In this case as well, the main message is that in order to preserve robustness against glitches and composability jointly, it is needed to implement registers to separate all the refreshed partial product computations and the compressed output. As detailed in [21], the latter requires $(d+1)^2+(d+1)$ registers for a 2-cycle multiplication,

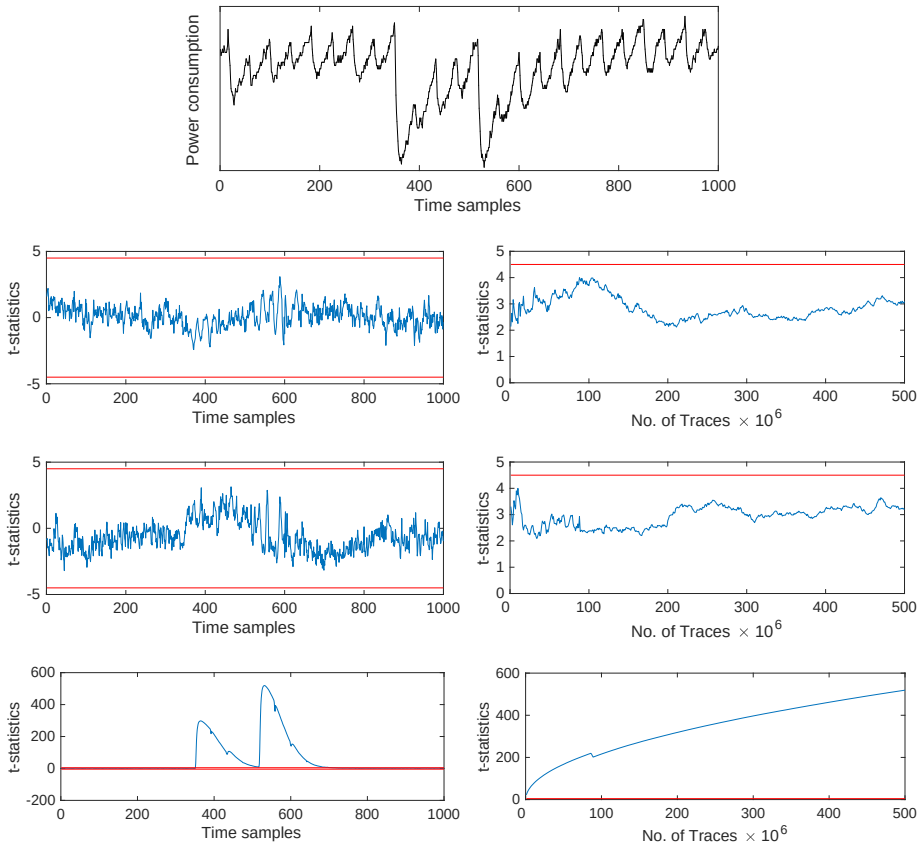


Fig. 16. Sample power trace and univariate non-specific t-test results with 500 million measurements for two composed DOM-indep multiplications ($d = 2$) based on $GF(2^4)$ multipliers with all registers applied. The second to fourth row show the t-statistics for the statistical moments 1 to 3, respectively, arranged like before.

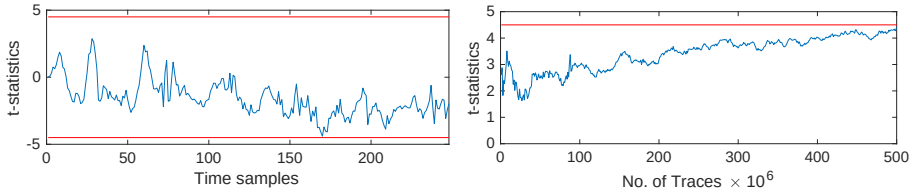


Fig. 17. Multivariate second-order non-specific t-test results, clocks $(t, t+4)$, with 500 million measurements for two composed DOM-indep multiplications ($d = 2$) based on $GF(2^4)$ multipliers with all registers applied, arranged like before.

which is quite expensive. Interestingly, our conclusion for DOM and UMA is in fact identical. Finding solutions (or showing impossibility) with less registers (or randomness), is one more direction for future investigations.

9 Further remarks and conclusions

In contrast with software-oriented masking, security proofs are not yet an established tool in hardware-oriented masking. One reason for this situation was the lack of an appropriate model that formally covers the local security and composability of masked gadgets in presence of physical defaults. As a result, engineering intuition and informal considerations of probing security with respect to glitches were often the only considered arguments supporting security claims of proposed masked circuits. The robust probing model in [21] now makes it possible to analyze and subsequently prove security guarantees of masked hardware gadgets. Our broad analysis of (scalable) hardware-oriented masking schemes revealed that not a single multiplication gadget which comes without a proof in the robust probing model actually delivers local *and* compositional security for arbitrary protection orders (at least when instantiated like proposed by the respective authors). This is confirmed by our empirical investigations, which showed that flaws with respect to the robust probing model can directly translate to exploitable leakage in real-world power measurements. Although the fact that these flaws lead to the most informative leakages depends on the implementations, it at least reveals an undesirable source of risk, especially as the claimed security orders increases. In fact, only when tweaking the ISW multiplier [31] in a way that makes it similar to the DOM-*indep* multiplier in [29], but employing its pipeline registers and additionally storing its outputs in a further register stage, one ends up with a gadget that is SNI in the presence of glitches. This gadget was proposed and proven secure for arbitrary orders in [21] and additionally, it is the only $(d+1)$-masked multiplication circuit that did not exhibit detectable leakage up to the d-th order in our experiments.

Acknowledgments. The authors want to thank Hannes Gross and Stefan Mangard for their valuable feedback, especially for their help improving the flaw identified for DOM-*dep* with their tool from EUROCRYPT 2018 [10].

François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). The work described in this paper has been supported in part by the European Commission through the ERC project 724725 (acronym SWORD) and by the German Research Foundation (DFG) through the project "NaSCA: Nano-Scale Side-Channel Analysis".

References

1. Side-channel AttacK User Reference Architecture. `http://satoh.cs.uec.ac.jp/SAKURA/index.html`.
2. Victor Arribas, Begül Bilgin, George Petrides, Svetla Nikova, and Vincent Rijmen. Rhythmic keccak: SCA security and low latency in HW. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):269–290, 2018.
3. Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 411–436. Springer, 2015.

4. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 457–485. Springer, 2015.
5. 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 *CCS 2016*, pages 116–129. ACM, 2016.
6. 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 *EUROCRYPT 2017*, volume 10210 of *LNCS*, pages 535–566, 2017.
7. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 616–648. Springer, 2016.
8. Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In *ASIACRYPT 2014*, volume 8874 of *LNCS*, pages 326–343. Springer, 2014.
9. Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A more efficient AES threshold implementation. In *AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 267–284. Springer, 2014.
10. 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 *EUROCRYPT 2018*, volume 10821 of *LNCS*, pages 321–353. Springer, 2018. `http://eprint.iacr.org/2017/897`.
11. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO '99*, volume 1666 of *LNCS*, pages 398–412. Springer, 1999.
12. Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 shares in hardware. In *CHES 2016*, volume 9813 of *LNCS*, pages 194–212. Springer, 2016.
13. Jeremy Cooper, Elke De Mulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test Vector Leakage Assessment (TVLA) Methodology in Practice. International Cryptographic Module Conference, 2013.
14. Jean-Sébastien Coron. Higher order masking of look-up tables. In *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 441–458. Springer, 2014.
15. Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side channel cryptanalysis of a higher order masking scheme. In *CHES 2007*, volume 4727 of *LNCS*, pages 28–44. Springer, 2007.
16. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In *FSE 2013*, volume 8424 of *LNCS*, pages 410–424. Springer, 2014.
17. Joan Daemen. Spectral characterization of iterating lossy mappings. In *SPACE 2016*, volume 10076 of *LNCS*, pages 159–178. Springer, 2016.
18. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, 2014.
19. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 401–429. Springer, 2015.

20. Hassan Eldib, Chao Wang, and Patrick Schaumont. Formal verification of software countermeasures against side-channel attacks. *ACM Trans. Softw. Eng. Methodol.*, 24(2):11:1–11:24, 2014.
21. Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults and the robust probing model. *IACR Cryptology ePrint Archive*, 2017:711, 2017.
22. Ashrujit Ghoshal and Thomas De Cnudde. Several masked implementations of the boyar-peralta AES s-box. In *INDOCRYPT 2017*, volume 10698 of *LNCS*, pages 384–402. Springer, 2017.
23. Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In *CHES 2002*, volume 2523 of *LNCS*, pages 198–212. Springer, 2003.
24. Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for Side channel resistance validation. In *NIST Non-invasive Attack Testing Workshop*, 2011.
25. Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In *EUROCRYPT 2017*, volume 10210 of *LNCS*, pages 567–597, 2017.
26. Hannes Gross, Rinat Iusupov, and Roderick Bloem. Generic low-latency masking in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(2), 2018.
27. Hannes Groß and Stefan Mangard. Reconciling d+1 masking in hardware and software. In *CHES 2017*, volume 10529 of *LNCS*, pages 115–136. Springer, 2017.
28. Hannes Groß and Stefan Mangard. A unified masking approach. *J. Cryptographic Engineering*, 8(2):109–124, 2018.
29. Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In *TIS@CCS 2016*. ACM, 2016. `http://eprint.iacr.org/2016/486`.
30. Hannes Groß, Stefan Mangard, and Thomas Korak. An efficient side-channel protected AES implementation with arbitrary protection order. In *CT-RSA 2017*, volume 10159 of *LNCS*, pages 95–112. Springer, 2017.
31. Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.
32. Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 351–365. Springer, 2005.
33. Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *CHES 2005*, volume 3659 of *LNCS*, pages 157–171. Springer, 2005.
34. Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.
35. Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action - - case study of PRINCE and midori -. In *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 517–547, 2016.
36. 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.
37. 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.

38. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 142–159. Springer, 2013.
39. Oscar Reparaz. A note on the security of higher-order threshold implementations. *IACR Cryptology ePrint Archive*, 2015:1, 2015.
40. Oscar Reparaz. Detecting flawed masking schemes with leakage detection tests. In *FSE 2016*, volume 9783 of *LNCS*, pages 204–222. Springer, 2016.
41. Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *CRYPTO 2015*, volume 9215 of *LNCS*, pages 764–783. Springer, 2015.
42. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In *CHES 2010*, volume 6225 of *LNCS*, pages 413–427. Springer, 2010.
43. Pascal Sasdrich and Michael Hutter. Protecting triple-des against DPA - A practical application of domain-oriented masking. In *COSADE 2018*, volume 10815 of *LNCS*, pages 207–226. Springer, 2018.
44. Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In *CHES 2015*, volume 9293 of *LNCS*, pages 495–513. Springer, 2015.
45. Tobias Schneider and Amir Moradi. Leakage assessment methodology - extended version. *J. Cryptographic Engineering*, 6(2):85–99, 2016.
46. Kai Schramm and Christof Paar. Higher order masking of the AES. In *CT-RSA 2006*, volume 3860 of *LNCS*, pages 208–225. Springer, 2006.

A On the Need of (Pipelining) Registers

In the second part of Section 8 we have shown that register stages are crucial ingredients for the construction of composable gadgets. However, up to this part it was only demonstrated that a lack of output registers can have serious consequences on the compositional security of locally secure gadgets. In this appendix, we provide experimental evidence for the fact that, even when output registers are employed, a lack of pipelining stages can lead to a reduction of the protection order as well. We illustrate this claim with UMA and DOM.

The starting point of our analysis is the illustration of Figure 10, where one could assume that the depicted registers are supposed to be enabled all at once and then be kept active for a determined number of clock cycles until the correct results are stable at the output. One might also assume that the registers are reset (e.g., to zero) before applying new values to the inputs, as it is a usual practice in hardware design (especially when pipelining is not a considered use case). We show that it is not possible to safely make those assumptions when composing UMA multiplications and argue that the same is true for the DOM-*indep* multiplier when implemented without pipelining registers.

In the case of UMA no pipelining registers (not even optional ones) are included in the paths for the inner-domain terms. Thus, the purple dashed registers in Figure 10 are not present in the UMA scheme as proposed by the authors. Accordingly, the result of the inner-domain terms will propagate to the output of a UMA gadget first. Let us assume for a moment that a state machine controlling this circuit iterates over the following three simple states. At first all registers are reset to zero, then the shared multiplication inputs are applied to the inputs of the circuit and afterwards all registers are enabled for 8 consecutive clock cycles. In this case $x_1 \cdot b_1$, $x_2 \cdot b_2$ and $x_3 \cdot b_3$ are evaluated right after the inputs are applied. Before being saved into the output register of the first multiplier these values are input to an XOR with zero (due to the reset of the registers), which does not change their value. Accordingly, after being enabled for one clock cycle, $x_1 \cdot b_1$, $x_2 \cdot b_2$ and $x_3 \cdot b_3$ are propagated to the second multiplier, where they are multiplied with all shares of b individually (i.e., two shares of b are combined in each multiplication without proper resharing). Thus, trivial univariate second-order leakage emerges due to the early propagation of partial results. This is confirmed by the univariate non-specific t-tests considering 200 million power traces in Figure 18, where we also performed a reset of all registers before each multiplication. When taking a look at the DOM-*indep* multiplier in Figure 13, it appears that omitting the green (and neighboring blue) registers leads to the same problem. To demonstrate this we also measured 100 million power traces of two composed DOM-*indep* multipliers without pipelining registers, but with output registers. The results of the t-test are depicted in Figure 19.

In fact in both of those cases the leakage is even more drastic when additionally removing the output registers as well. We have verified this again with experiments (100 million traces each), as apparent in Figures 20 and 21. We note again that we do not claim these bad combinations are the only possible

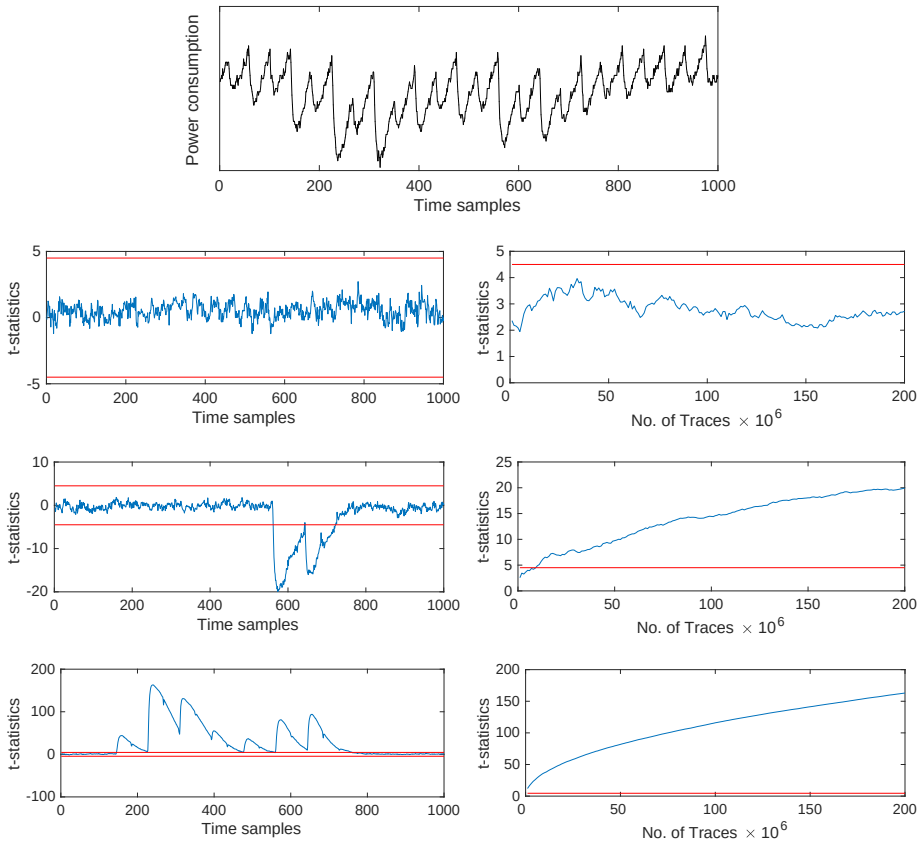


Fig. 18. Sample power trace and univariate non-specific t-test results with 200 million measurements for two composed UMA multiplications ($d = 2$) based on $GF(2^4)$ multipliers without pipeline registers for the inner-domain terms. The second to fourth row show the t-statistics for the statistical moments 1 to 3, respectively.

ones. We just mean that the authors' guidelines are not strictly sufficient to avoid these issues. For example, if the gadgets are not as directly connected as in our examples, but with synchronization stages and other modules in between, these problems may not arise. Furthermore, in all the presented cases the security issues can easily be fixed by putting additional constraints on the registers that have to be observed by a state machine (e.g., not allowing a reset; only activating the output register after a certain number of clock cycles; not propagating b to the second multiplier before the first one is finished; ...). However, our results highlight that without an explicit guideline on how to treat the register stages, it is strongly advised to use fully pipelined circuits, even when pipelining is not a considered use case, in order to mitigate the early propagation of partial results. This directly complies to the fact that the so far only multiplication gadget which has been proven secure in the robust probing model requires $(d + 1)^2$ registers

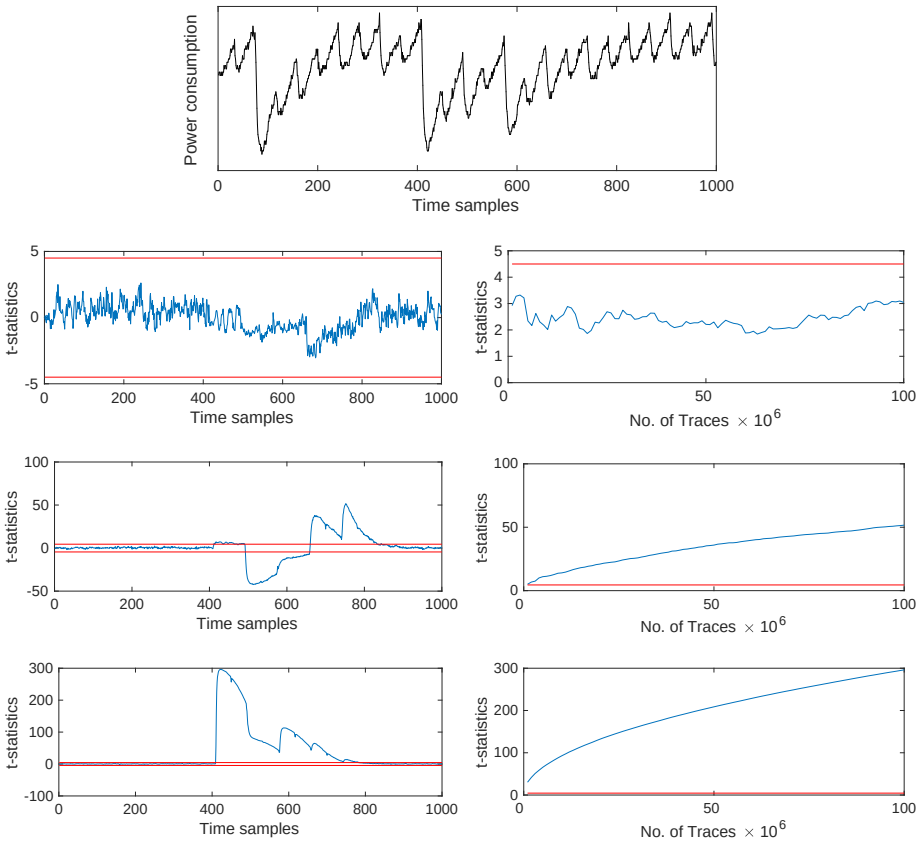


Fig. 19. Sample power trace and univariate non-specific t-test results with 100 million measurements for two composed DOM-*indep* multiplications ($d = 2$) based on $GF(2^4)$ multipliers with output registers but without pipeline registers applied. The second to fourth row show the t-statistics for the statistical moments 1 to 3.

to store the (refreshed) cross-products, and $(d + 1)$ registers to store the shared multiplication output. In this case no specific constraints have to be set on the registers and the gadget is suitable for all reasonable use cases.

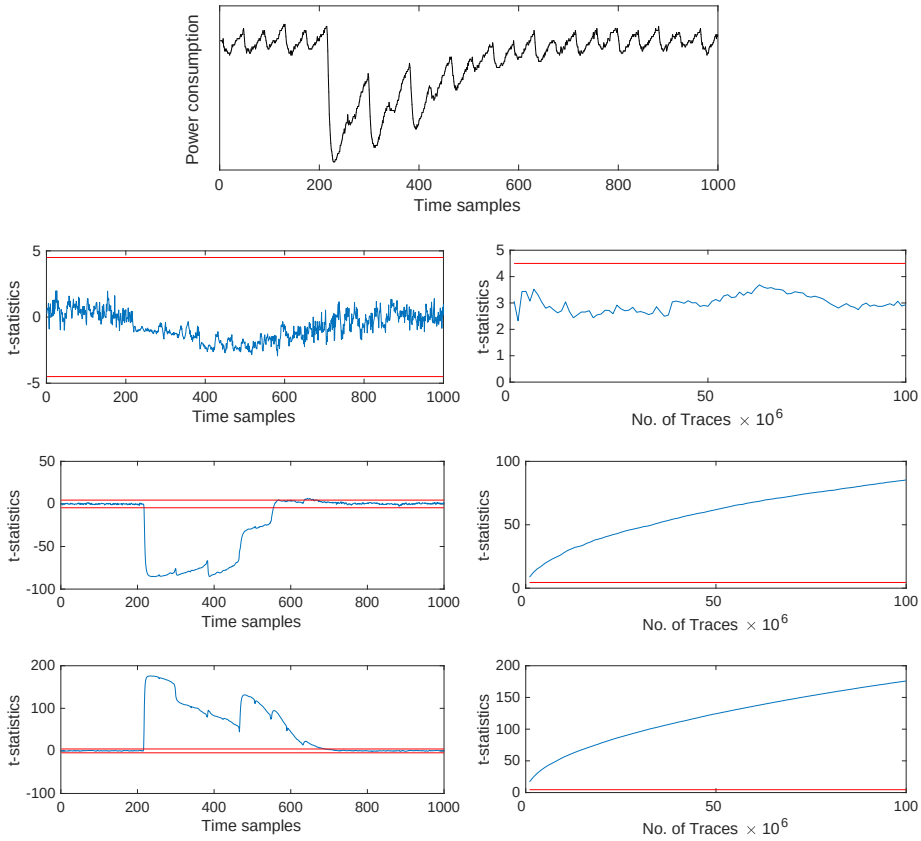


Fig. 20. Sample power trace and univariate non-specific t-test results with 100 million measurements for two composed UMA multiplications ($d = 2$) based on $GF(2^4)$ multipliers with only the mandatory registers applied. The second to fourth row show the t-statistics for the statistical moments 1 to 3, respectively, arranged like before.

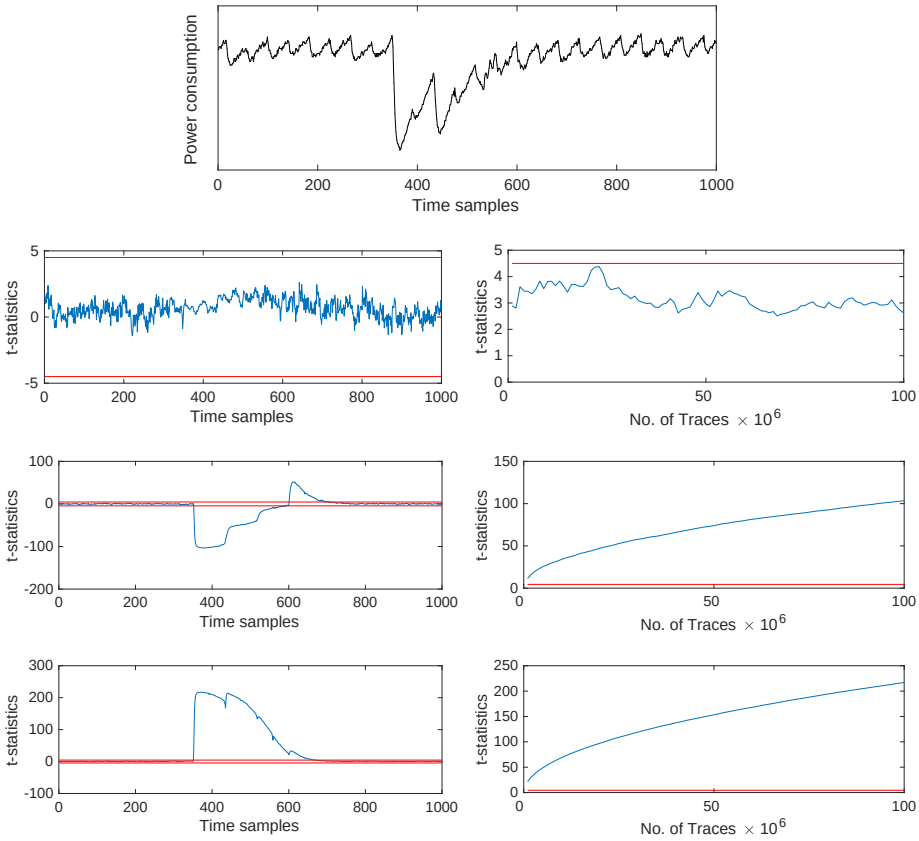


Fig. 21. Sample power trace and univariate non-specific t-test results with 100 million measurements for two composed DOM-*indep* multiplications ($d = 2$) based on $GF(2^4)$ multipliers with only the mandatory registers applied. The second to fourth row show the t-statistics for the statistical moments 1 to 3, respectively, arranged like before.