# Consolidating Security Notions in Hardware Masking

Lauren De Meyer[1], Begül Bilgin[1,2] and Oscar Reparaz[1,3]

[1] KU Leuven, imec - COSIC, Leuven, Belgium
firstname.lastname@esat.kuleuven.be
[2] Rambus, Cryptography Research, Rotterdam, the Netherlands
[3] Square Inc., San Francisco, USA

**Abstract.** In this paper, we revisit the security conditions of masked hardware implementations. We describe a new, succinct, information-theoretic condition called *d*-glitch immunity which is both *necessary and sufficient* for security in the presence of glitches. We show that this single condition includes, but is not limited to, previous security notions such as those used in higher-order threshold implementations and in abstractions using ideal gates. As opposed to these previously known *necessary* conditions, our new condition is also *sufficient*. On the other hand, it excludes avoidable notions such as uniformity. We also treat the notion of (strong) non-interference from an information-theoretic point-of-view in order to unify the different security concepts and pave the way to the verification of composability in the presence of glitches. We conclude the paper by demonstrating how the condition can be used as an efficient and highly generic flaw detection mechanism for a variety of functions and schemes based on different operations.

**Keywords:** Glitches · DPA · SCA · Verification · TI · SNI · Non-Completeness · Mutual Information · Information-theory · d-probing · Glitch Immunity

## 1 Introduction

Cryptographic algorithms are designed such that they are mathematically secure. An adversary with access to, for instance, the ciphertext and plaintext, should not be able to derive the secret key with reasonable computing power. However, this *black box* model often does not suffice in practice, as the existence of side-channels can significantly aide the adversary in his quest for secret information. Since the seminal work of Kocher [Koc96], we have learned of many cheap and scalable side-channel attacks (SCA) that successfully exploit information such as instantaneous power consumption or electromagnetic radiation to recover secret keys, effectively turning the adversary's *black box* into a *grey box*.

In the realm of SCA, the most well known technique is Differential Power Analysis (DPA) [KJJ99], a simple and efficient attack that exploits the fact that the power consumption of an embedded device depends on the intermediate values that the device computes on. In response, many countermeasures have been proposed, among which *masking* is one of the most established.

### 1.1 History and Motivation

**(In)accuracy of leakage functions.** Constructing masked circuits is non-trivial. First of all, describing the leakage of the device with high accuracy as a function of its behavior and performed operation is a tedious work. The typical methodology in developing a masking scheme is to represent the circuit and its leakage in an abstract way as accurate as possible; then prove their security in this abstraction.

A good example is the traditional *d-probing adversary*, where the adversary gets the noiseless information (leakage) of $d$ intermediates that have been probed. Schemes such as ISW [ISW03] are secure in this instantaneous model where the adversary does not see any transition or change in the intermediate. He sees only the *stabilized* value. However, as is shown repeatedly, the proposed countermeasures would show significant vulnerabilities if the abstraction of the leakage function is not accurate [MPG05, BGG$^+$14, PV17]. The naive power model used for ISW does not take into account physical defaults such as memory transitions or glitches that impact hardware implementations.

Glitches are unintentional and undesirable hardware artifacts causing unnecessary power consumption and hardware designers go to great lengths to minimize them for reasons beyond security. However, reducing glitches requires a careful process of path equalization which is a great challenge given factors such as the product and architecture variation, working environment and device age. Most importantly in our context, glitches can momentarily unmask values, thereby invalidating the security guarantees theoretically provided by masking and making it a security hazard [MPG05].

**Towards glitch security with *necessary but not sufficient* conditions.** Threshold implementations (TI) [NRR06] were the first provably secure masking scheme in the presence of glitches. The authors identified two key properties: non-completeness and uniformity, which together ensure the provable security of TI against first-order DPA in the presence of glitches on any hardware as long as the independent leakage assumption of masking holds.

The concept was extended to higher-order security by Bilgin *et al.* [BGN$^+$14], but it was later shown by Reparaz *et al.* [RBN$^+$15] that non-completeness and uniformity no longer suffice to achieve higher-order security due to the possibility of multivariate attacks. The authors also refined the definition of non-completeness to the level of individual variables and hence opened up the possibility for securely using only $d+1$ shares in the presence of glitches.

The property of non-completeness remains today a *necessary* requirement for the security of masked hardware implementations and therefore lies at the basis of every design process. However, it is still unclear how to define a condition that is not only necessary but also *sufficient*. This means that a lot of masked implementations have actually been designed without a clear understanding of the underlying security notions and how particular decisions impact the resistance against SCA.

**Verification tools.** As a result, many proposed countermeasures of the last years (whether they be Boolean masking, multiplicative or additive) have been shown to be vulnerable relatively quickly after being published. This is a common trend both in software and hardware masking [AG01, PGA06, SP06, RP10, BFGV12, BGN$^+$14, HT16]. This history of trial and error has given rise to a new wave of works on the verification of masking schemes, ranging from formal to statistical [Rep16, ANR18, Cor18, BGI$^+$18, BBFG18, SBY$^+$18]. Despite this being an active area of research, there are still various types of schemes for which no tools are available to prove their security. For example, for multiplicative and arithmetic masking, no verification tool for hardware and no comprehensive formal verification tool for software exists.

**Towards composability with *sufficient but not necessary* conditions.** Growth of the circuit and the resulting computational complexity pose a limit on our ability to verify implementations. In response, a lot of effort has been put into making schemes *provably* secure using the concept of (strong) non-interference which implies composability. More specifically, $d$-SNI gadgets can be composed to provide $d$-probing security which allows for a more efficient verification of large circuits. These notions were originally devised for ideal circuits [Bel15, BBD$^+$16], but have been extended with glitches among others by [FGMDP$^+$18, BBFG18]. The main disadvantage of this approach is that it is over-conservative (not a necessary condition) and typically results in more randomness usage than strictly required.

### 1.2 Our Contribution

In the next section, we introduce an information-theoretic metric, which is conceptually extremely simple as well as easy to verify. Above all, it is the first *necessary and sufficient* condition for $d$-probing security in the presence of glitches. We revisit the definition of probing security as introduced in [GM10, Def. 4], *i.e.* a circuit is $d$-probing secure if the mutual information of any set of $d$ probes with the secret is zero. We redefine this notion for hardware implementations using the adversary model of [RBN$^+$15], where each probed wire gives the adversary information about all the inputs to that wire up to the last synchronization point. By replacing each probe with its glitch-extended version (*i.e.* the set of inputs up to the last synchronization point), we obtain an information-theoretic condition for probing security in the presence of glitches. While models for probing security with glitch-extended probes have been discussed already in a number of previous works [RBN$^+$15, FGMDP$^+$18], we think its security condition has not been described formally with a single equation so far. Moreover, no prior work has described how to combine glitch-extended probes for higher orders. In this work, we fill that gap.

We compare our new property with existing notions such as non-completeness in §3 and (strong) non-interference in §4 and we unify these concepts by introducing new information-theoretic definitions. This consolidation into a single framework, brings much needed clarity and more understanding of how they are related. It also allows us to formulate a mathematical formula for the composability of hardware implementations. In addition, we prove that uniformity, despite being enforced in most works on masking, is not a necessary condition for secure masked hardware implementations.

Finally in §5, we detail how our new condition can be used to detect flaws and provably validate schemes. We demonstrate the flexibility of the condition and point out important features that are not yet included in state-of-the-art verification tools such as [BBFG18] in §5.2. We give examples of implementations which are difficult to verify with available tools today, because of incompatibility with the type of masking, restrictive input distributions and lack of support for randomness recycling among others.

## 2 SCA Security in the Presence of Glitches

### 2.1 Preliminaries

**Notation.** We use lowercase letters, *e.g.* $x$ to denote random variables and capital letters, *e.g.* $F$ for functions. Bold font is used for a sharing of these, *i.e.* $\boldsymbol{x} = (x_0, x_1, \ldots)$ is a sharing of $x$ and $\boldsymbol{F} = (F_0, F_1, \ldots)$ is a masked $F$. We denote specific realizations of $x$ by superscript $x^*$. Further, we let $\boldsymbol{x}_{\bar{i}} = (x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots)$ be the vector obtained by removing $x_i$ from $\boldsymbol{x}$.

The probability distribution on $x$ is denoted by $p(x)$ and $H(x)$ is the Shannon entropy of $x$: $H(x) = -\sum_{x^*} p(x^*) \log_2(p(x^*))$. For any $x, y$, we use $p(x|y)$ to represent the conditional probability of $x$ on $y$ and $I(x; y)$ the mutual information between $x$ and $y$. The two notions are connected by the relation $I(x; y) = H(x) - H(x|y)$. Note that mutual information is symmetric, *i.e.* $I(x; y) = I(y; x)$.

Finally, we denote a set of multiple random variables, *e.g.* $(x, y)$ by caligraphy letters $\mathcal{X}$. The union of two sets $\mathcal{X} \cup \mathcal{Y}$ is the set consisting of all variables that belong to either $\mathcal{X}$ or $\mathcal{Y}$ or both: $\mathcal{X} \cup \mathcal{Y} = \{x : x \in \mathcal{X} \text{ or } x \in \mathcal{Y}\}$.

**Statistical independence.** Recall that there are a multitude of equivalent ways to express the statistical independence of two discrete random variables $x$ and $y$. For instance, $x$ and $y$ are statistically independent when their mutual information is zero ($I(x; y) = 0$). It then follows directly from the relation between mutual information and entropy that $H(x) = H(x|y)$, *i.e.* the entropy of $x$ does not decrease when conditioned on $y$ (and vice versa). Alternatively, the statistical independence of $x$ and $y$ is also evidenced by $p(x, y) = p(x)p(y)$, which is in turn equivalent to $p(x) = p(x|y)$. In this work, we investigate the statistical independence by verifying that the probability distribution of $x$, conditioned on a specific instantiation of $y^*$, is independent of that $y^*$. In other words, we verify that

$$\forall y^* : p(x|y^*) = p^* \tag{1}$$

with $p^*$ some constant probability distribution independent of $y^*$. It is easy to show that statistical independence follows from this:

$$p(x) = \sum_{y^*} p(x|y^*)p(y^*) = p(x|y^*) \sum_{y^*} p(y^*) = p(x|y^*) \tag{2}$$

**Information-theoretic view.** In [GM10, Def. 4], Gammel and Mangard provided an information-theoretic definition for $d$-probing security. We repeat their definition below for completeness as throughout this paper, we also use information-theoretic notions. Gammel and Mangard trace back the origin of this metric to Siegenthaler's correlation immunity of a Boolean function [Sie84].

***d*-probing security.** *A circuit with secret $x$ is $d$-probing secure, if and only if for any observation set of $d$ wires $\mathcal{Q} = (q_1, q_2, \ldots, q_d)$ the following condition holds:*

$$I((q_1, q_2, \ldots, q_d); x) = 0 \tag{3}$$

This condition is sufficient and necessary for $d$-probing security. However, it does not account for glitches. In the rest of this section, we fill in this gap.

## 2.2 Glitchy Circuits

Before providing the condition, we first discuss our circuit and leakage model.

**Circuit model.** Assume we deal with a masked circuit as shown in Figure 1. The circuit handles a sensitive value $x$ which depends on the key. Every combinational block $C_i$ computes a single output wire $q_i$ from a set of input wires $\mathcal{R}_i = \{r_{i_1}, r_{i_2}, \ldots\}$. For example, in Figure 1, wire $q_{w_1+1}$ is computed by combinational block $C_{w_1+1}$ from inputs $\mathcal{R}_{w_1+1} = \{r_1, r_2\}$. Note that the wires $(r_i, q_i)$ can be constants, shares of multiple sensitive variables, or public values. Their specific roles do not matter here. The output wires of combinational blocks $(q_i)$ are input wires to synchronization points and carry unstabilized values. For brevity, we assume that these synchronization points are simultaneously clocked registers. Data from one register stage $(r_i)$ forms the input to a block of combinational logic that computes the input to the next register stage. These inputs to combinational blocks are considered *stable*. Since glitch-extended probes of intermediate wires inside a block $C_i$ are obviously included in the glitch-extended probe of wire $q_i$, we only consider probes on $q_i$ from now on.

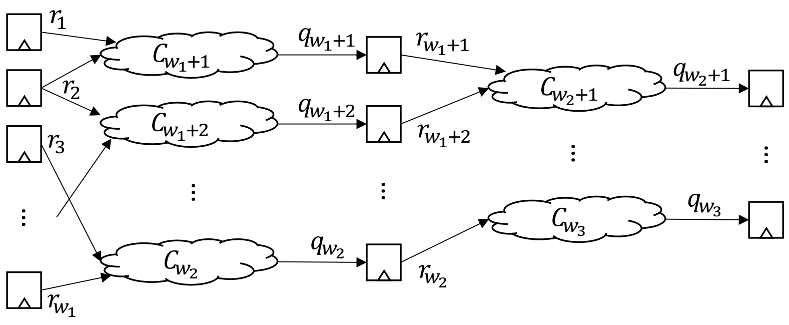

**Figure 1:** Masked circuit model

**The glitch function.** It is inefficient or even infeasible to predict the effect of glitches and enumerate all the possible temporary values that can occur on a wire in $C_i$ before the signal stabilizes at its intended function value. We provide an abstraction for this glitch-based leakage with a *glitch function.* The glitch function is *any* function leaked by a combinational block before it stabilizes on the intended function value. This can be an unexpected partial result because of propagation delays in some inputs. Alternatively, upon arrival of a late intermediate, one partial result transitions to another and the glitch function may compute the distance between both.

It is difficult to model the glitch function, since it depends on many variables, such as the logic library and synthesis process; hence, in this paper we consider it to be unknown. However, we can state some of its properties. An obvious, yet important property of the glitch function is that, under the independent leakage assumption, it depends exclusively on the inputs of the intended function. That is, when a wire in a combinational block $C_i$ glitches, it momentarily computes a glitch function that depends on $\mathcal{R}_i$ exclusively.

**Leakage model.** A model that takes into account glitches considers additional leakage compared to the traditional ISW $d$-probing model [ISW03]. We use the leakage model introduced in [RBN$^+$15] which covers all possible glitch functions as an abstraction: We assume that, whenever the adversary probes any single value within a combinational function, he also automatically obtains all inputs to the function (up to the last synchronization point), *for free.* That is, an adversary probing $q_i$ obtains the knowledge of the set $\mathcal{R}_i$. We provide the adversary with all inputs, so that he himself can compute the *worst possible glitch function.* The memory elements form the boundaries of the glitch extension and transform glitch-extended probes (consisting of multiple single-bit probes from the previous stage) into single-bit probes.

Our glitch model considers a worst-case scenario for the propagation of glitches. Šijačić *et al.* [SBY$^+$18] perform a post-place-and-route simulation of a circuit, which includes propagation delays and therefore also simulates the occuring glitches. Their model thus bares more resemblance to a realistic adversary compared to ours. However, as stated by Bertoni *et al.* in [BM16], two equal devices might exhibit a different behaviour in terms of glitches in practice. The latter therefore assume that every possible intermediate value of a function may occur due to a glitch. While Bertoni *et al.* use so-called *transients* to record exactly all possible transitions on a wire, we construct *glitch-extended probes*, which implicitly include these transitions, as well as other functions such as the differences between intermediates.

Our model thus assumes the very worst case of leaks and may be somewhat over-conservative, since this information might not actually be leaked by the circuit. This is the price we pay to work at a high level, where no information about the synthesis process, placing and routing or technology is required.

We note that this adversary model, here-on called *d-glitch-extended probing adversary*, has been considered in previous works [RBN$^+$15, FGMDP$^+$18], but without mathematical formalization. In [FGMDP$^+$18], it was also extended to cover other physical defaults, such as cross-talk. For now, we continue with the above described independent leakage model, including glitches but no coupling effects or register transitions. For its formal definition, we refer to that of the (1,0,0)-robust $d$-probing model of [FGMDP$^+$18].

## 2.3 A Sufficient Condition for $d$-Glitch-Extended Probing Security

In what follows, we introduce a new condition that ensures security against side-channel attacks in the presence of glitches. This security notion is elegant and conceptually simple, as well as easy to verify for circuits in practice.

**Property 1** ($d$-glitch immunity)**.** A circuit such as that in Figure 1 with sensitive data $x$ is $d$-glitch immune if and only if for any observation set of $d$ wires $(q_{i_1}, q_{i_2}, \ldots, q_{i_d})$ with respective glitch-extended probes $(\mathcal{R}_{i_1}, \mathcal{R}_{i_2}, \ldots, \mathcal{R}_{i_d})$, the following condition holds:

$$I(\mathcal{R}_{i_1} \cup \mathcal{R}_{i_2} \cup \ldots \cup \mathcal{R}_{i_d}; x) = 0 \tag{4}$$

**Lemma 1.** *A d-glitch immune circuit is d-glitch-extended probing secure.*

The reasoning is quite elementary: when the mutual information between $\mathcal{R} = \mathcal{R}_{i_1} \cup \mathcal{R}_{i_2} \cup \ldots \cup \mathcal{R}_{i_d}$ and the secret is zero, no combination of input wires $r_i \in \mathcal{R}$ can provide any information on the secret. Hence, no glitch function on $C_{i_1}, .., C_{i_d}$ or their combination, no matter the exact shape, can reveal any secret information. In particular, DPA would not be able to exploit leakage from it.

**Multi-variate security.** Note that $d$-glitch immunity puts no limitation on the register stage of probed wires $q_i$. Therefore, verification of higher-order security, let it be uni-variate or multi-variate, is conceptually as simple as verification of first-order security. We note that while glitch-extended probes have appeared in previous works [RBN$^+$15, FGMDP$^+$18], this is the first formal description of how to combine them to verify higher-order security. When testing security against $d$ glitch-extended probes, there are no functions (such as centered product or absolute difference) which manage to combine the probes without losing any information available to an adversary. The information-theoretic formulation in Equation (4) provides an almost trivial solution to this problem since it can consider the input wires of $d$ blocks jointly by mere concatenation of the extended probes. Hence, $d$-glitch immunity easily covers multi-variate security. This is in contrast with state-of-the-art concepts such as higher-order non-completeness and the TVLA t-test framework, which are uni-variate in nature.

**Sufficient and necessary.** Note that we only take into account the inputs to the combinational blocks $C_i$ and not the specific functions computed. On the one hand, this makes $d$-glitch immunity conceptually simple and quite easy to verify. On the other hand, it is a worst-case indication of leaks. A non-zero mutual information $I(\mathcal{R}; x) \neq 0$ points to the existence of some function on the inputs $\mathcal{R}$ that leaks sensitive information, but there is no guarantee that this function can occur as a glitch function on the circuit that implements $C_i$. Even if it does, the presence of such a leak does not imply that it can be exploited. Glitch immunity might therefore be over-conservative when one considers the implementation details (including the floorplan, temperature, etc.) and realistic noisy attack scenarios, but in this adversary model, it is both necessary and sufficient.

**Model Extension.** The title of this work refers to "hardware masking" in general even though we only discussed one particular hardware defect so far, *i.e.* glitches. However, the relevance of this work is not restricted to only this model, if we consider more different ways to redefine a probe. The information-theoretic framework allows us to extend the $d$-probing model with glitches while maintaining the same security condition (Equation (3)), by essentially replacing normal probes with glitch-extended probes (Equation (4)). The same can be done with other probe definitions, to obtain a security condition for other models, which take into account more physical effects. For example, in [FGMDP+18], two other types of probe extensions are discussed: transitions and couplings:

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

Another (slightly less restrictive) possibility is to include transitions by means of the XOR between two consecutively stored values of a memory cell. This would correspond to an assumption of Hamming distance leakage.

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

By replacing or complementing the glitch-extended probes $\mathcal{R}$ in condition (4) with transition-extended and/or coupling-extended probes (or others), the information-theoretic framework can be brought to many more hardware and software models. The model definition only requires a probe definition, which accurately reflects the information available to the adversary. In the remainder of this work, we continue only with glitch-extended probes for the sake of clarity.

## 3 Threshold Implementations

Traditionally, the provable security of threshold implementations (TI) [NRR06] against first-order attacks relies on three conditions:

**T1 Correctness.** The masked function should compute a masked representation of the correct unmasked output. This property does not relate to security.

**T2 Non-completeness.** Any share of the masked function must be independent of at least one input share. This property is central to security in the presence of glitches.

**T3 Uniformity.** The masked function uses a uniform sharing of the input and transforms this input into a uniform sharing of the output. This property is especially important when composing masked blocks.

Recall the definition of a uniform sharing with $Sh(x)$ the set of all valid sharings of $x$:

**Uniform sharing [Bil15].** *A sharing $\boldsymbol{x}$ is uniform if and only if there exists a constant $p^*$ such that for all $x$ we have:*

$$p(\boldsymbol{x}|x) = \begin{cases} p^* & \text{if } \boldsymbol{x} \in Sh(x) \\ 0 & \text{else} \end{cases} \qquad \text{and} \qquad \bigoplus_{\boldsymbol{x} \in Sh(x)} p(\boldsymbol{x}) = p(x) \tag{5}$$

In the rest of this section, we discuss the relation between $d$-glitch immunity and the TI conditions. In particular, we elaborate on the necessity and satisfactoriness of these conditions for first order and higher orders. From these relations, we can interpret $d$-glitch immunity as being a generalization of the TI properties.

## 3.1 Non-Completeness and Uniformity Imply 1-Glitch Immunity

It is a well-known result that T2 and T3 imply first-order security in the presence of glitches, since this was proven in [NRR06]. Hence, T2 and T3 together are sufficient for 1-glitch-extended probing security. In what follows, we show that a circuit satisfying T2 and T3 also fulfills 1-glitch immunity.

**Lemma 2.** *A circuit satisfies* 1*-glitch immunity if it satisfies non-completeness (T2) and uniformity (T3).*

*Proof.* The assumptions for each stage of TI are that the input sharing $\boldsymbol{x}$ is uniform and that each wire $q_i$ at the end of that stage must be independent of at least one input share. Without loss of generality (wlog) we assume that $q_i$ is independent of $x_i$ and thus

$$\mathcal{R}_i = \boldsymbol{x}_{\bar{i}} \tag{6}$$

Lemma 5 of [Bil15] states that $\boldsymbol{x}_{\bar{i}}$ and the secret $x$ are independent for any choice of $i$ if the masking $\boldsymbol{x}$ is uniform. Hence,

$$I(\boldsymbol{x}_{\bar{i}}; x) = 0 \tag{7}$$

Combining (6) together with (7) forms (4): $I(\mathcal{R}_i; x) = 0$. □

## 3.2 Glitch Immunity Implies Non-Completeness

The concept of non-completeness was extended to higher orders by Bilgin *et al.* [BGN+14].

**$d^{\text{th}}$-order non-completeness [Bil15].** *Any combination of up to $d$ component functions $F_i$ of a shared function $\boldsymbol{F}$ must be independent of at least one input share.*

Clearly the non-completeness definition aligns with the glitching adversary of this work as the component functions $F_i$ correspond to the combinational block functions $C_i$ calculated from register to register.

**Lemma 3.** *$d$-glitch immunity implies $d^{th}$-order non-completeness.*

*Proof.* We use a simple proof by contraposition. Suppose that non-completeness is not fulfilled, *i.e.* there exists a set of $d$ blocks $(C_{i_1}, C_{i_2}, \dots, C_{i_d})$ that jointly depend on all input shares $\boldsymbol{x}$. Since $x = \bigoplus_i x_i$, we clearly have then that $I(\mathcal{R}_{i_1} \cup \mathcal{R}_{i_2} \cup \ldots \cup \mathcal{R}_{i_d}; x) \neq 0$ and hence glitch immunity is not satisfied. □

## 3.3 A Sufficient Condition for Higher-Order Security

We have demonstrated the well-known fact that T2 and T3 together form a sufficient condition for first-order security from an information-theoretic point-of-view. Moreover, we have shown that non-completeness is a necessary condition for higher-order security in this adversary model, as expected. These relations are summarized in Figure 2. However, as presented by Reparaz *et al.* [RBN+15], $d^{\text{th}}$-order non-completeness even together with uniformity is not sufficient for circuits consisting of multiple register stages. This is mainly due to the fact that TI conditions focus on the circuit behavior of a single stage and when higher-order security is considered, this becomes a disadvantage. In contrast, $d$-glitch immunity does not have such a limitation as it considers multiple stages. This observation immediately brings up the idea to extend the TI uniformity condition to cover not only a single stage but multiple stages, for example by requiring the combination of any $d$ non-linear function inputs to be jointly uniform. We do not investigate this idea further in this paper. Instead, we show that uniformity is not a necessary condition to achieve security.

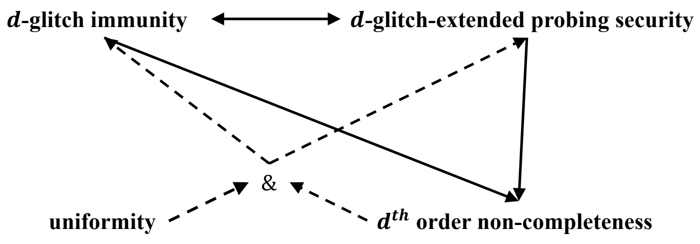
d-glitch immunity
d-glitch-extended probing security
&
uniformity
d^th order non-completeness

**Figure 2:** Illustration of the relations between different security notions for hardware security. The arrows indicate an implication relation. When the line is dotted, it is only valid for security order $d = 1$. The full lines are for any order $d$.

## 3.4 (Un)Necessity of Uniformity

Glitch immunity does not imply non-completeness and uniformity (simultanenously). In fact, uniformity is not a necessary condition. The only requirement for input sharings is included in Eqn. (4); that is, the entropy of $d$ input shares does not decrease when conditioned on the secret. In other words, the mutual information of a set of any $d$ input shares with the shared secret $x$ must be zero:

$$I((x_{i_1}, x_{i_2}, \ldots, x_{i_d}); x) = 0 \tag{8}$$

The use of uniform gadgets does make composition easier, as shown in [NRR06], but it is not strictly necessary.

It is therefore possible to have non-uniform mappings that satisfy Property 1. We show this in Appendix A with a case study of non-uniform AND gates. While previous authors [RBN+15, FGMDP+18] already reported the fact that uniformity is not sufficient, its nonnecessity has not been stated nor demonstrated in any published work. Note that in the case of a $d+1$-sharing, the condition becomes $I(\boldsymbol{x}_{\bar{i}}; x) = 0$, $\forall i$, which implies uniformity. Hence, uniformity *is* a necessary condition for $d+1$-sharings.

## 4 Non-Interference

When circuits get large, it is infeasible to verify probing security by exhaustive probing. For this reason, the concept of strong non-interference has been introduced for masked software implementations. In this section, we unify the treatment of this security notion with glitch immunity (Property 1) in order to achieve a formal definition for composability in the presence of glitches. The probes in the following definitions are instantaneous (*i.e.* no glitches). We consider a gadget with a $d+1$-input sharing $\boldsymbol{x}$.

**Definition 1** (Simulatability [BBP+16])**.** *A set $\mathcal{Q} = \{q_1, \ldots, q_l\}$ of $l$ probes can be simulated with at most $t$ shares of each input, if there exists a set $\mathcal{I} = i_1, \ldots, i_t$ of $t$ indices in $\{0, \ldots, d\}$ and a random function $S$ with $t$ inputs and $l$ outputs such that for any fixed input shares $(x_i)_{0 \le i \le d}$, the distributions $\{q_1, \ldots, q_l\}$ and $\{S(x_{i_1}, \ldots, x_{i_t})\}$ are identical.*

**$d$-non-interference (NI) [BBP+16].** *A gadget is d-non-interferent (d-NI) if and only if every set of at most d probes can be simulated with at most d shares of each input.*

**$d$-strong non-interference (SNI) [BBP+16].** *A gadget is d-strong non-interferent (d-SNI) if and only if for every set $\mathcal{I}$ of $t_1$ probes on intermediate variables (i.e., no output wires or shares) and every set $\mathcal{O}$ of $t_2$ probes on output shares such that $t_1 + t_2 \le d$, the* *set $\mathcal{I} \cup \mathcal{O}$ of probes can be simulated using at most $t_1$ shares of each input.*

When a gadget is $d$-NI, it is also $d$-probing secure. However, probing security (and $d$-NI) is not composable and verifying a circuit by simulating probes on every single wire is not scalable. The authors of [BBD$^+$16] therefore propose to build each circuit from composable gadgets. A gadget is considered composable if it is $d$-SNI. This essentially means the following two things:

- If a gadget is $d$-SNI, it is $d$-NI and thus $d$-probing secure.
- Any circuit built from $d$-SNI gadgets, is $d$-NI and thus $d$-probing secure.

The use of composable (SNI) gadgets allows a divide-and-conquer approach for the security verification of large circuits. The ease of verification comes at the cost of higher implementation and randomness cost since the demands for strong non-interference are quite high. It has been shown that depending on the function graph, it is sufficient for some gadgets to be NI, rather than SNI [BBP$^+$16]. Nevertheless, NI and SNI are both sufficient but not necessary conditions for probing security of masked software implementations. This is illustrated by first-order threshold implementations [NRR06], which are 1-probing secure but neither 1-NI, nor 1-SNI. In addition, SNI is a sufficient condition for the composability of gadgets, but it is neither necessary for probing security, nor for composability.

The concepts of NI and SNI were originally defined without regard for hardware defaults such as glitches. In [FGMDP$^+$18], the authors provide a *robust* $d$-(S)NI definition as follows in order to extend the definition for glitchy circuits.

**$(g, t, c)$-robust $d$-probing secure (or $d$-NI/SNI)** *circuits are secure in the $d$-probing model (or $d$-NI/SNI) with an adversary whose probes are extended with glitches if $g = 1$, with transitions if $t = 1$ and with couplings if $c \geq 1$.*

In this paper we only consider glitches, *i.e.* $(1, 0, 0)$-robust $d$-probing. Despite the existence of the concept of robust SNI, it remains unclear how to automate the verification of composability of hardware gadgets, as it is unclear how to define a single mathematical equation.[1]

In this section, we first redefine (S)NI for ideal circuits from an information-theoretic point of view. This eventually allows to extend the definitions for the presence of glitches. For simplicity, we assume from now on that the masked implementation uses $d + 1$ shares even though similar treatment is possible for more shares.

**A Note on Fields.** The definition of simulatability in [BBP$^+$16] considers a simulator with $t$ input *bits* and $l$ output *bits*. This is contradictory to how the notions of (strong) non-interference are applied in proofs and in tools such as [Cor18, BBFG18], where simulations are typically performed at the word-level in some field $\mathbb{GF}(2^n)$. While a formal verification in a larger field implies also formal verification over bits, the reverse is not true. We would like to see this distinction made more clear in the future, along with the corresponding assumptions about the underlying hardware, *i.e.* are bits assumed to leak independently or jointly grouped by words? We discuss for example bit-slicing as an example in § 5.2.

## 4.1 Redefining Non-Interference

We consider $\mathcal{Q}$ any set of at most $d$ probes in some field $\mathbb{GF}(2^n)$ in a gadget with input sharing $\boldsymbol{x}$. According to the NI definition, the view of an adversary that probes $\mathcal{Q}$ should be perfectly simulatable by a simulator that has access to only $d$ of the $d + 1$ input shares. The view of the adversary is created using knowledge of all input shares $\boldsymbol{x}$, *i.e.* we write

[1] The work of [BBFG18], which has been done independently and simultaneously, bypasses the need for a mathematical formula and does the automatic verification by symbolic manipulation, thereby limiting the scope of their tool, as we will see in the next section.

it as $p(\mathcal{Q}|\boldsymbol{x})$. This is the distribution of the probes $\{q_1, \ldots, q_l\}$ from Definition 1 and must thus be identical to the distribution of the output of some simulator $S(x_{i_1}, \ldots, x_{i_t})$. Without loss of generalization (wlog), we assume that the simulator uses all input shares except $x_i$. The view of the simulator can hence be written as $p(\mathcal{Q}|\boldsymbol{x}_{\bar{i}})$. An important detail in the definition of simulatability is that all input shares $(x_i)_{0 \leq i \leq d}$ must be fixed. Hence, the probability distributions are over the randomness that is used in the gadget only.

We therefore consider the following property:

**Property 2.** A gadget with $d+1$ input shares $\boldsymbol{x}$ satisfies Property 2 if and only if for any observation set of at most $d$ probes $\mathcal{Q} \in \mathbb{GF}(2^n)^d$, the following condition holds:

$$\exists i : p(\mathcal{Q}|\boldsymbol{x}) = p(\mathcal{Q}|\boldsymbol{x}_{\bar{i}}) \Leftrightarrow \exists i : I(\mathcal{Q}; x_i|\boldsymbol{x}_{\bar{i}}) = 0 \tag{9}$$

where the probability is over the random coins used in the gadget.

In other words, each probe set $\mathcal{Q}$ and at least one share $x_i$ must be conditionally independent given the other shares $\boldsymbol{x}_{\bar{i}}$. It is clear that $d$-non-interference and Property 2 are equivalent, when one considers the definition of simulatability.

**Lemma 4.** *A gadget with $d+1$ input shares $\boldsymbol{x}$ is d-NI over $\mathbb{GF}(2^n)$ if and only if it satisfies Property 2, i.e. d-non-interference and Property 2 are equivalent.*

*Proof.*

- $\Rightarrow$ We prove by contraposition that $d$-NI implies Property 2: if there exists a set $\mathcal{Q}$ for which Eqn. (9) is not true, *i.e.* $\forall i : p(\mathcal{Q}|\boldsymbol{x}_{\bar{i}}) \neq p(\mathcal{Q}|\boldsymbol{x})$, then there does not exist a simulator $S$ and an input set $\mathcal{I}$ of at most $d$ indices such that the distributions of $\mathcal{Q}$ and $S(x_\mathcal{I})$ are identical, *i.e.* we cannot simulate $\mathcal{Q}$ using only $d$ shares and thus the gadget is not $d$-NI.
- $\Leftarrow$ In the other direction, Property 2 implies the existence of a simulator $S$ and an input set $I = \boldsymbol{x}_{\bar{i}}$ such that the distributions of $\mathcal{Q}$ and $S(x_\mathcal{I})$ are identical, *i.e.* Property 2 implies $d$-non-interference.

□

It is well known that $d$-non-interference is stronger than $d$-probing security, *i.e.* $d$-NI implies Eqn. (3) for any set $\mathcal{Q}$. The same can be said for Property 2:

**Lemma 5.** *Property 2 implies d-probing security over $\mathbb{GF}(2^n)$, i.e.*

$$\exists i : I(\mathcal{Q}; x_i|\boldsymbol{x}_{\bar{i}}) = 0 \Rightarrow I(\mathcal{Q}; x) = 0 \tag{10}$$

*Proof.* We first present two intermediate results:

I1. $p(\boldsymbol{x}_{\bar{i}}|x) = p(\boldsymbol{x}_{\bar{i}})$; This is obviously a necessary requirement for any $d+1$-sharing of the input. In particular, it has been shown to be automatically true when the input sharing is uniform in [Bil15, Lemma 5]. It also follows from glitch immunity (see eqn (8)).

I2. $p(\mathcal{Q}|\boldsymbol{x}_{\bar{i}}, x) = p(\mathcal{Q}|\boldsymbol{x}, x) = p(\mathcal{Q}|\boldsymbol{x})$; The equality follows from the redundant information in $(\boldsymbol{x}, x)$, since $x = \sum_i x_i$. Furthermore, since Property 2 implies that $p(\mathcal{Q}|\boldsymbol{x}) = p(\mathcal{Q}|\boldsymbol{x}_{\bar{i}})$, it follows that $p(\mathcal{Q}|\boldsymbol{x}_{\bar{i}}, x) = p(\mathcal{Q}|\boldsymbol{x}_{\bar{i}})$.

We can now prove (10) by showing the statistical independence of $\mathcal{Q}$ and $x$, given I1 and I2 (*i.e.* given Property 2). Note that probing security does not require fixed input shares so the probability here is over the randomness used both in the circuit and in the initial sharing of $x$.

$$\begin{aligned}
p(\mathcal{Q}|x) &= \sum_{\boldsymbol{x}^*_{\bar{i}}} p(\mathcal{Q}, \boldsymbol{x}^*_{\bar{i}}|x) \\
&= \sum_{\boldsymbol{x}^*_{\bar{i}}} p(\mathcal{Q}|\boldsymbol{x}^*_{\bar{i}}, x) p(\boldsymbol{x}^*_{\bar{i}}|x) \\
&= \sum_{\boldsymbol{x}^*_{\bar{i}}} p(\mathcal{Q}|\boldsymbol{x}^*_{\bar{i}}) p(\boldsymbol{x}^*_{\bar{i}}) && \leftarrow \text{(using I1 and I2)} \\
&= \sum_{\boldsymbol{x}^*_{\bar{i}}} p(\mathcal{Q}, \boldsymbol{x}^*_{\bar{i}}) \\
&= p(\mathcal{Q})
\end{aligned}$$

□

**For uniform input sharings.** When the input sharing $\boldsymbol{x}$ is uniform, $d$-non-interference implies a stronger and more simple property.

**Property 3.** A gadget with $d+1$ input shares $\boldsymbol{x}$ satisfies Property 3 if and only if for any observation set of at most $d$ probes $\mathcal{Q} \in \mathbb{GF}(2^n)^d$, the following condition holds:

$$\exists i : I(\mathcal{Q}; x_i) = 0 \tag{11}$$

In other words, each set of probes $\mathcal{Q}$ must be *marginally* independent of at least one input share $x_i$.

**Lemma 6.** *A gadget with a* uniform $d+1$ *input sharing $\boldsymbol{x}$ is $d$-NI over $\mathbb{GF}(2^n)$ only if it satisfies Property 3, i.e. for each probe set $\mathcal{Q} \in \mathbb{GF}(2^n)^d$:*

$$\exists i : I(\mathcal{Q}; x_i|\boldsymbol{x}_{\bar{i}}) = 0 \Rightarrow \exists i : I(\mathcal{Q}; x_i) = 0$$

*Property 3 is thus a necessary but not sufficient condition for NI.*

*Proof.* In the case of a uniform input sharing, we can state that disjunct sets of shares (*e.g.* $\boldsymbol{x}_{\bar{i}}$ and $x_i$) are independent, *i.e.* $p(\boldsymbol{x}_{\bar{i}}|x_i) = p(\boldsymbol{x}_{\bar{i}})$. We use Lemma 4 to show that in that case, $d$-NI implies Property 3.

$$\begin{aligned}
p(\mathcal{Q}|x_i) &= \sum_{\boldsymbol{x}^*_{\bar{i}}} p(\mathcal{Q}, \boldsymbol{x}^*_{\bar{i}}|x_i) \\
&= \sum_{\boldsymbol{x}^*_{\bar{i}}} p(\mathcal{Q}|\boldsymbol{x}^*_{\bar{i}}, x_i) p(\boldsymbol{x}^*_{\bar{i}}|x_i) \\
&= \sum_{\boldsymbol{x}^*_{\bar{i}}} p(\mathcal{Q}|\boldsymbol{x}^*_{\bar{i}}) p(\boldsymbol{x}^*_{\bar{i}}) && \leftarrow \text{(using Property 2 (NI) and uniformity)} \\
&= \sum_{\boldsymbol{x}^*_{\bar{i}}} p(\mathcal{Q}, \boldsymbol{x}^*_{\bar{i}}) \\
&= p(\mathcal{Q})
\end{aligned}$$

□

## 4.2 Redefining *Strong* Non-Interference

The NI and SNI definitions are very similar apart from the number of input shares that can be used in the simulation. We can thus likewise redefine strong non-interference.

**Property 4.** Consider a gadget with $d+1$ input shares $\boldsymbol{x}$. Let $\mathcal{Q}$ be any observation set of at most $d$ probes in $\mathbb{GF}(2^n)$ of which $t_1(\mathcal{Q})$ are intermediates and $t_2(\mathcal{Q})$ are output probes such that $t_1(\mathcal{Q}) + t_2(\mathcal{Q}) \leq d$. The gadget satisfies Property 4 if and only if for any such $\mathcal{Q}$ the following condition holds:

$$\exists T \subset \{0, \ldots, d\} \text{ with } |T| = t_1(\mathcal{Q}) \text{ such that } I(\mathcal{Q}; \boldsymbol{x}_{\bar{T}} | \boldsymbol{x}_T) = 0 \tag{12}$$

where the probability is over the random coins used in the gadget.

**Lemma 7.** *A gadget with $d+1$ input shares $\boldsymbol{x}$ is d-SNI over $\mathbb{GF}(2^n)$ if and only if it satisfies Property 4, i.e. d-strong non-interference and Property 4 are equivalent.*

The proof is very similar to that of Lemma 4. We leave it to the reader.

When no outputs are probed and thus $t_2(\mathcal{Q}) = 0$, then Eqn. (12) aligns with Eqn. (9). For example, when $d = 2$, the tuples of probes can be divided into three groups.

- for each tuple of probes $\mathcal{Q}$ for which $t_2(\mathcal{Q}) = 0$, verify that $\exists i : I(\mathcal{Q}; x_i | \boldsymbol{x}_{\bar{i}}) = 0$.
- for each tuple of probes $\mathcal{Q}$ for which $t_2(\mathcal{Q}) = 1$, verify that $\exists i \neq j : I(\mathcal{Q}; (x_i, x_j) | \boldsymbol{x}_{\bar{i,j}}) = 0$ or equivalently $\exists i : I(\mathcal{Q}; \boldsymbol{x}_{\bar{i}} | x_i) = 0$.
- for each tuple of probes $\mathcal{Q}$ for which $t_2(\mathcal{Q}) = 2$, verify that $I(\mathcal{Q}; \boldsymbol{x}) = 0$

When the input sharing is uniform, we can assume that for any set of indices $T \subset \{0, \ldots, d\}$, $\boldsymbol{x}_T$ is independent from $\boldsymbol{x}_{\bar{T}}$, *i.e.* two disjunct sets of shares are independent of each other. Thus similarly to Lemma 6, $d$-strong non-interference with a uniform input sharing implies marginal independence of $Q$ and some input set $\boldsymbol{x}_{\bar{T}}$, *i.e.*

$$\exists T \subset \{0, \ldots, d\} \text{ with } |T| = t_1(\mathcal{Q}) \text{ such that } I(\mathcal{Q}; \boldsymbol{x}_{\bar{T}}) = 0$$

## 4.3 Towards Strong Non-Interference in the Presence of Glitches

These information-theoretic definitions are easily extended to include security in the presence of glitches. Following the adversary model of this work, we should only replace each probed wire $q_i$ with its glitch-extended probe $\mathcal{R}_i$.

**Property 5.** Consider a gadget with $d+1$ input shares $\boldsymbol{x}$. Let $\mathcal{Q} = (q_{i_1}, q_{i_2}, \ldots)$ be any observation set of at most $d$ wires and let $\mathcal{R} = \mathcal{R}_{i_1} \cup \mathcal{R}_{i_2} \cup \ldots$ be the corresponding glitch-extended probe. Let $t_1(\mathcal{Q})$ be the number of intermediate wires in $\mathcal{Q}$ and $t_2(\mathcal{Q})$ the number of output wires in $\mathcal{Q}$ such that $t_1(\mathcal{Q}) + t_2(\mathcal{Q}) \leq d$. The gadget satisfies Property 5 if and only if for any such $\mathcal{Q}$ the following condition holds:

$$\exists T \subset \{0, \ldots, d\} \text{ with } |T| = t_1(\mathcal{Q}) \text{ such that } I(\mathcal{R}; \boldsymbol{x}_{\bar{T}} | \boldsymbol{x}_T) = 0 \tag{13}$$

**Lemma 8.** *A gadget with $d+1$ input shares $\boldsymbol{x}$ is d-SNI in the presence of glitches over $\mathbb{GF}(2^n)$ if it satisfies Property 5.*

From now on, we refer to Property 5 as $d$-Glitch Strong Non-Interference ($d$-GSNI). It also corresponds to (1,0,0)-robust $d$-SNI as defined in [FGMDP+18].

**How to use $d$-GSNI?** We illustrate Lemma 8 by adopting one of the examples of Faust *et al.* [FGMDP+18]. Consider the following well-known second-order secure multiplier of three shares $(x_0, x_1, x_2)$ and $(y_0, y_1, y_2)$: In the first stage, nine products $x_i y_j$ are computed, of which only the cross-terms are remasked:

$$\begin{array}{lll} t_{0,0} = x_0 y_0 & t_{1,0} = x_1 y_0 \oplus r_1 & t_{2,0} = x_2 y_0 \oplus r_2 \\ t_{0,1} = x_0 y_1 \oplus r_1 & t_{1,1} = x_1 y_1 & t_{2,1} = x_2 y_1 \oplus r_3 \\ t_{0,2} = x_0 y_2 \oplus r_2 & t_{1,2} = x_1 y_2 \oplus r_3 & t_{2,2} = x_2 y_2 \end{array} \tag{14}$$

**Table 1:** Scaled probability distributions of the glitch-extended probe of $\{[z_0]_{reg}, t_{0,1}\}$

| | | $p(z_0, x_0, y_1 \| \boldsymbol{x}^*, \boldsymbol{y}^*)$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $(y_1^*, x_0^*)$ | $(x_1^*, x_2^*, y_0^*, y_2^*)$ | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| (0,0) | * | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| (0,1) | * | 0 | 0 | 4 | 4 | 0 | 0 | 0 | 0 |
| (1,0) | * | 0 | 0 | 0 | 0 | 4 | 4 | 0 | 0 |
| (1,1) | * | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 |

These intermediate nine shares $t_{i,j}$ are stored in a register. In the next stage, they are compressed into three output shares.

$$\begin{aligned} z_0 &= [t_{0,0}]_{reg} \oplus [t_{0,1}]_{reg} \oplus [t_{0,2}]_{reg} \\ z_1 &= [t_{1,0}]_{reg} \oplus [t_{1,1}]_{reg} \oplus [t_{1,2}]_{reg} \\ z_2 &= [t_{2,0}]_{reg} \oplus [t_{2,1}]_{reg} \oplus [t_{2,2}]_{reg} \end{aligned} \tag{15}$$

Next, as argued in [FGMDP+18, §5.2], it is necessary to store also these three shares into a register in order to achieve composability. Indeed, if we consider Eqn. (15) directly as output wires, then the multiplication is not $d$-GSNI. We verify this by enumerating every single input sharing $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. For each input sharing, we compute the probability distribution of the glitch-extended probes of an observation set of two outputs ($t_2 = 2$). For example, for observation set $\mathcal{Q} = \{z_0, z_1\}$, we investigate the distribution of $\mathcal{R} = \{t_{0,0}, t_{0,1}, t_{0,2}, t_{1,0}, t_{1,1}, t_{1,2}\}$. For Eqn. (13) to be satisfied, this distribution must be identical for each input sharing. This is not the case. On the other hand, the distribution of $\mathcal{Q}$ itself *is* identical for each input sharing $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, *i.e.* $I(\mathcal{Q}; (\boldsymbol{x}, \boldsymbol{y})) = 0$. Hence, a register is indeed needed before the output. In that case $\mathcal{R}$ is the extended probe of an observation set that corresponds to two *intermediate* wires, *i.e.* $t_2 = 0$. We enumerate all possible $(x_0^*, x_1^*, y_0^*, y_1^*)$. For each, we check that the probability distribution of $\mathcal{R}$ is identical for each $(x_2^*, y_2^*)$. This is the case, hence the conditional mutual information of the glitch-extended probe with input shares $(x_2, y_2)$ is zero: $I(\mathcal{R}; (x_2, y_2)|(x_0, x_1, y_0, y_1)) = 0$.

As a final example, consider the observation set $\mathcal{Q} = \{[z_0]_{reg}, t_{0,1}\}$ consisting of one output wire and one intermediate wire ($t_2 = 1$). Its glitch-extended probe is $\mathcal{R} = \{z_0, x_0, y_1\}$[2]. For each fixed $(x_0^*, y_1^*)$, we find that the probability distribution of $\{z_0, x_0, y_1\}$ is constant for each $(x_1^*, x_2^*, y_0^*, y_2^*)$. The distributions are shown in Table 1. Hence $I(\mathcal{R}; (x_1, x_2, y_0, y_2)|(x_0, y_1)) = 0$. Moreover, since we are working with uniform shares, it is implied that also $I(\mathcal{R}; (x_1, x_2, y_0, y_2)) = 0$. Indeed, the probability distribution $p(\mathcal{R}|x_1^*, x_2^*, y_0^*, y_2^*)$ is $(4, 4, 4, 4, 4, 4, 4, 4)$ for any $(x_1^*, x_2^*, y_0^*, y_2^*)$. This example does not tell us more than the proof already given in [FGMDP+18] but serves to illustrate that Property 5 could be used to create similar proofs in a more automated way. Note that in this example, we work over bits, *i.e.* $\mathbb{GF}(2^n)$ with $n = 1$.

**Conclusion.** In this section we brought the notions of (strong) non-interference into the information-theoretic framework for side-channel security. These new equations clarify the link with probing security and open up the possibility to port the notions to glitchy environments. Furthermore, the information-theoretic approach will allow to use these security concepts in more realistic settings that are on the one hand more noisy and on the other hand include more types of leakage (*e.g.* transition leakage, coupling, ...).

[2]The random input has no influence on the probability distributions.

## 5 Using Glitch Immunity to Detect Flawed Masking Schemes

In order to verify $d$-glitch immunity, one has to check whether each glitch-extended probe's distributions corresponding to different secrets are identical or not (cf. Eqn (1)).

This is essentially also done by the MaskVerif tool of Barthe *et al.* [BBFG18]. By combining it with a verification of the (S)NI properties, they avoid the prohibiting complexity of exhaustive search for all intermediates and achieve an efficient and powerful verification tool for both $d$-probing and $d$-glitch-extended probing security. However, their tool does not completely exploit the flexibility and variability of glitch immunity and is therefore not applicable to every circuit.

In this section, we demonstrate a proof-of-concept tool that uses $d$-glitch immunity to validate masked hardware circuits. Note that this section considers $d$-glitch-extended probing security only. This is in contrast, with MaskVerif [BBFG18], which also considers (glitch-)SNI. Since we do not incorporate the optimizations of [BBFG18], our performance is not competitive with theirs. We simply wish to demonstrate how glitch immunity can and should also be used to verify implementations, which are currently out of scope for [BBFG18].

Finally, we also describe how the condition of glitch immunity can be relaxed to be used for flaw detection and practical security evaluation in a way that is more effective and informative than in a noisy lab environment.

### 5.1 Description

Our tool takes the HDL description of a circuit directly as input and needs no other user-provided information apart from the required security order $d$. We describe two parts: a *preprocessing* step prepares the input for the *verification* step.

**Preprocessing.** The preprocessing step parses the HDL code and builds a software implementation (in C), which can simulate the entire circuit at bit level. Apart from the circuit itself, the preprocessing step also extracts from the netlist a list of all register inputs $q_i$ and the corresponding glitch-extended probes $\mathcal{R}_i$, *i.e.* the set of inputs to the combinational block $C_i$ that determines wire $q_i$. The C implementation is generated such that for a given circuit, it can compute the exact values of all registers $r_i$ and use these to construct samples for probability distributions. Per simulation $(k)$, one sample $sample_I^k$ is created for each unique $d$-tuple of observed probes $\mathcal{Q}_I = \{q_{i_1}, q_{i_2}, \dots, q_{i_d}\}$ and its corresponding set of glitch-extended probes $\mathcal{R}_I = \mathcal{R}_{i_1} \cup \mathcal{R}_{i_2} \dots \cup \mathcal{R}_{i_d}$. Concatenating the values on the wires in $\mathcal{R}_I$ results in a sample of bit-width $|\mathcal{R}_I|$. Different probe combinations thus result in samples of different widths, but the width of a probe combination is the same in each simulation. The probe $\mathcal{R}_I$ can be a combination of random inputs, public values and shares of various sensitive variables. The specific role of each wire is unimportant and need not be tracked.

**Verification.** The verification step uses the prepared software implementation to simulate the circuit for different inputs and to collect the samples of glitch-extended probes. The application keeps two histograms for each possible $d$-probe $\mathcal{Q}_I$. The histograms corresponding to different probes have a different support because of the variable length of the probes $|\mathcal{R}_I|$. In simulation $k$ of the circuit, the application receives one sample per probe ($sample_I^k$). The sample is added to one of the two histograms, depending on the value of the unshared (secret) input $x$. This is illustrated in Figure 3. A circuit with $K$ input wires requires $2^K$ simulations. When all simulations are complete, the two histograms correspond to exact joint probability distributions of $\mathcal{R}_I$ and are compared. If a probe is found for which the histograms do not match exactly, the circuit is not $d$-glitch immune, *i.e.* there is a dependency on the secret which *could* result in leakage of

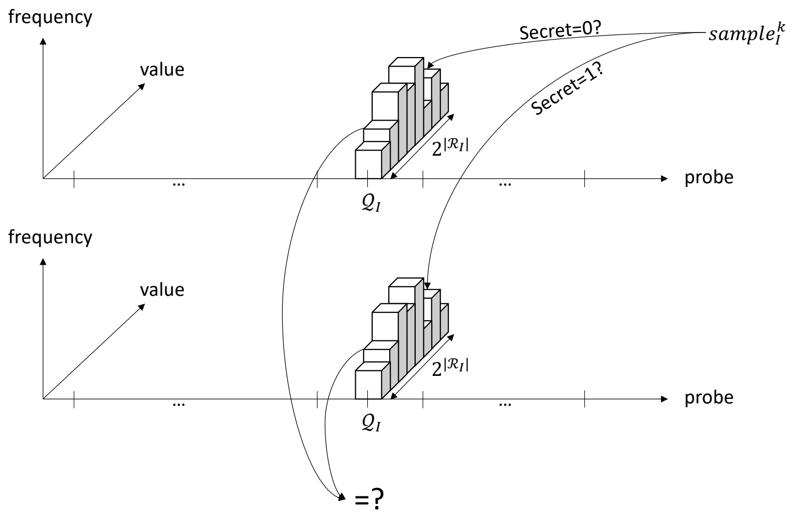

**Figure 3:** How to use glitch immunity to detect flaws.

sensitive information through some glitch function on those inputs. The circuit is *provably* $d$-glitch-extended probing secure if for each $d$-probe, the distributions are identical for each secret. Note that the functionality of this part is independent of the security order $d$. It builds and compares histograms based on variable-width samples received from the software implementation. Whether these samples were built from one, two or three probes in the preprocessing step, is of no consequence.

**Optimizations.** It is possible that different wires in the circuit have the same set of inputs ($\mathcal{R}_i = \mathcal{R}_j$ for $i \neq j$). Also by concatenating different glitch-extended probes for higher-order security, it is possible to obtain duplicates ($\mathcal{R}_i \cup \mathcal{R}_j = \mathcal{R}_k \cup \mathcal{R}_l$ for $i \neq j \neq k \neq l$). We avoid redundant samples by removing all but one copy of each $\mathcal{R}_I$ from the final list of probes.

The C implementation simulates the circuit at bit level, *i.e.* each multiple-bit variable has been split into single-bit variables and only bitwise operators (AND,OR,NOT,XOR) are used. This allows us to bitslice the simulation: using *for example* 32-bit integers, we can simulate the circuit for 32 different inputs in parallel on a single core. We do not put further effort into optimizations.

**Complexity.** As this methodology is completely exhaustive, the complexity naturally also increases quickly for growing circuits. Note however that the size of a circuit in this context should not be measured by the number of gates, as this does not influence the runtime of the verification. We identify three important factors that impact the complexity. Firstly, the number of inputs ($K$) to a circuit determines the number of simulations required ($2^K$). Next, naturally, the number of samples to simulate grows with the order of verification $d$. Finally, a glitch-extended probe of size $|\mathcal{R}|$ leads to a histogram of size $2^{|\mathcal{R}|}$. This affects the memory complexity of the verification and therefore also the timing. Of these three factors, the security order $d$ is the most important, since it also directly influences the other two. A higher security order typically results in more inputs and also in larger glitch-extended probes. We note again that our goal here is not to reduce the complexity. It has been shown in [BBFG18] that (S)NI verification can be used to relieve the cost of simulations in some cases.

**Examples.** We are able to exhaustively validate the $d$-glitch-extended probing security of small gates such as the Domain-Oriented-Masking (DOM) AND-gate [GMK16] (with 2 or 3 shares) and the first-order secure Keccak S-box from [GSM17] within 0.02 seconds. A 4-share DOM AND gate is verified within 0.3 seconds. It is much more interesting to look at a flawed example such as the higher-order threshold implementation described in [BGN$^+$14].

This higher-order secure KATAN construction has been discussed in depth in other works [SM15, RBN$^+$15, Rep16] and it is well known that it exhibits a multi-variate flaw. In particular, the authors of [RBN$^+$15] claim that the secret is leaked when the construction is iterated and one combines probes from cycle 1 and cycle 7. Using the above described tool, we now find that multiple iterations are not needed and that multi-variate leakage of the secret occurs even within one iteration of the round function, i.e. by combining probes from cycle 1 and cycle 2. We recall the mini-cipher described in [RBN$^+$15, Eq. (5)], which targets second-order security. The round function receives three inputs $a, b, c$, each in five shares. The circuit computes a five-share representation of $d = ab \oplus c$. This is done in two stages. In the first step, $d$ is computed in ten shares:

$$\begin{aligned}
d_0 &= c_1 \oplus a_1b_1 \oplus a_0b_1 \oplus a_1b_0 & d_1 &= c_2 \oplus a_2b_2 \oplus a_0b_2 \oplus a_2b_0 \\
d_2 &= c_3 \oplus a_3b_3 \oplus a_0b_3 \oplus a_3b_0 & d_3 &= c_0 \oplus a_0b_0 \oplus a_0b_4 \oplus a_4b_0 \\
d_4 &= a_1b_2 \oplus a_2b_1 & d_5 &= a_1b_3 \oplus a_3b_1 \\
d_6 &= c_4 \oplus a_4b_4 \oplus a_1b_4 \oplus a_4b_1 & d_7 &= a_2b_3 \oplus a_3b_2 \\
d_8 &= a_2b_4 \oplus a_4b_2 & d_9 &= a_3b_4 \oplus a_4b_3
\end{aligned}$$

The ten shares are stored in registers for the sake of non-completeness. Next, they are compressed back to five shares $(\tilde{d_0}, \dots, \tilde{d_4})$ by $\tilde{d_i} = d_i$ for $i < 4$ and $\tilde{d_4} = d_4 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_8 \oplus d_9$. Now, consider two (glitch-extended) probes: the first on $q_1 = \tilde{d_4}$, *i.e.* $\mathcal{R}_1 = \{d_4, d_5, d_6, d_7, d_8, d_9\}$ and a second on $q_2 = b_0$, *i.e.* $\mathcal{R}_2 = \{b_0\}$. We detect that the joint probability distribution of $\mathcal{R}_1 \cup \mathcal{R}_2$ is not independent of the secret. Recall that our condition assumes the worst possible glitch function, which does not necessarily occur in the circuit. However, in this case we find that it only takes a glitch on $\tilde{d_4}$ that reveals the intermediate result $d_4 \oplus d_9$. Indeed, the joint distribution of $\{d_4 \oplus d_9, b_0\}$ depends on the unshared secret $b$. The single round function circuit is therefore not second-order secure in the presence of glitches, contrary to what was previously believed. We use a single core of a 3.2 GHz Intel Core i5-6500 CPU. It takes 2,1 seconds to find this flaw. We can correct the compression phase as follows:

$$\begin{aligned}
\tilde{d_0} &= d_0 \oplus d_4 & \tilde{d_3} &= d_3 \oplus d_8 \\
\tilde{d_1} &= d_1 \oplus d_7 & \tilde{d_4} &= d_5 \oplus d_6 \\
\tilde{d_2} &= d_2 \oplus d_9
\end{aligned}$$

It takes 12,5 seconds to enumerate all glitch-extended probe tuples and compute the exact joint probability distributions for every secret input. With the new compression, the probability distributions are identical for each secret. Note that the gadget is still vulnerable when iterated, as discussed in [RBN$^+$15].

## 5.2 Advantages and Applications

We now enumerate the advantages of glitch immunity and a number of use cases for which MaskVerif [BBFG18] is not yet compatible. We show that there is essentially no restriction on the type of circuits that can be verified using glitch immunity, which means that it should be possible to extend MaskVerif with these functionalities.

**Ease of Use vs. Performance.** A considerable advantage of glitch immunity, is that we do not need to know the "roles" of the wires, *i.e.* which share and which variable a wire belongs to. Our tool only needs to know which wires are combined in a combinational block and this is easily derived from the netlist. The advantage is that less user-provided input means less room for human error. This is in contrast with tools that try to prove non-completeness or non-interference. In those cases, the tool needs to know not only which share is carried by each wire but also which sets of wires carry shares of the same variable. However, it is this additional information which allows MaskVerif to be more efficient. There is thus a trade-off between the amount of information provided to the tool and the level of optimization possible. Similarly, the size of the glitch-extended probes can be reduced if one knows which wires carry fresh randomness.

**Multiplicative Masking.** At CHES 2018, a new hardware implementation of AES was presented, which uses multiplicative masking as opposed to Boolean masking to do the Galois field inversion [DRB18]. Since no tool for verifying the circuit with multiplicative masking was available, the authors were forced to rely on a manual verification to prove the security of the scheme (cf. the appendices of [DRB18]). We are able to use our tool to exhaustively verify the glitch-extended probing security of their first-order S-box in 20h. The second-order S-box is prohibitive in the size of its glitch-extended probes (a tuple of probes can have as many as 32 bits) and the resulting memory requirements for the histograms. Its automated formal verification thus requires that MaskVerif becomes compatible with multiplicative masking.

**Arithmetic Masking.** Another popular type of masking is arithmetic masking, which has been the topic of many works over the last years [HT16, Cor17]. As with multiplicative masking, MaskVerif currently does not allow verification of these applications. As a result, Coron released an alternative tool to formally verify arithmetically masked implementations [Cor18]. However, this tool is limited to software implementations, only verifies $d$-(S)NI properties (not $d$-probing) and suffers from false negatives. While our tool is only meant for hardware implementations, we can easily adapt it to investigate the joint probability distributions of normal probes $\mathcal{Q}_I$ instead of glitch-extended probes $\mathcal{R}_I$ and use it on the implementations of Hutter and Tunstall [HT16] to demonstrate glitch immunity's compatibility with arithmetic masking. Our tool validates their second-order secure implementation over $\mathbb{GF}(2)$ in 18 minutes.

**Randomness Recycling.** Belaid *et al.* show in [BBP$^+$16] how the right combination of NI and SNI gadgets can lower the fresh randomness cost of a circuit, compared to one that consists exclusively of SNI gadgets. In many cases however, an even lower randomness cost can be obtained by recycling random masks. Consider for example the Boolean masked Kronecker Delta function of [DRB18], shown in Figure 4, which consists of seven masked AND gates sharing a number of fresh random bits. The MaskVerif tool allows modularisation, *i.e.* allows the use of sub-gadgets in larger ones. However, for now, MaskVerif assumes that each gadget receives fresh randomness, which means that the verification of the Kronecker delta function cannot exploit the modularisation. Having a verification tool compatible with randomness recycling can be useful to find optimally customized implementations with minimum randomness costs. By treating fresh randomness as any other input, manipulated and re-used by the circuit, glitch immunity trivially allows verfication of randomness recycling. Using our tool, we are able to verify the glitch-extended probing security of the first-order implementation with randomness recycling in 0.18 seconds.

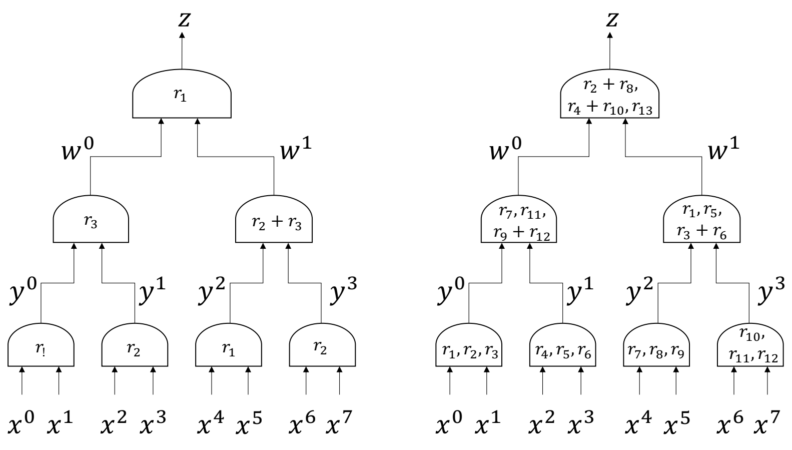

**Figure 4:** Randomness recycling in a Boolean masked Kronecker delta function [DRB18]

**Verification in $\mathbb{GF}(2)$.** Another specific though not uncommon property of the Kronecker delta function of [DRB18] is that it operates in $\mathbb{GF}(2)$. In MaskVerif, it appears the only way to work at bit-level is to provide the input in Verilog form. However, bit-level implementations might also occur in software, for instance when bit-slicing is used to parallelize computations with independent inputs. Examples are the implementations of the Kronecker delta [GPQ11] or the AES S-box [BGRV15]. In fact, from the description of MaskVerif in [BBFG18], we can only guess in which domain the inputs are interpreted and the exhaustive simulations are performed. This is related to our note in §4 about the field in which simulatability should be considered.

**Non-Uniformity.** As demonstrated in Appendix A, uniformity is not a necessary property for secure implementations. However, without verification tools in which the distribution of inputs can be specified, it is difficult to exploit this "observation". Since glitch immunity does not make any assumptions on the distribution of masks, it is perfectly suitable to verify for example low entropy masking schemes. These have only been studied for application in software. Given Lemma 1, investigating low entropy masking for hardware is an interesting direction for future work. The specification of input distributions is also related with multiplicative masking, since it is well known that a multiplicative mask is not allowed to be zero [GT02, TSG02]. This feature was therefore also useful in the use of our tool for the verification of [DRB18].

## 5.3 Efficient Flaw Detection in Practice

Provable security for masking schemes is important, which is why so many tools for this purpose exist [BBFG18, Cor18, BGI$^+$18]. However, provable schemes can still be implemented incorrectly. For this reason, implementations are usually deployed onto a hardware platform and validated using test vector leakage assessment (TVLA) [BCD$^+$13]. While this is an important step in determining whether an implementation is practically secure, the lab-based t-test (or $\chi^2$ test [MRSS18]) involves a lot of uncertainty. On the one hand, lab environments are quite noisy, which sometimes allows flaws to remain undetected. Furthermore, the results of TVLA strongly depend on the specific setup, which is different for every lab. On the other hand, it was shown at CHES 2018 [DEM18] that even secure schemes can show leakage. Flaw detection using simulation and glitch immunity can fill an important gap between proving theoretical security of a scheme and evaluating an implementation in the lab. Efficient flaw detection in software masking schemes was proposed first by Reparaz [Rep16]. We can now extend that work, not only by including flaw detection for hardware schemes, but also for software, since we can replace the moment-based t-test by a $\chi^2$ test for each separate security order, which naturally aligns with the information-theoretic definitions of both glitch immunity and probing security. Since flaw detection should be fast, we adapt our glitch-immunity tool to use random simulations as in [Rep16]. This way, it can even be used for masked circuits that cannot be formally verified due to a number of inputs that is infeasible to exhaustively simulate.

**Flaw Detection with glitch immunity.** The preprocessing described in § 5.1 remains identical. The verification step becomes a flaw detection step, by randomly choosing inputs to feed to the software implementation, instead of exhaustively simulating it for all possible inputs. We still enumerate every possible $d$-tuple of register input wires $\mathcal{Q}_I = \{q_{i_1}, \ldots, q_{i_d}\}$ and verify the condition for the corresponding sampled distribution of $\mathcal{R}_I$. Since these distributions are not exact, a zero mutual information is highly unlikely, if not impossible. The significance of a non-zero mutual information is typically measured using Pearson's $\chi^2$ test. For each pair of histograms, we therefore use such a test to verify the null hypothesis that "the two distributions are identical". Our $\chi^2$ test implementation is similar to the one described in [MRSS18]. We compute the p-value of the test, which records the probability of our observations under the null hypothesis. We reject the null hypothesis if the p-value is below a threshold (in this case $10^{-5}$). As in [BCD+13, MRSS18], we perform the hypothesis test using two plaintext classes: one fixed and one random, although fixed versus fixed is also possible.

**Distinction from TVLA.** The statistical approach on big circuits implies that it cannot be used for showing *provable* security. However, the verification is extremely informative and relevant for flaw detection and *practical* security evaluation. It is orthogonal and not to be compared to tests done in a noisy lab environment [BCD+13] since our approach achieves a lot more accuracy and genericity. Firstly, we assume an ideal noise-free world, independent of the physical aspects implying that we do not need to repeat the test as the layout, library, device etc. change. Moreover, the $\chi^2$ test is much more powerful than the moment-based t-test in its ability to detect *any* dependency of distributions on the secret. Note also the difference with the TVLA methodology either used in practical evaluations [BCD+13] or in simulation [Rep16]. There, one constructs a set of traces *once* and evaluates $d^{\text{th}}$-order security by comparing the different order moments of the samples. In contrast, for *each* security order $d$, we build different samples and compare entire probability distributions, not only statistical moments. It is as if we leak entire glitch-extended probes under an identity leakage model to an adversary in a noiseless environment and verify whether the secret can be distinguished from this. This is also different from the usage of $\chi^2$ tests in [MRSS18]. We demonstrate the condition's ability to find flaws with some examples below.

**Applications.** We repeat the experiment with higher-order threshold implementations from §5.1, but this time with the statistical version of the tool. We perform a fixed versus random test with fixed input $(a, b, c) = (0, 0, 0)$. It only takes 0,5 seconds to detect the flaw, when simulating 100 000 times with random inputs. The sample where the dependency is detected corresponds to that found by the exact tool. The statistic $\chi$ and degrees of freedom $\nu$ are respectively 30 068.8 and 2047, corresponding to a p-value of 0.0.

We further deploy the tool in the first-order secure *round function* of a Keccak implementation from [GSM17], which was shown to be insecure in the presence of glitches due to a missing register stage in [ABP+18]. We simulate not only the S-box but the entire 200-bit round function, including all linear operations, since they introduce dangerous dependencies in the inputs to the non-linear blocks. The circuit thus has a 200-bit input in two shares as well as 200 bits of randomness as input. Despite such a prohibitive number of inputs, 1 million simulations in approximately 3 minutes suffice to detect the flaw (p-value 0.0) at the input of the non-linear blocks. Finally, we again demonstrate the compatibility with arithmetic masking, by applying the tool to the third-order implementation of the first version of [HT16]. With 1 000 samples in 1 minute 20 seconds, the tool confirms the flaw noted by [Cor18] (p-value 0.0).

**Information theoretic vs computational security.** In practice, it typically suffices to have computational security. It was for example noted by Daemen [Dae16a, Dae16b] that it is unclear how to exploit the non-uniformity in his initial Keccak S-box threshold implementation [BDPVA10]. More recently, Wegener *et al.* demonstrated that no leakage can be detected in practice, even when iterating the Keccak round function for 1800 rounds [WBM18]. The condition of glitch immunity is very amenable for this purpose. The flaw detection tool essentially relaxes the glitch immunity condition by bounding the divergence between secret-condition distributions instead of requiring a zero mutual information.

$$\max |p(\mathcal{R}|x^{(0)}) - p(\mathcal{R}|x^{(1)})| \leq \epsilon \tag{16}$$

where $\epsilon$ is an arbitrarily small number (*e.g.* $2^{-40}$). Eqn. (4) can be seen as an extreme version of Eqn. (16) when setting $\epsilon = 0$. This follows the reasoning that, if we need a lot of samples (measurements) to distinguish two secret-conditioned distributions in a *noiseless* simulation, an adversary will also need *at least* as many traces during DPA in a realistic attack setting.

**Possibility for leakage functions.** A further tradeoff can be made between the accuracy and the efficiency of the verification. While we normally build probability distributions for the exact glitch extended probes, we can also replace this "identity function" by another leakage function, such as for example the Hamming weight. This reduces the histogram sizes considerably (from $2^{|\mathcal{R}_I|}$ to $2^{\log(|\mathcal{R}_I|)} = |\mathcal{R}_I|$) and thus allows for *practical* security evaluation of even larger circuits.

## 6 Conclusion

In this work, we revisited an information-theoretic approach to $d$-probing security and extended it to include glitches. The result is a concept of $d$-glitch immunity, the first necessary and *sufficient* condition for $d$-probing security in the presence of glitches.

We related glitch immunity to the security properties that form the basis of threshold implementations. While non-completeness is necessary and implied in glitch immunity, we showed that uniformity is not a necessary condition.

We redefined the software security notions of (strong) non-interference in the information-theoretic framework. We extended the definitions to include glitches and introduced the concept of $d$-glitch strong non-inteference, a sufficient condition for composability of hardware gadgets. It remains an open problem to find a *necessary* and sufficient condition for composability.

The information-theoretic approach makes extending probing security and non-interference conditions with glitches almost trivial. In the same way, another redefinition of probes can extend our current models to include other unfortunate effects such as described in the robust probing model, while easily remaining compatible with current tools.

Finally, we showed how glitch immunity can be used both for proving the security of small gadgets and for flaw detection in larger circuits. In particular, we demonstrate the flexibility of the condition and how it fills gaps in the current state-of-the-art.

**Acknowledgements**

This work was supported in part by the NIST Research Grant 60NANB15D346. Oscar Reparaz and Begül Bilgin are postdoctoral fellows of the Fund for Scientific Research - Flanders (FWO) and Lauren De Meyer is funded by a PhD fellowship of the FWO. The authors would like to thank Thomas De Cnudde, François-Xavier Standaert and Vincent Rijmen for fruitful discussions.

## References

[ABP+18] Victor Arribas, Begül Bilgin, George Petrides, Svetla Nikova, and Vincent Rijmen. Rhythmic Keccak: SCA security and low latency in HW. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):269–290, 2018.

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. An implementation of DES and AES, secure against some attacks. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.

[ANR18] Victor Arribas, Svetla Nikova, and Vincent Rijmen. VerMI: Verification Tool for Masked Implementations. In *International Conference on Electronics, Circuits, and Systems*, page 4, Bordeaux,FR, 2018. IEEE.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 116–129. ACM, 2016.

[BBFG18] Gilles Barthe, Sonia Belaïd, Pierre-Alain Fouque, and Benjamin Grégoire. MaskVerif: a formal tool for analyzing software and hardware masked implementations. Cryptology ePrint Archive, Report 2018/562, 2018.

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648. Springer, 2016.

[BCD+13] G. Becker, J. Cooper, E. De Mulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, et al. Test vector leakage assessment (TVLA) methodology in practice. In *International Cryptographic Module Conference*, volume 1001, page 13, 2013.

[BDPVA10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Building power analysis resistant implementations of Keccak. In *Second SHA-3 candidate conference*, volume 3, page 2. Citeseer, 2010.

[Bel15] Sonia Belaïd. *Security of Cryptosystems Against Power-Analysis Attacks. (Sécurité des cryptosystèmes contre les attaques par analyse de courant).* PhD thesis, École Normale Supérieure, Paris, France, 2015.

[BFGV12] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. Theory and practice of a leakage resilient masking scheme. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 758–775. Springer, 2012.

[BGG$^+$14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

[BGI$^+$18] Roderick Bloem, Hannes Groß, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal verification of masked hardware implementations in the presence of glitches. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 321–353. Springer, 2018.

[BGN$^+$14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[BGRV15] Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. DPA, bitslicing and masking at 1 GHz. In Güneysu and Handschuh [GH15], pages 599–619.

[Bil15] Begül Bilgin. *Threshold implementations : as countermeasure against higher-order differential power analysis.* PhD thesis, University of Twente, Enschede, Netherlands, 2015.

[BM16] Guido Bertoni and Marco Martinoli. A methodology for the characterisation of leakages in combinatorial logic. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 363–382. Springer, 2016.

[Cor17] Jean-Sébastien Coron. High-order conversion from Boolean to arithmetic masking. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic*

*Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 93–114. Springer, 2017.

[Cor18] Jean-Sébastien Coron. Formal verification of side-channel countermeasures via elementary circuit transformations. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, volume 10892 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 2018.

[Dae16a] Joan Daemen. On non-uniformity in threshold schemes. `https://www.cosic.esat.kuleuven.be/events/acm-ccs2016/wp-content/uploads/sites/4/2016/11/nonUniformVienna.pdf`, 2016. Talk.

[Dae16b] Joan Daemen. On non-uniformity in threshold sharings. In Begül Bilgin, Svetla Nikova, and Vincent Rijmen, editors, *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*, page 41. ACM, 2016.

[DEM18] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware masking, revisited. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(2):123–148, 2018.

[DRB18] Lauren De Meyer, Oscar Reparaz, and Begül Bilgin. Multiplicative masking for AES in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):431–468, Aug. 2018.

[FGMDP$^+$18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):89–120, Aug. 2018.

[GH15] Tim Güneysu and Helena Handschuh, editors. *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*. Springer, 2015.

[GM10] Berndt M. Gammel and Stefan Mangard. On the duality of probing and fault attacks. *J. Electronic Testing*, 26(4):483–493, 2010.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. *IACR Cryptology ePrint Archive*, 2016:486, 2016.

[GPQ11] Laurie Genelle, Emmanuel Prouff, and Michaël Quisquater. Montgomery's trick and fast implementation of masked AES. In Abderrahmane Nitaj and David Pointcheval, editors, *Progress in Cryptology - AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, volume 6737 of *Lecture Notes in Computer Science*, pages 153–169. Springer, 2011.

[GSM17] Hannes Groß, David Schaffenrath, and Stefan Mangard. Higher-order side-channel protected implementations of KECCAK. In Hana Kubátová, Martin Novotný, and Amund Skavhaug, editors, *Euromicro Conference on Digital System Design, DSD 2017, Vienna, Austria, August 30 - Sept. 1, 2017*, pages 205–212. IEEE Computer Society, 2017.

[GT02] Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In Jr. et al. [JKP03], pages 198–212.

[HT16] Michael Hutter and Michael Tunstall. Constant-time higher-order Boolean-to-arithmetic masking. *IACR Cryptology ePrint Archive*, 2016:1023, 2016.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[JKP03] Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[MRSS18] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. Leakage detection with the x2-test. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1), 2018.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

[PGA06] Emmanuel Prouff, Christophe Giraud, and Sébastien Aumônier. Provably secure s-box implementation based on fourier transform. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 216–230. Springer, 2006.

[PV17] Kostas Papagiannopoulos and Nikita Veshchikov. Mind the gap: Towards secure 1st-order masking in software. In Sylvain Guilley, editor, *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop,* *COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, volume 10348 of *Lecture Notes in Computer Science*, pages 282–297. Springer, 2017.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[Rep16] Oscar Reparaz. Detecting flawed masking schemes with leakage detection tests. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 204–222. Springer, 2016.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[SBY+18] Danilo Sijacic, Josep Balasch, Bohan Yang, Santosh Ghosh, and Ingrid Verbauwhede. Towards efficient and automated side channel evaluations at design time. In Lejla Batina, Ulrich Kühne, and Nele Mentens, editors, *PROOFS 2018, 7th International Workshop on Security Proofs for Embedded Systems, colocated with CHES 2018, Amsterdam, The Netherlands, September 13, 2018*, volume 7 of *Kalpa Publications in Computing*, pages 16–31. EasyChair, 2018.

[Sie84] Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Trans. Information Theory*, 30(5):776–780, 1984.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Güneysu and Handschuh [GH15], pages 495–513.

[SP06] Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.

[TSG02] Elena Trichina, Domenico De Seta, and Lucia Germani. Simplified adaptive multiplicative masking for AES. In Jr. et al. [JKP03], pages 187–197.

[WBM18] Felix Wegener, Christian Baiker, and Amir Moradi. Shuffle and mix: On the diffusion of randomness in threshold implementations of Keccak. Cryptology ePrint Archive, Report 2018/1092, 2018. `https://eprint.iacr.org/2018/1092`.

## A First-order security of non-uniform AND gates

We focus on the composition of two AND gates as shown in Figure 5. We will use two types of refreshing after the first gate to illustrate that uniformity is not necessary to achieve theoretical security. Note that these AND gates are designed to demonstrate our cause and therefore are neither optimal nor considered for use in another setting.

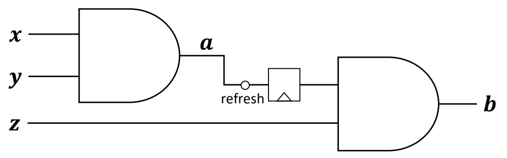

**Figure 5:** Circuit of two AND gates

We consider $x, y$ and $z$ as our sensitive variables. The first AND gate receives 3-share input values $\mathbf{x}$ and $\mathbf{y}$ and outputs 6-share value $\mathbf{a}$ as follows.

$$\begin{aligned} a_0 &= x_0y_0 & a_3 &= x_0y_1 \oplus x_1y_0 \\ a_1 &= x_1y_1 & a_4 &= x_1y_2 \oplus x_2y_1 \\ a_2 &= x_2y_2 & a_5 &= x_2y_0 \oplus x_0y_2 \end{aligned} \tag{17}$$

The second AND gate uses the 6-share output $\mathbf{a}$ with 3-share $\mathbf{z}$ as follows.

$$\begin{aligned} b_0 &= a_0z_0 \oplus a_0z_1 \oplus a_1z_0 \oplus a_4z_1 \oplus a_5z_0 \oplus a_5z_1 \\ b_1 &= a_0z_2 \oplus a_1z_1 \oplus a_1z_2 \oplus a_2z_1 \oplus a_3z_1 \oplus a_3z_2 \\ b_2 &= a_2z_0 \oplus a_2z_2 \oplus a_3z_0 \oplus a_4z_0 \oplus a_4z_2 \oplus a_5z_2 \end{aligned} \tag{18}$$

As expected, the application of the second AND gate is insecure and a single probe on either $b_0, b_1$ or $b_2$ can reveal the secret $(x, y)$. Table 2 demonstrates this fact through the probability distributions of each of the shares of $b$, which clearly each depend on the secret. The sum of the frequencies is always $2^7 = 128$ since each $b_i$ has 9 input bits (3 shares of $x, y$ and $z$) of which 2 are fixed by the secret $(x, y)$.

**Table 2:** Scaled probability distributions (frequencies) of $b_i$ for specific secrets $(y^*, x^*)$

| | $p(b_0\|x^*, y^*)$ | | $p(b_1\|x^*, y^*)$ | | $p(b_2\|x^*, y^*)$ | |
|---|---|---|---|---|---|---|
| $(y^*, x^*)$ | 0 | 1 | 0 | 1 | 0 | 1 |
| $(0, 0)$ | 100 | 28 | 100 | 28 | 112 | 16 |
| $(0, 1)$ | 84 | 44 | 92 | 36 | 88 | 40 |
| $(1, 0)$ | 84 | 44 | 92 | 36 | 88 | 40 |
| $(1, 1)$ | 84 | 44 | 92 | 36 | 72 | 56 |

This effect was previously [Bil15] contributed to the non-uniformity of the output shares $\boldsymbol{a}$, as shown in Table 3. The first AND gate has only 6 inputs bits (3 shares of $x$ and $y$) of which 2 are fixed by the secret. Hence, each probability distribution in Table 3 sums up to $2^4 = 16$.

Clearly, if there is no remasking after the first AND gate, the second AND gate leaks sensitive information. On the other hand, if we refresh every output share $a_i$, then the second AND gate receives a uniform sharing and is secure. However, a uniform sharing $\boldsymbol{a}$ is not required for $d$-probing security with or without glitches. We look at an intermediate approach where the output of the first gate is re-masked such that it is not completely uniform but the second AND gate does not leak sensitive information.

**Table 3:** Scaled probability distributions (frequencies) of $\boldsymbol{a}$ for specific secret $(y^*, x^*)$

| $(y^*, x^*)$ | $p(\boldsymbol{a}\|x^*, y^*)$ $\boldsymbol{a}$ = 000000 to 111111 |
|---|---|
| (0,0) | 7001011000000000000000000000000000000000000000000000000002202000 |
| (0,1) | 4000000001100000000101000100001000100100010000010100100000000000 |
| (1,0) | 4000000001100000000101000100001000100100010000010100100000000000 |
| (1,1) | 0110100120000000200000000020000020000000020000000000200000000000 |

**The Good.** Let's consider the following remasking using 2 bits of randomness:

$$
\begin{aligned}
a_0 &\leftarrow a_0 & a_3 &\leftarrow a_3 \oplus r_1 \oplus r_2 \\
a_1 &\leftarrow a_1 \oplus r_1 & a_4 &\leftarrow a_4 \oplus r_2 \\
a_2 &\leftarrow a_2 & a_5 &\leftarrow a_5
\end{aligned}
\tag{19}
$$

One can see in Table 4 that the sharing $\boldsymbol{a}$ is still not uniform.

**Table 4:** Scaled probability distributions (frequencies) of remasked $\boldsymbol{a}$ for specific secret $(y^*, x^*)$

| $(y^*, x^*)$ | $p(\boldsymbol{a}\|x^*, y^*)$ $\boldsymbol{a}$ = 000000 to 111111 |
|---|---|
| (0,0) | 7001011001701001017010017001011002202000200200202002002002202000 |
| (0,1) | 7001011001701001017010017001011002202000200200202002002002202000 |
| (1,0) | 7001011001701001017010017001011002202000200200202002002002202000 |
| (1,1) | 0170100170010110700101100170100120020020022020000220200020020020 |

However, this non-uniformity does not lead to leakage in the second AND gate. To verify this, we investigate the mutual information between the secret and the glitch-extended probe $\mathcal{R}_i$ of each output $b_i$, by looking at their probability distributions for different secret inputs. Note that there is synchronization after the randomization, by for example a register stage. The output shares $(b_0, b_1, b_2)$ thus depend respectively on the sets $\mathcal{R}_0 = (a_0, a_1, a_4, a_5, z_0, z_1), \mathcal{R}_1 = (a_0, a_1, a_2, a_3, z_1, z_2)$ and $\mathcal{R}_2 = (a_2, a_3, a_4, a_5, z_0, z_2)$. Since the sharing of $\mathbf{z}$ is uniform and used in a non-complete way, we will ignore those inputs for now for brevity. Table 5 shows clearly that these distributions are also not uniform. However, they are identical for each of the secret inputs $(x^*, y^*)$. The result stays the same for the secret $(x^*, y^*, z^*)$ which is omitted here for readability. It automatically follows that the probability distributions of each of the outputshares $b_0, b_1$ and $b_2$ are also independent of the secret. This means that this remasking allows a secure composition of the two AND gates, provided their separation by a register stage.

**Table 5:** Scaled probability distributions (frequencies) of $\mathcal{R}_i$ for any fixed secret $(x^*, y^*)$

| | $\mathcal{R}_i$ = 0000 to 1111 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p(\mathcal{R}_0\|x^*, y^*)$ | 8 | 2 | 8 | 2 | 8 | 2 | 8 | 2 | 4 | 2 | 4 | 2 | 4 | 2 | 4 | 2 |
| $p(\mathcal{R}_1\|x^*, y^*)$ | 9 | 3 | 9 | 3 | 3 | 1 | 3 | 1 | 9 | 3 | 9 | 3 | 3 | 1 | 3 | 1 |
| $p(\mathcal{R}_2\|x^*, y^*)$ | 8 | 2 | 8 | 2 | 8 | 2 | 8 | 2 | 4 | 2 | 4 | 2 | 4 | 2 | 4 | 2 |

**The Bad.** Suppose now that instead of the randomization in Eqn. (19), we do the following remasking.

$$\begin{aligned} a_0 &= a_0 & a_3 &= a_3 \oplus r_1 \oplus r_2 \\ a_1 &= a_1 \oplus r_1 & a_4 &= a_4 \\ a_2 &= a_2 & a_5 &= a_5 \oplus r_2 \end{aligned} \tag{20}$$

We see clearly in Table 6 that the joint probability distribution of the input set $\mathcal{R}_0$ depends on the secret inputs even though there is no correlation of the secret with output share $b_0$ itself. This indicates that the remasking is sufficient in the case of a classic probe on $b_0$, but not in the presence of glitches. The joint probability distributions are shown in Table 6.

**Table 6:** Scaled probability distributions (frequencies) of $\mathcal{R}_0 = (a_0, a_1, a_4, a_5)$ and $b_0$ for various secrets $(y^*, x^*)$

| | $p(\mathcal{R}_0\|x^*, y^*)$ | | | | | | | | | | | | | | | | $p(b_0\|x^*, y^*)$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $(y^*, x^*)$ | $\mathcal{R}_0 = 0000$ to $1111$ | | | | | | | | | | | | | | | | 0 | 1 |
| $(0,0)$ | 8 | 2 | 8 | 2 | 4 | 2 | 4 | 2 | 8 | 2 | 8 | 2 | 4 | 2 | 4 | 2 | 160 | 96 |
| $(0,1)$ | 8 | 2 | 8 | 2 | 4 | 2 | 4 | 2 | 8 | 2 | 8 | 2 | 4 | 2 | 4 | 2 | 160 | 96 |
| $(1,0)$ | 8 | 2 | 8 | 2 | 4 | 2 | 4 | 2 | 8 | 2 | 8 | 2 | 4 | 2 | 4 | 2 | 160 | 96 |
| $(1,1)$ | 6 | 4 | 6 | 4 | 6 | 0 | 6 | 0 | 6 | 4 | 6 | 4 | 6 | 0 | 6 | 0 | 160 | 96 |

**The Ugly.** With the good remasking as in Eqn. (19), we can even do certain compositions under the ISW setting with ideal gates. That is, even if the third input $\mathbf{z} = \mathbf{x}$, the output is still secure as shown in Table 7, *i.e.* $p(b_i|(x^*, y^*)) = p(b_i)$.

**Table 7:** Scaled probability distributions (frequencies) in the second AND gate with $z = x$

| | $p(b_i\|x^*, y^*)$ | | $p(\boldsymbol{b}\|x^*, y^*)$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $(y^*, x^*)$ | 0 | 1 | 000 | 011 | 101 | 110 | 001 | 010 | 100 | 111 |
| (0,0) | 40 | 24 | 28 | 0 | 0 | 12 | 0 | 12 | 12 | 0 |
| (0,1) | 40 | 24 | 28 | 0 | 0 | 12 | 0 | 12 | 12 | 0 |
| (1,0) | 40 | 24 | 28 | 0 | 0 | 12 | 0 | 12 | 12 | 0 |
| (1,1) | 40 | 24 | 0 | 20 | 20 | 0 | 20 | 0 | 0 | 4 |

However, in a glitchy environment, we do not necessarily get composability. Neither $\mathcal{R}_0 = (a_0, a_1, a_4, a_5, x_0, x_1)$ nor $\mathcal{R}_1 = (a_0, a_1, a_2, a_3, x_1, x_2)$ are independent from the secret.

**Conclusion.** In order to avoid leakage of sensitive data, what we need is independence of the secret from the sensitive data. In this case study, we have demonstrated that this does not necessarily require uniform distributions. Using $d$-glitch immunity as a verifying mechanism, we can create custom gates with minimal randomness consumption.