Can We Tolerate Small Side-Channel Leakages: The Role of Registers in Glitch-Stopping Circuits

Artemii Ovchinnikov[1], Jelle Biesmans[1], Kris Myny[1], Ventzislav Nikov[2], and Svetla Nikova[1]

[1] COSIC, KU Leuven, Belgium
firstname.lastname@esat.kuleuven.be
[2] venci.nikov@gmail.com

Abstract. Research on cryptographic algorithms implemented in hardware and protected against side-channel attacks has advanced rapidly in recent years. Generalized masking schemes, such as Threshold Implementations (TI) and Domain-Oriented Masking (DOM), currently provide a solid theoretical security foundation. Security models, including the probing model and its various extensions, enable formal verification of these guarantees. In addition, established guidelines for designing securely composable gadgets, along with tools for the automatic generation of masked designs, have further advanced the field. Experimental security assessment approaches, such as the Test Vector Leakage Assessment (TVLA) complement these efforts. Consequently, the primary focus of the research community has shifted toward optimizing existing techniques and bridging the gap between theoretical and practical security models.

In this work, we demonstrate a case in which side-channel leakage, caused by glitches, can be concealed during experimental assessment in a setup that is theoretically not robustly secure. This effect arises due to specific patterns of glitch propagation. We investigate whether a particular layout of the complete logic chain can further contribute to a design's resistance to side-channel attacks, potentially reducing latency and area by relaxing glitch-mitigation requirements, such as reducing the number of pipeline registers. To this end, we introduce new adversarial model which further relaxes the model of Müller and Moradi, introduced at CHES 2024. To illustrate the practical relevance of our proposal, we provide experimental evidence by modifying a well-known, provably secure AES S-box design by De Cnudde, rendering it insecure under the robust probing model. We conduct TVLA of power consumption for both FPGA-based (physical) and ASIC-like (simulation) implementations of our netlists, demonstrating the absence of detectable leakage, similar to the originally robustly secure version of the algorithm.

Keywords: Masking · Side-Channel Analysis · Probing Models · Glitches · FPGA · ASIC · Spectre · AES

1 Introduction

For over a quarter of a century, both research and industrial communities working on cryptographic hardware implementations have been dealing with a major threat introduced by Kocher *et al.* [22], known as differential power analysis (DPA). Such side-channel attacks (SCA) allow adversaries to retrieve secret data processed by an embedded system. They exploit the relationship between data-dependent switching activity in logic gates and the device's power consumption, allowing statistical attacks [5] without breaking theoretically secure algorithms. Other examples of SCA include electromagnetic emission [15,33] and timing attacks [23].

Numerous approaches have been proposed to eliminate the above-mentioned dependency, which are commonly categorized into *hiding* and *masking*. *Hiding* aims to obscure the leakage by inserting dummy operations, adding random delays, and employing other obfuscation techniques [26]. These methods increase noise: signal components that are unrelated to the processed secret data but contribute to the overall power consumption of the device, thereby reducing its correlation with the secret.

Masking, in contrast, aims to make the processed data appear uniformly distributed. This is one of the most extensively studied solutions based on secret sharing schemes, which allow to correctly process the data that is statistically independent of the unmasked values. The latter is achieved by introducing sufficient randomness and ensuring that the leakages of the shared functions are independent [8,16]. One of the most widely adopted masking techniques is Boolean masking, in which the shares can be recombined to recover the original value using the bitwise XOR operation. This scheme has gained popularity in cryptographic hardware implementations due to its simplicity when applied to affine and linear functions, which constitute a large part of symmetric-key algorithms for a proper diffusion. Specifically, each such operation can be instantiated multiple times, equal to the number of shares, and applied to each share individually. However, symmetric-key cryptography also relies on non-linear functions for substitution layers (S-boxes), which poses challenges, as Boolean masking cannot be straightforwardly applied to them.

To address this issue, different approaches were proposed, either aiming to mask non-linear AND operations [35] or to design frameworks that transform SCA-insecure circuits into masked ones [19]. However, despite their value to the research community, early works either did not consider or underestimated the impact of unintentional gate switching activity, known as *glitches*, which can leak more information through side channels than assumed by existing frameworks [27]. Later techniques, such as Threshold Implementations (TI) [31] and Domain-Oriented Masking (DOM) [17], account for these physical effects, leading to more robust designs.

Despite the algorithmic complexity of implementing masking countermeasures, they have attracted significant attention from the research community, as their security can be theoretically proven. To provide general guidelines for the design of SCA-secure cryptographic circuits, Ishai *et al.* [19] introduced one

of the first theoretical models of an adversary with physical access to a device, commonly referred to as the d-probing adversary. In this model, d denotes the number of wires observed by the adversary within a given time period, typically a single clock cycle in hardware implementations. If the information observed by probes placed arbitrarily within the masked design remains independent of the secret, the circuit is considered SCA-secure. To account for the effects of glitches, the d-probing model was later extended to the *robust probing model* [14], which is widely considered the strongest adversary model to date. Crucially, a design that is secure in this model is also considered secure in the noisy leakage [32] and the bounded moment leakage model, as shown by Barthe et al. [2]. The latter model restricts the adversary to analyzing up to the d-th statistical moment of the leakage distribution, thereby establishing a connection with practical leakage assessment techniques, such as TVLA [3], which are widely used to evaluate the DPA resistance of masked implementations.

Those findings contributed to the development of robust, secure designs and composable security notions [1,7], the latter allowing the automatic generation of masking [20,21]. Despite the availability of strong theoretical security guarantees, these models and design recommendations can be overly conservative. Auto-generated, composable designs tend to incur high area and randomness requirements, often underperforming hand-crafted designs. As a result, recent efforts have focused on optimizing the proposed countermeasures, studying different techniques to overcome the effect of glitches [36,37] or relaxing existing models. The latter includes the introduction of *Robust but Relaxed probing (RR-probing) model* [30], which represents a step toward bridging the gap between strong theoretical security guarantees and practical attacker models. This model implies a more realistic probe information collected from the glitch propagation mechanism, as it accounts for the functions implemented by the gates encountered along the probe's propagation and is data dependent. Using this model, it has become possible to demonstrate the security of certain masked glitch-immune circuits, such as LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL) [25]. These circuits do not propagate glitches by design, making the application of the robust probing model to them over-conservative, unrealistic, and incorrect. Even with this relaxation in place, there remains a gap between theoretical security guarantees and the practical observation of leakage.

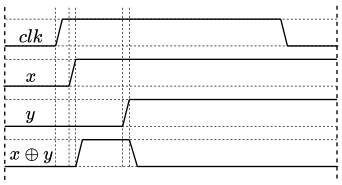


Fig. 1. Illustration of a glitch in a combinational gate. The output briefly spikes due to asynchronous input changes.

Contributions Our goal is to investigate the discrepancy between the magnitude of leakage caused by glitches in silicon and that predicted by theoretical probing models. We revisit several axioms for secure glitch-stopping circuits introduced

over 20 years ago with TI, such as the use of registers to prevent leakage. We study a range of examples in the robust probing model [14], the relaxed probing model [30], and a variant of the RR-probing model, which we term the ***highly relaxed probing (HR-probing) model***. We further examine the role of registers and glitch-induced leakage in the presence of noise (FPGA measurements) as well as in noiseless settings (Spectre power simulations). Our work builds on recent studies that aim to simultaneously mitigate specific glitches and optimize existing implementations [24], while also taking into account known security flaws in these designs [29]. In our study, we investigate a modified AES design, originally introduced by De Cnudde *et al.* [12]. Our implementation omits a register stage between the final two stages of the AES S-box from the original work. This is considered insecure in the robust and RR-probing models but practical FPGA experiments and Spectre simulations do not exhibit first-order leakage. We argue that leakage detection is avoided due to the circuit's complexity and the fact that the glitch behavior can be shaped by careful engineering of the combined stage. We reason that in the HR-probing model the resulting glitch behavior of the investigated circuit does not consistently reveal sufficient information about the masked secrets.

2 Preliminaries

We begin this section with an introduction to the notation used throughout the paper. We then introduce Threshold Implementations (TI) [31], a variant of masking that is provably secure in the presence of glitches. We finalize the section with a description of existing adversary models.

2.1 Notation

We denote stochastic variables by capital letters in italic, e.g., X, and deterministic Boolean variables by lowercase letters, e.g., x. Field elements are represented in bold $\mathbf{x} = (x^{n-1}, \ldots, x^1, x^0)$, where $\mathbf{x} \in \mathbb{GF}(2^n)$, with coordinates indexed by lowercase superscripts. For operations in $\mathbb{GF}(2^n)$, we use $\oplus$ for XOR, and $\odot$ for multiplication. Bitwise operations follow the same convention, with the multiplication symbol $\odot$ omitted when it is clear from the context. To model masking, we use vector spaces and denote their components as subscripts for share indices, e.g., $\overline{x} = (x_0, x_1, \ldots, x_{s-1}) \in \mathbb{F}_2^s$, with the total number of shares denoted by s. The bit values from two consecutive clock cycles are written as $x[clk], x[clk-1]$, and are subsequently referred to as x, x', respectively, for brevity. In addition to the mathematical notation, we adopt the following definitions describing the behavior of digital signals.

Definition 1 (Switch and Glitch of a Signal).

- ***Switch*** — *a transition of a signal between logical levels;*
- ***Glitch*** — *a sequence of two consecutive switches.*

That said, the glitch illustrated in Figure 1 for the $x \oplus y$ signal consists of two switches ($0 \to 1$ and $1 \to 0$).

2.2 Threshold Implementations

To design countermeasures against side-channel attacks, we employ Boolean masking [8,16]. In this technique, a secret bit $x \in \mathbb{F}_2$ is split into s shares $\overline{x} = (x_0, x_1, \ldots, x_{s-1}) \in \mathbb{F}_2^s$, such that $\bigoplus_{i=0}^{s-1} x_i = x$. Each share should be statistically independent of the original secret. To achieve this, the first $s-1$ shares are drawn uniformly at random $r_0, \ldots, r_{s-2} \xleftarrow{\$} \mathbb{F}_2$ and the last share is computed as $x_{s-1} = x \oplus r_0 \oplus r_1 \oplus \cdots \oplus r_{s-2}$. Such a sharing is called *uniform* if all possible maskings of the secret are equally likely.

Masking a single variable is theoretically simple, but issues arise when those masked values must be processed by functions while preserving correctness and uniformity of the result. To address this, Nikova *et al.* [31] proposed Threshold Implementations (TI), a technique that enables secure masking of functions by satisfying three properties, namely *correctness, non-completeness, uniformity.*

Definition 2 (Threshold Implementations [31]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a function, and let $\overline{F} : \mathbb{F}_2^{ns_x} \to \mathbb{F}_2^{ms_y}$ be a masking of F. The masking $\overline{F}$ is said to be:*

1. correct *if, for all $x_0, \ldots, x_{s_x-1} \in \mathbb{F}_2^n$,*
$$\bigoplus_{i=0}^{s_y-1} F_i(x_0, \ldots, x_{s_x-1}) = F\left(\bigoplus_{i=0}^{s_x-1} x_i\right);$$
2. non-complete *if any share function F_i depends on at most s_x-1 input shares;*
3. uniform *if $\overline{F}$ maps a uniform random masking of any $x \in \mathbb{F}_2^n$ to a uniform random masking of $F(x) \in \mathbb{F}_2^m$.*

Later, the intuition that TIs are secure in the presence of glitches has been formally demonstrated by Dhooghe *et al.* [13]. To make the underlying notion of security precise, we introduce the adversary models on which it is based.

2.3 Probing Models

When discussing probing models, we typically represent a masked algorithm under evaluation as a circuit, or equivalently as a directed acyclic graph. Wires carrying digital values (bits) correspond to edges, and binary operations (logic gates) correspond to vertices. Such a circuit can be augmented with special *random gates* with fan-in 0, which produce random bits that may be used for refreshing. To store specific execution states of the algorithm for synchronization or glitch-stopping purposes, we introduce *memory gates*, which output the previously stored value whenever the circuit is active and store the value present at their input. In practice, random gates can be realized using pseudo-random number generators (PRNG), whereas memory gates can be implemented as registers (based on flip-flops and synchronized by a clock signal in ASICs and FPGAs).

We illustrate this abstraction using an example circuit, namely one half of a DOM AND gate [17] with $d+1 = 2$ shares (Figure 2), in order to further highlight

the capabilities of the considered adversary models. In this example, x_0 denotes a share of variable x, y_0, y_1 denote shares of y, and r is a fresh random bit. The share x_1 is absent; consequently, only a joint observation of both y_0 and y_1 can reveal information about the secret. We analyze this example under different probing models.

ISW Probing Model In the foundational model by Ishai *et al.* [19], the adversary is granted the capability of observing (probing) up to d intermediate wires in a circuit, designed according to the principles described above, during its execution. The probes may be reallocated every clock cycle, but their placement remains fixed within a cycle. If the joint distribution of any d observed signals is independent of any function of the secret input of the entire algorithm, which includes plaintext and key information in cryptographic context, then the circuit is considered secure. We note that, despite providing a foundation for theoretical description of a side-channel adversary, circuits that are secure in ISW probing model are not necessarily secure in more realistic scenarios where dynamic hazards, such as glitches and transitions, contribute to the overall information leakage [27].

To illustrate this, we refer to Figure 2(a), which models a first-order adversary with $d = 1$. The probe may be placed on any wire of the combinational circuit between two registers. However, placing the probe on the outputs of the leftmost register reveals no more information than a single bit. Similarly, placing a probe after AND gates or after the lower XOR gate does not reveal more information than a single share of the secret. Consequently, the most advantageous location to place a probe for the adversary is at the input of the second, rightmost register. According to the definition of ISW probing adversary, this probe reveals information of the stable signal carried by this wire. Although the wire carries information about the secret $y = y_0 \oplus y_1$, the presence of the fresh random bit r renders the resulting distribution uniform. As a result, probing any single wire of the circuit depicted in Figure 2(a) does not allow the adversary to distinguish the secret.

Robust Probing Model In practice, signal propagation delays between logic and memory gates may differ, which can lead to short time intervals during which a probe observes a signal combining multiple shares before it is re-masked. This may provide the adversary with an opportunity to mount statistical attacks [22] or, in some cases, to recover the secret value from a single measurement [26]. To eliminate this issue, an extension of ISW probing model was introduced by Faust *et al.* [14], known as the *robust probing model*. This model is the most comprehensive probing model proposed to date. It accounts for three types of physical effects that may arise in ASIC and FPGA implementations, namely:

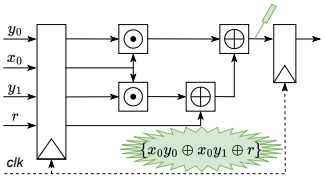
- Glitches - unintended signal transitions in combinational circuits [27,31];
- Transitions - state transition leakage in memory cells (registers) [9];
- Couplings - crosstalk effects between adjacent wires caused by parasitic capacitance [11].

In the robust probing model, a single ISW-type probe is extended according to the rules of a chosen set of physical effects. We illustrate the impact of these effects using the same circuit employed to depict the ISW probing case. In corresponding figures, the solid probe symbol indicates the originally placed probe, whereas dashed probes represent the additional probe positions induced by the considered physical effects.

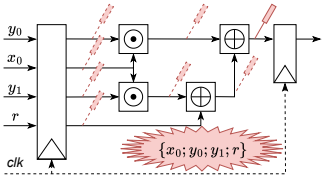
Figure 2(b) illustrates the case of a *glitch-extended* probe. Since the probe, placed at the input of the leftmost register, remains fixed there for an entire clock cycle and signals require time to stabilize, it observes all intermediate signal variations occurring before the cycle's idle phase. These transient glitches can leak information about the inputs of the preceding gate. For example, in the case of an XOR gate, observing a glitch on its output implies that either at least one input had a glitch, or both inputs switched with a delay. Since glitches can arise at any combinational gate in a circuit, their effects were first taken into account in Consolidating Masking Schemes (CMS) [34], where a probe is assumed to be extendable backwards either to the primary inputs of the circuit or to the outputs of the preceding register stage. This intuition was later formalized by Faust *et al.* [14] in the robust glitch-extended probing model. Registers, being synchronized with the clock, stabilize the state and do not propagate glitches. Thus, a probe considered safe under the standard ISW probing adversary model effectively extends throughout the entire combinational logic, granting the adversary access to all intermediate wires. Consequently, this enables statistical distinguishability of secret y, because the adversary observes both of its shares y_0, y_1. In the glitch-extended probing model, the optimal probe placement is therefore at the input of the next register stage, as it covers all inputs influencing the probed wire's signal.

Figure 2(c) illustrates the effect of a *transition-extended* probe. This type of leakage relates to the transitions between consecutive values stored in a register. Specifically, when a register switches its state at the beginning of a clock cycle, an adversary can briefly observe both the previous and current register outputs. In theory, a set of d probes can be expanded to $2d$ probes, placed on the register outputs and observing two consecutive clock cycles. As shown in Figure 2(c), such an extension is generally safe, unless there exists a correlation between the input and the output of the register through the following combinational logic. For this reason, transitional leakage is never assessed as a standalone extension; it is always combined with the glitch-extended probe. This combination extends the probe placed in Figure 2(b) to the values: $\{y_0[clk]; y_0[clk-1]; y_1[clk]; y_1[clk-1]; x_0[clk]; x_0[clk-1]; r[clk]; r[clk-1]\}$, yielding theoretically the most powerful *glitch-transition-extended* probing adversary.

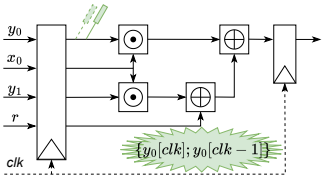
We also consider the *coupling-extended* probe, although this effect only arises during the placement and routing steps of hardware design and does not contribute to the theoretical security guarantees defined by the robust probing model. As mentioned above, it results from crosstalk between adjacent wires, which occurs only if the wires are physically neighboring in the circuit layout. Figure 2(d) illustrates an example in which the outputs of AND gates are neigh-



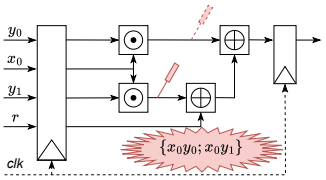
(a) ISW probing model



(b) Glitch-extended probing model



(c) Transition-extended probing model



(d) Coupling-extended probing model

Fig. 2. Probing rules for ISW [19] and the robust probing [14] models

boring wires, potentially leaking some information about the secret. However, coupling effects are generally much weaker in statistical tests, like TVLA, than glitches and transitions [11] and cannot be accounted for at the RTL design stage.

Overall, the robust glitch-transition-extended probing model grants the probing adversary maximal capabilities among the known models. A design that is secure under this model is expected to exhibit no leakage in practical assessments. However, the model can be over-conservative, potentially classifying some designs as insecure even if they are safe in practice.

Robust but Relaxed Probing Model To account for more realistic glitch behavior, Müller and Moradi introduced an extension of the robust probing model, referred to as the *Robust but Relaxed Probing Model* (RR-probing) [30]. The main intuition behind this work is that glitch patterns do not occur uniformly and differ across combinational gates; therefore, a probe does not always need to be extended to the inputs of the preceding gate. For example, consider an AND gate with at least one stable input equal to 0. In this case, the adversary always observes 0 at the output, regardless of the value of the other input. Consequently, the probe reveals no information about either input of the gate, since the adversary cannot determine which input is stable, which one is switching, or whether both inputs are stable. As a result, glitches originating from preceding combinational gates do not propagate further. A similar intuition applies to XOR gates, which do not reveal additional information about their inputs if both inputs are stable. In their work, Müller and Moradi introduced two new notions: *probe-extension* and *glitch-propagation* functions, defined for each basic logic gate with

fan-in 2 and fan-out 1. The probe-extension function F_Z^{gate} specifies whether a probe can be extended backwards through the gate. If $F_Z^{\text{gate}} = 1$, the adversary may extend its probe according to the robust probing rules; otherwise, the probe backwards extension stops there. The glitch-propagation function G_Z^{gate} specifies whether a glitch actually occurs at the gate output. It always equals zero for the wires representing register outputs or initial inputs of the circuit.

Evidently, the RR-probing model is data-dependent. Although it bridges the gap between the robust probing model and practical side-channel leakage assessments, its applicability is currently limited to certain instances of LMDPL logic. In particular, it is possible to show, via exhaustive search, that a probe placed at the output of an LMDPL gate does not observe all of its inputs, thereby reducing the adversary's power and enabling a formal security proof [30]. For masking techniques without precharge, such as threshold implementations, domain-oriented masking, or the examples shown in Figure 2, there exist cases in which an RR-probe expands into a robust glitch-transition probe. However, this is not always the case and depends strongly on the input data and the circuit type.

3 Highly Relaxed Probing Model

In this paper, we demonstrate that the RR-probing model can be further adapted to more practical scenarios. To this end, we revisit several information-leakage assumptions introduced in [30] and propose an additional relaxation, which we will call the *Highly Relaxed Probing Model* (HR-probing). We emphasize that, unlike its predecessors, this model is not robust, as it relies on assumptions about the physical layout of the evaluated circuit.

We begin by modeling the adversary's capabilities. To formalize the HR adversary model, we first provide a visualization of the considered probing scenarios. This differs from the RR-probing model, where an adversary may observe either two consecutive outputs of a gate or two consecutive values on each input wire of a basic logic gate (XOR, XNOR, OR, NOR, AND, NAND), that is, $\{z', z\}$ or $\{x', x, y', y\}$, respectively. The observable extensions, in RR probing, depend on the gate type and the preceding signal switches. Without loss of generality, it suffices to consider one balanced gate (XOR) and one unbalanced gate (AND), as these cases cover all relevant switch and glitch behaviors up to symmetry. Tables 1a and 6a of the original RR-probing work [30] enumerate possible patterns of output switches based on two consecutive values of the inputs X and Y for the AND and XOR gates, respectively. As noted by the authors, the tables are simplified and do not account for propagation delays. Subsequent work by Moradi [29] partially addresses this limitation by analyzing glitch patterns of differently masked AND gates (ISW, E-ISW, and DOM). In particular, Moradi demonstrated that it is possible to distinguish based only on the output switches, the input (secret) values of y when analyzing a single domain that forms one half of the ISW and E-ISW gadgets. He argued that for those two gadgets ensuring controlled delays and prioritizing randomness propagation does not provide se-

curity against the RR-probing adversary, whereas the DOM-gadget guarantees security by making y indistinguishable regardless of the observed glitch patterns by and RR adversary. However, the examined glitch patterns in [29] remain incomplete since they do not explicitly capture the cases in which glitchy inputs are provided, even though such effects are supposedly covered by the formal RR definitions.

Our next aim is to extend these examples to incorporate switch propagation. In addition, we argue that observing switching activity does not necessarily imply that a probe is extendable in practice. In particular, in the HR-probing model the adversary can observe all switching events and their timings at the probe location and, only in certain cases (in contrast to RR-probing), can extend a probe backward along the signal path. It generally occurs when the adversary can control (or has knowledge of) one of the inputs while attempting to infer the other. However, such expansion scenarios are considerably less likely than what is assumed in the RR-probing model. Consequently, the HR adversary is weaker than RR and non-robust, yet also more realistic. A single HR-probe observes the complete set of switches occurring on a wire *within a single clock cycle*. However, HR attacker is not necessarily able to align these observed switches with their originating signals and can therefore draw conclusions about the input data only in a probabilistic manner. Importantly, such conclusions depend entirely on the physical layout of the probed circuit, which may or may not be known to the attacker. As a result, one may either prove security against this adversary under specific layout assumptions or quantify the resulting information leakage.

3.1 XOR gate probing

We begin investigating switch patterns with the XOR gate, as featured in Table 6a of the original RR-probing work [30]. Consider a circuit similar to that in Figure 2, but consisting solely of XOR gates, as shown in Figure 3. As previously observed, placing a probe at any position other than wire ① does not reveal sufficient information to determine the secret variable y. When the probe is placed at position ①, an ISW probe observes the stable value $y_0 \oplus y_1 \oplus r$, which is secure due to masking by a fresh random bit r. A glitch-extended probe, in contrast, observes all circuit inputs and therefore leaks information about y.

We now examine what is observed by an RR probe. According to the RR-probing definitions, any switch of XOR gate output leaks information about both of its inputs. Consider the case in which x_0 switches while all other signals remain stable. Since x_0 contributes to all XOR gates in the circuit, wires ②, ③, and ④ each experience one transition, while wire ① may glitch due to unequal propagation delays. Under the RR-probing abstraction, the probe therefore reconstructs the secret y by observing

$$\mathcal{P}_{RR}^{①} = \{x_0, y_0, y_1, r\} \tag{1}$$

However, this abstraction does not accurately reflect practical behavior, which we will illustrate below.

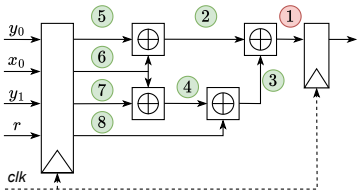


Fig. 3. XOR chain, operating on a secret y, share x_0, and a fresh random bit r.

We model the leakage behavior precisely for both $y = 0$, where $(y_0, y_1) \in \{(0,0),\ (1,1)\}$, and $y = 1$, where $(y_0, y_1) \in \{(0,1),\ (1,0)\}$. As illustrated in Figure 4, the glitch patterns observed on wire ① are identical for the stable cases $y = 0$ and $y = 1$. The symmetric case $x_0 : 1 \to 0$ is omitted for brevity. Consequently, the example used to justify the robustness of the RR probe does not hold in practice for this circuit, as the secret variable y remains indistinguishable; in particular, it cannot be determined whether it is equal to 0 or 1. We emphasize that this does not invalidate the RR-probing model, which intentionally captures worst-case, data-dependent scenarios. Rather, it highlights that, with a single probe and a single trace, recovering y may be impossible, even under relaxed assumptions, leaving room for further refinement of the model.

We continue with the circuit of Figure 3 to identify which practical switch scenarios enable our adversary to extract information about y, and, in particular, whether such information reveals the actual value of y or merely allows distinguishing between observations corresponding to different values of y. The latter distinction is important, because a violation of indistinguishability leads to potentially detectable leakage in practical statistical tests such as TVLA—where different sets of side-channel measurements are compared. We count the number of switches induced on wire ① by each input transition and argue that both the number and the timing of these switches determine the leakage. It leads to the following general conclusion:

Lemma 1 (XOR gate switching). *Let the two inputs of an XOR gate exhibit k_1 and k_2 switches, respectively, with $k_1 \geq k_2$. Then the output exhibits k_1+k_2-2i switches for some $i \in \{0, \ldots, k_2\}$. Equivalently, the number of output switches lies between $k_1 - k_2$ and $k_1 + k_2$.*

Proof. Each switch on an input of an XOR gate induces a potential output transition. If switches on the two inputs occur at distinct times, their effects propagate independently and contribute additively to the number of output switches. However, when a switch on one input coincides in time with a switches on the other input, the two transitions cancel at the output of the XOR gate and produce no output switches.

At most k_2 such cancellations can occur, since the second input has only k_2 switches. Each cancellation removes two potential output switches. Therefore, starting from the maximal case of $k_1 + k_2$ output switches, the number of output switches can be reduced by $2i$ for any $i \in \{0, \ldots, k_2\}$, yielding the stated range. □

The Lemma 1 allows a straightforward composition throughout a circuit consisting of XOR gates. We now recall our initial example in Figure 3. To

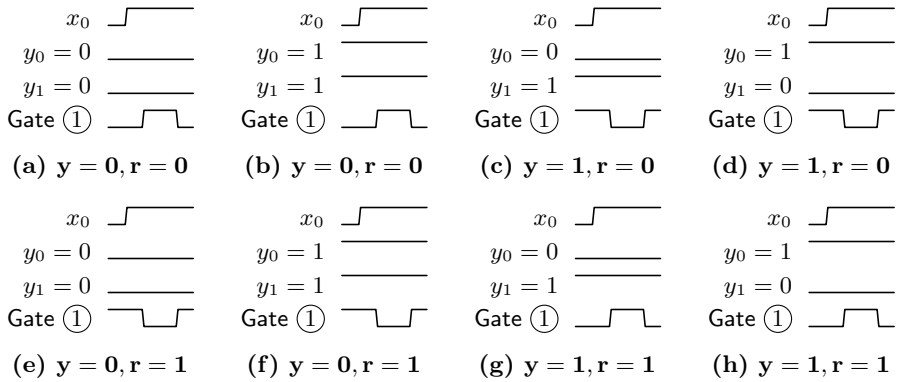


Fig. 4. Signal switches and glitch behavior in the XOR gate chain, illustrating indistinguishability between $y = 0$ and $y = 1$.

capture the worst-case scenario, we assume that each switch is associated with a distinct propagation time. Then, each input may induce the following number of switches:

- r causes 0 or 1 switches further noted as $\{0, 1\}$ switches;
- x_0 causes $\{0, 2\}$ switches, as it affects two XOR gates;
- y_0 causes $\{0, 1\}$ switches;
- y_1 causes $\{0, 1\}$ switches;
- $\{y_0; y_1\}$ jointly cause $\{0, 1, 2\}$ switches, with an important distinction:
  - if y remains constant across two consecutive clock cycles, it causes $\{0, 2\}$ switches;
  - if y transitions as $0 \to 1$ or $1 \to 0$, it always causes exactly $\{1\}$ switch.

Combining the cases above, the signal on wire ① may exhibit between 0 and 5 switches. Therefore, we analyze the distinguishability of the four transitions of the secret bit y, namely $y : \{0 \to 0,\ 0 \to 1,\ 1 \to 0,\ 1 \to 1\}$. As noted above, these transitions fall into two groups: the set $\{0, 1, 2, 3, 4, 5\}$ of possible switch counts for $y : \{0 \to 0,\ 1 \to 1\}$, and the set $\{1, 2, 3, 4\}$ for $y : \{0 \to 1,\ 1 \to 0\}$. However, the probabilities of individual switch counts differ: observing 0 or 5 switches requires, respectively, that all inputs remain unchanged or that all of them toggle, whereas, for example, observing 2 switches can arise from several distinct input patterns. To cover all possibilities in the circuit of Figure 3, we consider the $2^4 = 16$ input-transition cases per input wire and, in principle, all $5! = 120$ permutations of arrival orders for the resulting switches (accounting for the fact that the wire carrying x_0 may incur different delays after the split). In practice, the synthesized layout restricts the number of realizable propagation delay orderings: while many permutations are theoretically possible, only a limited subset can occur on an actual device (after FPGA placement or ASIC tape-out), depending on the routing choices, the physical distances between gates and the cell types. Moreover, for an XOR-only circuit, permutations of arrival times do

not affect the count of switches. As we consider the worst-case scenario, where XOR gates propagate every incoming switch, enumerating all 5! orders changes only the timing but leaves the distribution over switch counts unchanged. To quantify distinguishability between the groups of y transitions, we compute the *total variation distance* (TVD) between the induced distributions of observed switch patterns. This directly determines the adversary's distinguishing advantage under optimal single-trace hypothesis testing.

$$\mathrm{Adv}_{HR} = \mathrm{TVD}(P, Q) = \frac{1}{2} \sum_{k} |p(k) - q(k)| \tag{2}$$

where P and Q denote the two distributions of switches k being distinguished, and $p(k)$ and $q(k)$ are their corresponding probability mass functions. Importantly, this metric does not reveal the actual value of y; instead, it only indicates how y changed between two consecutive inputs. For the circuit in Figure 3, we define the distributions P corresponding to $y = \{0 \to 0,\ 1 \to 1\}$, and Q corresponding to $y = \{0 \to 1,\ 1 \to 0\}$.

$$P(k) = \mathbf{Pr}[K = k | y \in \{0 \to 0,\ 1 \to 1\}]$$
$$Q(k) = \mathbf{Pr}[K = k | y \in \{0 \to 1,\ 1 \to 0\}]$$
$$\text{for } k \in \{0, 1, 2, 3, 4, 5\}$$

k	0	1	2	3	4	5
$P(k)$	1/8	1/8	2/8	2/8	1/8	1/8
$Q(k)$	0	2/8	2/8	2/8	2/8	0

We then obtain the following calculations:

$$\mathrm{TVD}(P, Q) = \frac{1}{2} \sum_{k=0}^{5} |P(k) - Q(k)| = 0.25 = \mathrm{Adv}_{HR}$$

The adversary's distinguishing advantage is normalized between 0 and 1, where 0 corresponds to performance no better than random guessing, and 1 corresponds to always identifying the correct hypothesis. In terms of advantage over the random guesser, the adversary has 12.5% better chances of distinguishing between the two groups in our case. Designs that are secure under the robust probing model yield zero advantage to our HR-probing adversary.

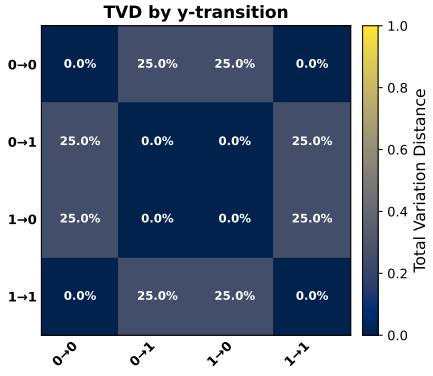


Fig. 5. Total variation distances between all pairs of y transitions (XOR chain).

To conclude the analysis, we provide a pair-wise comparison of all sets of y switching, depicted in Figure 5. It further corroborates the adversary's distinguishing advantage of 0.25 and, moreover, confirms the indistinguishability within the pairs $y : 0 \to 0$ versus $y : 1 \to 1$ and $y : 0 \to 1$ versus $y : 1 \to 0$.

As stated at the beginning of this section, the HR-probing adversary observes all switches and their timings at the probe location. In practice, however, hardware delays are not perfectly stable. Even if delays follow a nominal pattern due to careful circuit design, their timings cannot be assumed to be precise and constant across executions. Consequently, the adversary cannot align the observed switches with the specific inputs that generated them, unless the circuit is deliberately engineered to enforce particular delay constraints, causing specific switches to occur at similar times. Moreover, some switches may cancel when multiple input transitions arrive at an XOR gate simultaneously.

The adversary therefore observes the following sequence at the probe location:

$$\mathcal{P}_{HR}^{\textcircled{1}} = \{x'_0 \oplus y'_0 \oplus x'_0 \oplus y'_1 \oplus r', x_0 \oplus y_0 \oplus x_0 \oplus y_1 \oplus r; \mathcal{SW}\} = \{z', z; \mathcal{SW}\} \quad (3)$$

where $\mathcal{SW}$ denotes the sequence of switches.

3.2 AND gate probing

The case of the AND gate is more intricate, as its output switching behavior depends on joint switching patterns of the inputs rather than on their individual transitions alone. Specifically, an output switch occurs if and only if one input is stable at 1 while a switch occurs on the other input. Consequently, even if multiple glitches arrive at the inputs, they may follow one another in such a way that no switches propagate to the output.

Lemma 2 (AND gate switching). *If the two inputs of an AND gate carry k_1 and k_2 switches then the output has between 0 and $(k_1 + k_2)$ switches.*

Proof. Each switch on an input of an AND gate can produce at most one output switch, since the output is a Boolean signal. Therefore the output can exhibit no more than $k_1 + k_2$ switches in total. That said, to prove our claim, it is sufficient to demonstrate the test-cases for both extremal cases. We provide a visual example to facilitate the claim of the Lemma in Figure 6. It demonstrates cases with 0 and with $k_1 + k_2$ number of switches, confirming our initial claim. □

Corollary 1. *If the propagation delays of all input switches are distinct, then the total number of output switches of an AND gate $a \odot b$ with k_1 switches on a and k_2 switches on b can be expressed in terms of the event sequence $\mathcal{SW} = \langle e_1, \ldots, e_K \rangle$, where $e_j \in \{A, B\}$ indicates whether the j-th event is a switch on a or b, and $K = k_1 + k_2$. $\mathbf{1}[\cdot]$ is the indicator function. Let*

$$\alpha_j = \big(\#\{\, i < j : e_i = A \,\}\big) \bmod 2, \qquad \beta_j = \big(\#\{\, i < j : e_i = B \,\}\big) \bmod 2.$$

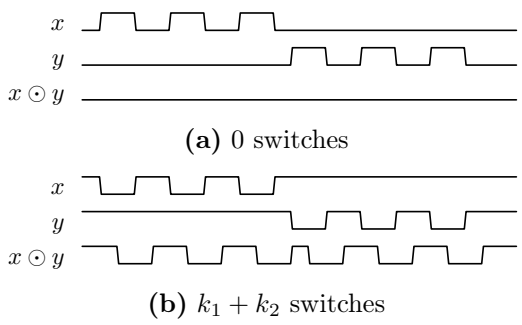
Then the number of output switches equals

$$K_{\mathrm{AND}}(a', b'; \mathcal{SW}) = \sum_{j=1}^{K} \Big(\mathbf{1}[e_j = A]\,(b' \oplus \beta_j) + \mathbf{1}[e_j = B]\,(a' \oplus \alpha_j) \Big), \quad (4)$$

Proof. Consider first the case where both inputs originate from register outputs, so that $k_1, k_2 \in \{0, 1\}$ and $\mathcal{SW} = \langle e_1, e_2 \rangle$. Since no two switches arrive simultaneously, the AND output can toggle at most once per event. If $e_1 = A$ and $e_2 = B$, the output toggles first iff $b' = 1$, and then iff $a' \oplus 1 = 1$ at the second event, yielding $K_{\mathrm{AND}} = (b') + (a' \oplus 1)$. Similarly, if $e_1 = B$ and $e_2 = A$, we obtain $K_{\mathrm{AND}} = (a') + (b' \oplus 1)$. These two cases correspond exactly to the two terms in Equation 4. For general $k_1, k_2 \in \mathbb{N}$, each event toggles exactly one input, and the output toggles at event j iff the non-switching input is 1 immediately prior to that event. The parities α_j and β_j track the instantaneous input values, and summing these per-event toggle conditions over all K events yields Equation 4. □

In contrast to the XOR gate, the AND gate's switching behavior may in some cases reveal the exact values of its inputs, but only when those inputs are equal to 1. Another important difference is that, for an AND gate, the order in which input switches arrive matters, as follows from Corollary 1. The same is also illustrated in Table 1 of the RR-probing model work [30]. Since the RR adversary is robust, it observes a glitch in both cases S_6 and S_9. This is possible because real hardware delays are not stable and may vary across clock cycles.

We illustrate the difference between the pure XOR-chain case from Section 3.1 and a combined design beginning with AND gates using the example in Figure 2. Unlike in the XOR case, we must now consider all orderings of the switches produced by x_0, y_0, y_1 and by the second branch of x_0 (to account for the rare case where a split wire induces slightly different delays). Moreover, the two AND gates cannot be analyzed independently, as both depend on the same transition pattern of x_0. As a result, we must consider $4! = 24$ possible delay-ordering permutations of the four relevant input transitions. We omit r, as it affects only the XOR sub-chain, which is composable regardless of ordering. In addition, we complement the cases with $4^4 = 256$ (4 per input wire) input-transition patterns, since the transition pattern itself influences the behavior of the AND gates. We omit the detailed calculations here. Instead, we implemented a program that enumerates all scenarios and produces $4! = 24$



(a) 0 switches

(b) $k_1 + k_2$ switches

Fig. 6. Signal switches behavior in the AND gate as a proof for Lemma 2.

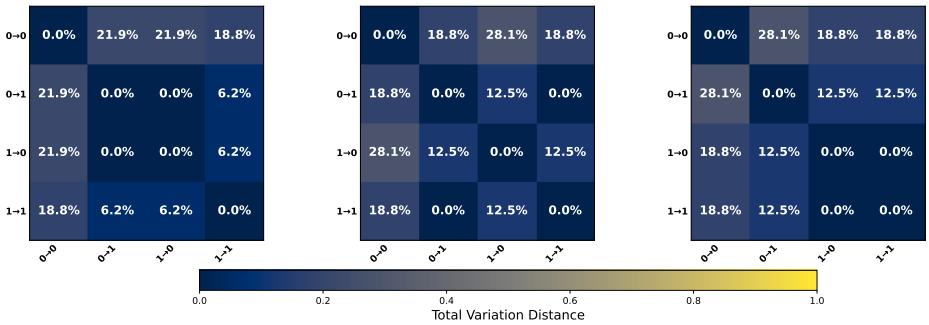


Fig. 7. Heatmaps, case of Section 3.2.

heatmaps, one for each possible delay-ordering. Among these 24 cases, only three distinct heatmaps appear, shown in Figure 7. These heatmaps fall into three groups: (i) twelve cases where the shares of y arrive in a different order than the corresponding branches of x_0; (ii) six cases where the y shares always arrive later; and (iii) six cases where the y shares always arrive earlier. This observation suggests that the analysis can be further optimized by exploiting this structural grouping. Namely, for secret-dependent behavior, only the relative ordering of the secret's shares matters, which reduces the analysis from factorial to exponential complexity. Introducing additional structural assumptions can reduce the computational complexity even further. Based on the input-arrival ordering, we can now compute the corresponding advantage of the HR-probing adversary for the circuit in Figure 2:

$$\mathrm{Adv}_{HR} \in [0.219,\, 0.281].$$

This aligns with our earlier claim that the adversary effectively observes the initial and final states of the probed wire, together with all intermediate switches, which collectively determine its ability to distinguish between different sets of secret transitions. However, the adversary does not necessarily know the order in which these switches were induced by the input signals. Furthermore, estimating the adversary's advantage becomes more complex when AND gates are preceded by XOR operations, as this increases the number of possible switch-propagation patterns affecting the AND gates.

Finally, the adversary's advantage depends on the specific pattern of delay propagation, which may vary even across logically equivalent implementations produced by different synthesis outcomes, and which, under perfect synchronization, may eliminate certain switches in the XOR chain. Consequently, the advantage values we compute represent a maximal adversarial advantage; certain implementations——especially those with long XOR chains combined with wide and balanced layers of AND operations——may exhibit significantly lower advantages and thus may not leak under distinguishability-based tests, such as TVLA.

Definition 3 (HR-probing adversary). *An HR-probing adversary observes, on a probed wire w and for each clock cycle, the pair of logic values* ($v_{\mathrm{prev}}(w)$, $v_{\mathrm{curr}}(w)$) *together with the full, time-ordered sequence of switching events* $\mathcal{SW}(w)$. *Given two secret-behavior hypotheses, the HR advantage equals* TVD *between the induced distributions of the observation. Feasible switch counts and schedules are constrained by Lemma 1, Lemma 2, and Equation 4.*

4 Case Study: AES S-box

We examine a masked implementation of the AES S-box to illustrate our research idea in practice. AES is a block cipher standardized by NIST [10], operating on a 128-bit state and key, each arranged as 16 bytes in a square structure. The cipher consists of 10 rounds, with each round applying the S-box to each byte, followed by ShiftRows and a matrix multiplication on the state's columns. The AES S-box is the only nonlinear layer of the cipher, operating on 8-bit values. Masking linear operations is straightforward, as each operation can be replicated for every masking share. By contrast, designing a securely masked AES S-box is challenging due to the complexity of finite field operations.

We re-implemented a design similar to that proposed by De Cnudde *et al.* [12], which employs the Canright decomposition technique [6] to enable masking of only low-degree functions. That implementation uses $d+1 = 2$ shares, providing security against a first-order SCA adversary. This minimal number of shares was chosen to facilitate the implementation of the linear operations across the full AES algorithm. In this work, we focus exclusively on the S-box implementation to demonstrate the applicability of our HR-probing model. In particular, we show that the distinguishability of the tested secret bits depends on the circuit layout, as different layouts induce different switch patterns. Since the original implementation is robustly secure, we deliberately remove one synchronization step to render it insecure and thus enable evaluation of our HR-probing model.

The decomposed S-box consists of six stages. For schematic details, we refer the reader to Figures 2 and 3 of the original work [12]. The first stage implements a linear map that performs a change of basis. The second stage implements a nonlinear multiplication in $\mathbb{GF}(2^4)$, followed by a square-scaling operation in $\mathbb{GF}(2^4)$, which is affine. At this stage, the data bus is divided into two 4-bit halves. Remasking is applied to the outputs of these operations to restore uniformity. The third stage is analogous to the second and implements similar operations in the smaller field $\mathbb{GF}(2^2)$. The outputs are again re-shared to maintain uniformity. The fourth stage motivates the preceding decompositions. In this stage, inversion in $\mathbb{GF}(2^2)$ is linear and is implemented by bit swapping. This layer is followed by two additional multipliers, whose outputs are concatenated to form four 4-bit shares. In addition to these operations, the registers following stages 2, 3, and 4 each store 4 bits derived from the originally split 8-bit input, with each bit represented by two shares. We highlight these buses because they are used in the subsequent fifth stage, which we investigate in more details with sixth stage.

4.1 Fifth and Sixth Stage Combination

Stage 5 has multiple inputs, which can be divided into three groups. The first two groups consist of the signals propagated from Stage 2 after the split following the linear-map stage. Each group comprises 4-bit buses divided into two shares. We denote these buses as Top_1, Top_2 and Bot_1, Bot_2, corresponding to the upper and lower groups, respectively. The third group consists of four buses of so-called middle shares, each also four bits wide. These shares are re-masked in the preceding stage and stored in the register before Stage 5, as they are compressed in Stage 5 into two middle shares, denoted Mid_1 and Mid_2. Combining corresponding shares, such as Top_1 with Top_2, Bot_1 with Bot_2, or Mid_1 with Mid_2, would directly reveal secret information. Therefore, Stage 5 combines only statistically independent variables using $\mathbb{GF}(2^4)$ multiplication blocks. As a result, the following products are computed:

$$Bot_1Mid_1 = Bot_1 \odot Mid_1, \qquad Top_1Mid_1 = Top_1 \odot Mid_1, \tag{5}$$
$$Bot_1Mid_2 = Bot_1 \odot Mid_2, \qquad Top_1Mid_2 = Top_1 \odot Mid_2, \tag{6}$$
$$Bot_2Mid_1 = Bot_2 \odot Mid_1, \qquad Top_2Mid_1 = Top_2 \odot Mid_1, \tag{7}$$
$$Bot_2Mid_2 = Bot_2 \odot Mid_2, \qquad Top_2Mid_2 = Top_2 \odot Mid_2. \tag{8}$$

We present the equations in this form because the corresponding multiplication results are concatenated, producing the output of Stage 5 as four 8-bit shares, with the $BotMid$ values forming the most significant nibble. In the original implementation, this block is considered secure because it does not fully recombine any of the shares, thereby preserving non-completeness. The subsequent refreshing step restores uniformity of the output masking and enables a further compression in Stage 6, as the introduction of fresh randomness maintains the non-completeness of the compressed terms as well. It is worth noting that the aforementioned uniformity issue arises only in the special case where the middle share equals the zero nibble, i.e., $(Mid^3, \ldots, Mid^0) = (0, 0, 0, 0)$. Stage 6 of the original design applies inverse linear maps to each of the two remaining shares after compression:

$$Out_1 = ILM(Bot_1Mid_1_Top_1Mid_1 \oplus Bot_1Mid_2_Top_1Mid_2), \tag{9}$$
$$Out_2 = ILM(Bot_2Mid_1_Top_2Mid_1 \oplus Bot_2Mid_2_Top_2Mid_2). \tag{10}$$

Clearly, if the register between Stages 5 and 6 is removed, a robust probing adversary gains the opportunity to observe both shares of the secret nibble Mid. The RR-probing adversary, while depending on the input data, still provides robustness and can therefore be extended to cover all input bits. However, we view this design as a particularly suitable example for evaluation under the HR-probing model. This intuition stems from the fact that a $\mathbb{GF}(2^4)$ multiplication can be represented as a long chain of XOR gates due to its internal structure. As discussed in Section 3.1, XOR chains propagate switches. In the case of $\mathbb{GF}(2^4)$ multiplication, each output bit may switch up to 18–22 times, depending on the configuration of the corresponding AND gates. To increase the number

of switches and further obscure their patterns, we swap the positions of the compression and ILM layers. As a result, this modification requires twice as many ILM instances. However, the resulting overhead amounts to only 116 gate equivalents (GE)[3], whereas removing the register between Stages 5 and 6 saves more than 140 GE.

4.2 Unbalanced Tree Version of the Design

We implement the combination of the fifth and sixth stages in two different ways, while keeping the remainder of the S-box identical to the original design of De Cnudde *et al.* For both approaches, we aimed to prevent the preceding $\mathbb{GF}(2^4)$ and ILM layers from being optimized by the netlist synthesizer. To this end, we enforced the absence of optimizations across module boundaries in our Verilog RTL using synthesis constraints, such as `keep_hierarchy` in Xilinx ISE and `set_dont_touch` together with `compile -exact_map` in Synopsys Design Compiler.

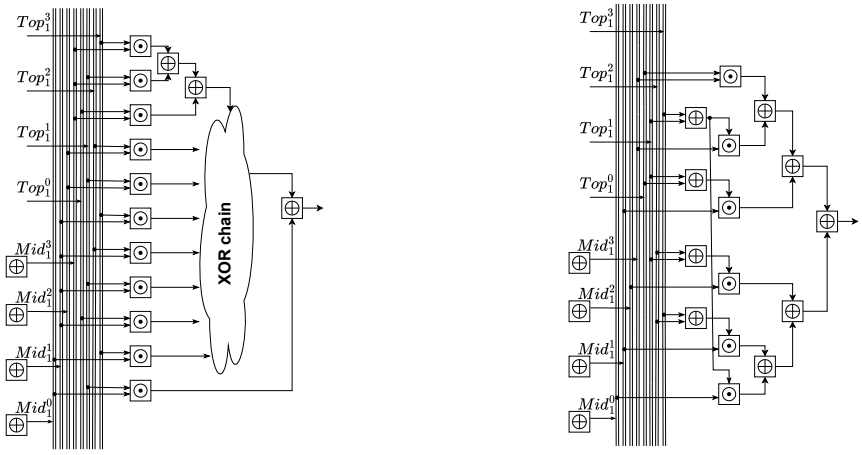
For the first design, we ensured that each gate in the combined stage is instantiated as a standalone submodule. In addition, XOR chain following AND-gate layer is implemented in an unbalanced manner, a portion of which, amounting to at least 1/4 of the entire compression chain, is shown in Figure 8(a)[4]. To estimate the TVD for a probe placed at the compression step, we optimize our code to operate on probabilistic switch propagation rather than enumerating all exact switch patterns. We further assume that the switching order observed after compression is stable, because the chain is unbalanced and the signals Top_i^j and Bot_i^j propagate faster, originating directly from registers. Under these realistic conditions, we obtain $Adv_{HR} \in [1.704e^{-3}, 2.991e^{-3}]$. This version fails the TVLA evaluation both on the FPGA (Section 5.1) and in the Spectre simulations (Section 5.2), exhibiting noticeable—though not severe—leakage. This behavior can be attributed to the unbalanced structure of the tree, which resembles the example discussed in Section 3.2.

4.3 Balanced Tree Version of the Design

To mitigate the likelihood of leakage, we redesign the combined stages such that the corresponding operation tree is balanced (Figure 8(b)). In addition, we allow the synthesizer to optimize the logic within each $\mathbb{GF}(2^4)$ multiplication block. In this way, security is preserved up to the compression step, while increasing the likelihood of unpredictable switching behavior. Balancing the tree reduces the stability of switching patterns, as the timing delays observed by an HR-probe become more closely aligned. Consequently, it becomes difficult to attribute a particular observed switch to a specific input signal. Focusing

[3] Evaluated using the NANGATE 45 nm open cell library

[4] To save space, we omit the full layout for a single S-box output bit, as it comprises more than 80 gates. The complete layout description can be found at `https://github.com/KULeuven-COSIC/HR_probing_model`



(a) Unbalanced tree **(b)** Balanced tree

Fig. 8. Two versions of the five-stage AES S-Box layout.

on a timing pattern analogous to the unbalanced-tree case—although potentially obscured by other delay permutations—yields an approximate estimate of $Adv_{HR} \in [1.2577e^{-4}, 2.4613e^{-4}]$. We evaluate this version using Spectre simulations and observe that it passes the TVLA test (Section 5.2).

5 Experimental Analysis

Our contribution to the experimental analysis is twofold. First, we evaluate the security of the AES S-box implemented on an FPGA. We begin with the six-stage implementation in order to reproduce and confirm the results of De Cnudde *et al.* [12], which report the absence of leakage. We then investigate two variants of specially configured five-stage implementations described in Sections 4.2 and 4.3. The goal of the latter experiments is to demonstrate how different combinational logic layouts can affect the leakage assessment. One should note that the results of FPGA-based tests may be influenced by several factors that can obscure leakage. These include, but are not limited to, platform-induced noise, noise generated by a PRNG located on the same evaluation board, placement and routing optimizations.

To mitigate the influence of these factors, we complement our FPGA experiments with a second contribution: Spectre simulations for the test cases described above. It allows to retrieve noiseless power simulations, since we only simulate the netlists of our AES S-boxes. Further details on Spectre and the methodology of these tests are provided in Section 5.2.

For the traces generated by both methods, we employ TVLA [3] to analyze the result and assess whether the applied tests detect potential leakage. This assessment method enables the observation of statistically significant differences

between two data sets using Welch's t-test. The standard practice is to compare the means of trace sets collected under fixed and random input conditions. A threshold crossing, commonly defined as $|t| > 4.5$, is treated as evidence of a statistically significant difference between the two trace sets and, consequently, as an indication of potential leakage. Conversely, the absence of a threshold crossing indicates that no statistically significant leakage was detected under the considered test configuration.

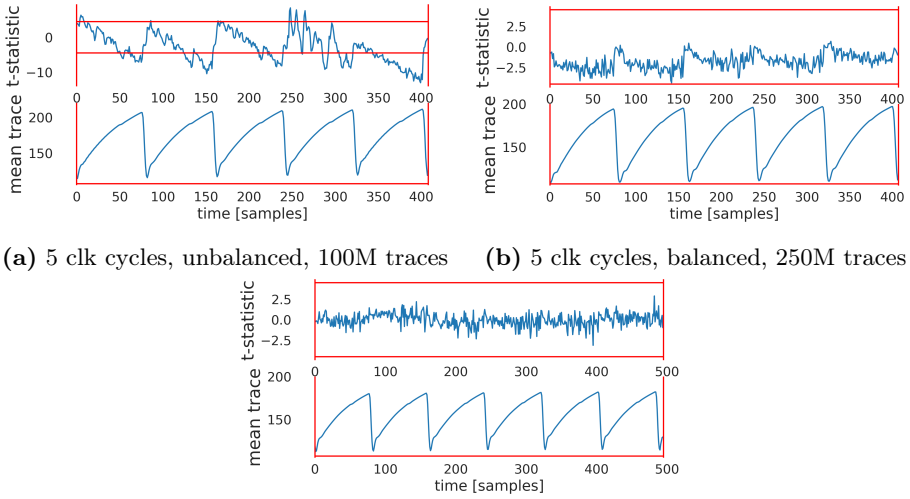
5.1 FPGA Experiments

For the FPGA-based experiments, we implement the AES S-box on the main chip of the Sakura-G board [18], which features a Spartan-6 FPGA. In addition, we deploy a pseudo-random number generator (PRNG) for fresh randomness generation based on Keccak-f[800] [4]. Since each S-box evaluation requires only 64 bits of randomness, Keccak-f[800] can be considered sufficient as a PRNG for up to 12 clock cycles. To ensure adequate entropy, we run Keccak-f[800] for approximately 24 clock cycles after providing the initial seed. The setup operates at a stable clock frequency of 6.144 MHz. The masked inputs are generated on the control board of the Sakura-G and therefore do not contribute to the measured power consumption of the S-box. We collect power consumption traces by monitoring the voltage drop across a $1\,\Omega$ shunt resistor placed on the VDD path of the target FPGA, using a digital oscilloscope with a sampling rate of 500 MS/s. This configuration allows us to capture approximately 80 sample points per clock cycle.

We begin by testing the original $d+1 = 2$ shared, first-order secure S-box, re-implemented based on the description by De Cnudde *et al.* [12], as a sanity check of our experimental setup. The results are shown in Figure 9(c), confirming the absence of leakage after collecting 100 million traces. We then consider the case described in Section 4.2 depicted in Figure 10(a). The first-order TVLA results exhibit clearly observable leakage, with the statistic growing to $|t| \approx 10$. The leakage is observed at every clock cycle, as we start supplying masked inputs during initialization stage prior catching the trace image, so the full pipeline is busy, meaning that there is an input reached stage 5 at the moment of the first clock cycle. Even though the leakage is observable, it is not considered extremely large for 100 million runs. We conclude our tests running the second version of five-stage implementation, implementing $\mathbb{GF}(2^4)$ blocks in the balanced way, described in Section 4.3. This TVLA result indicates no first-order leakage with 250 million traces collected, confirming our initial assumption. We proceed with simulation-based experiments to verify our model assumptions in a noiseless setting.

5.2 Spectre Experiments

Spectre is a transistor-level circuit simulator provided by Cadence Design Systems, capable of accurately simulating the currents and voltages in an integrated circuit. It is based on the SPICE simulation methodology and enables accurate,



(a) 5 clk cycles, unbalanced, 100M traces **(b)** 5 clk cycles, balanced, 250M traces

(c) 6 clk cycles, secure, 100M traces

Fig. 9. TVLA results for the AES S-boxes implemented on FPGA.

time-resolved analysis of switching activity for side-channel evaluation. Spectre requires only a netlist of the design and the corresponding cell library to run the simulations. For our experiments, we use the open-source FreePDK45 transistor models together with the NANGATE 45 nm cell library, and we synthesize the netlists using Synopsys Design Compiler. The synthesis tool is configured to prevent cross-module optimizations, ensuring that the individual RTL modules remain structurally distinct. Moreover, Spectre allows us to evaluate a standalone AES S-box without interference from the PRNG, as fresh random bits can be pre-generated and supplied directly as simulation inputs. Since no physical platform is involved, the results are free from platform-induced noise.

In contrast to FPGA-based measurements, Spectre does not use a fixed sampling rate; instead, it records voltage changes only at the time points where switching activity occurs in the simulated netlist. Although SPICE-based simulations have historically been considered complex and slow, taking between 4 and 26 hours to evaluate a masked DOM gate with an additional XOR operation, and potentially unreliable due to false positives introduced by Analysis Time-Step (ATS) optimizations [28], Spectre provides high temporal resolution precisely where it is needed most. To reduce false positives in our experiments, we use *Spectre X* configured in the most accurate *CX* preset. For the initial part of a clock cycle, where most glitches occur, new samples are generated with sub-picosecond precision (less than 0.1 ps), while during idle portions of the clock cycle, the sampling rate is reduced to 0.1 ns. This approach substantially improves simulation efficiency: we were able to evaluate our fully masked AES S-box implementations within 3–5 days, even when simulating twice the number of clock cycles required for proper initialization. In our experiments, we record the output signals as well as selected intermediate signals to verify the

correctness of our tests, which increases the data volume and, consequently, the simulation time.

As mentioned above, Spectre uses a dynamic sampling rate, which is illustrated in Figure 10. As a result, TVLA cannot be applied directly, because each trace contains a variable number of switching events in the combinational logic. To address this issue, we apply linear interpolation on a per-cycle basis as follows. First, we scan all traces to identify the clock edges (the stimulation frequency is fixed at 100 MHz, so each clock cycle has a constant duration). Second, for each cycle, we determine the trace that contains the largest number of samples. Finally, we interpolate all remaining traces to match this maximal sample count, ensuring uniform sampling across experiments.

We perform the same set of experiments previously carried out on the FPGA. For additional noise reduction, we supply only a single input value to the S-box. Furthermore, we allow several clock cycles to elapse to ensure that the pipeline is fully initialized and not affected by transient input noise. Because Spectre simulations are significantly more time-consuming and inherently noise-free, we collect a reduced number of traces. As a sanity check, we first evaluate the original De Cnudde design over 20,000 traces, again confirming the absence of leakage (Figure 10(a)).

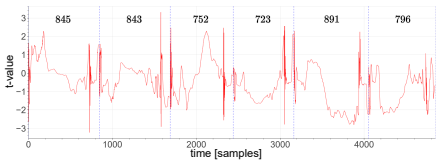
Next, we evaluate the reduced-latency design from Section 4.2. As shown in Figure 10(b), this version exhibits leakage, particularly in the penultimate stage, when the actual data is replaced by zeros. These observations confirm our intuition that an unbalanced tree is a suboptimal choice in settings with minimal noise.

Finally, we evaluate the second variant of our implementation described in Section 4.3. The balanced structure increases the indistinguishability of the timing of switches, and the subsequent recombination of inputs further reinforces this effect. As shown in Figure 10(c), this design exhibits no observable leakage.

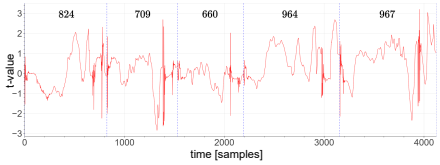
6 Conclusion

In this work, we demonstrated that conventional probing models, including the robust and RR-probing models, can be overly conservative for complex circuits where glitches interact in non-trivial ways. By introducing the HR-probing model, we provided a framework that captures data-dependent glitch propagation more accurately and reflects the practical limits of leakage observability. Using this model, we showed that the modified S-box implementation is secure against first-order attacks in both noiseless simulations and FPGA measurements, despite being theoretically vulnerable under standard robust probing assumptions.

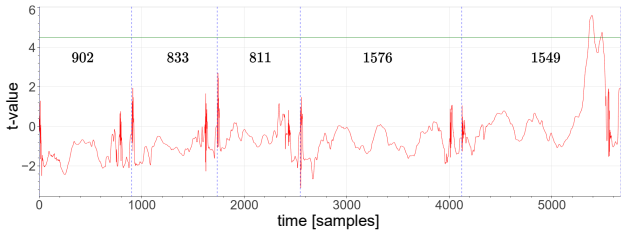
However, the HR-probing model is not yet fully formalized, and its applicability to general circuits remains limited, as the success probability of an HR-probing adversary depends strongly on the specific layout of the implementation. Future work should focus on generalizing the model to arbitrary designs and in-



(a) 6 stage S-box, secure version, 20k traces.



(c) 5 stage S-box, balanced tree version, 20k traces.



(b) 5 stage S-box, unbalanced tree version, 10k traces.

Fig. 10. TVLA results for the AES S-box simulated in Spectre.

tegrating it with standard cell libraries to enable automated leakage analysis of practical layouts.

Acknowledgments. This work was supported by the Flemish Government through the Cybersecurity Research Program with grant number: VOEWICS02. In addition, this work was partially supported by the Flemish Government through FWO Project "MatSec": G0ABK25N, and FWO mobility project: VS07724N. This preprint has not undergone peer review or any post-submission improvements. The Version of Record of this contribution is published in Lecture Notes in Computer Science (volume 16832), and is available online at `https://doi.org/10.1007/978-3-032-31130-6_9`.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 116–129. CCS 2016, Association for Computing Machinery (2016)
2. Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F.X., Strub, P.Y.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 535–566. Springer International Publishing (2017)

3. Becker, G., Cooper, J., DeMulder, E., Goodwill, G., Jaffe, J., Kenworthy, G., Kouzminov, T., Leiserson, A., Marson, M., Rohatgi, P., et al.: Test vector leakage assessment (tvla) methodology in practice. In: International Cryptographic Module Conference. vol. 1001, p. 13. sn (2013)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. pp. 313–314. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004. pp. 16–29. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
6. Canright, D.: A very compact s-box for aes. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2005. pp. 441–455. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
7. Cassiers, G., Standaert, F.: Trivially and efficiently composing masked gadgets with probe isolating non-interference. IEEE Trans. Inf. For. Sec. **15**, 2542–2555 (2020)
8. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 398–412. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
9. Coron, J.S., Giraud, C., Prouff, E., Renner, S., Rivain, M., Vadnala, P.K.: Conversion of security proofs from one leakage model to another: A new issue. In: Schindler, W., Huss, S.A. (eds.) Constructive Side-Channel Analysis and Secure Design. pp. 69–81. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
10. Daemen, J., Rijmen, V.: Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197 (Nov 2001)
11. De Cnudde, T., Bilgin, B., Gierlichs, B., Nikov, V., Nikova, S., Rijmen, V.: Does coupling affect the security of masked implementations? In: Guilley, S. (ed.) Constr. Side-Channel Anal. and Sec. Design. pp. 1–18. Springer Internn. Publ. (2017)
12. De Cnudde, T., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V., Rijmen, V.: Masking aes with d+1 shares in hardware. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptogr. Hardw. and Emb. Systems – CHES 2016. pp. 194–212. Springer Berlin Heidelberg
13. Dhooghe, S., Nikova, S., Rijmen, V.: Threshold implementations in the robust probing model. In: Proceedings of ACM Workshop on Theory of Implementation Security Workshop. p. 30–37. TIS'19, Association for Computing Machinery (2019)
14. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(3), 89–120 (2018)
15. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems — CHES 2001. pp. 251–261. Springer Berlin Heidelberg (2001)
16. Goubin, L., Patarin, J.: Des and differential power analysis the "duplication" method. In: Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems. pp. 158–172. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
17. Gross, H., Mangard, S., Korak, T.: Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In: Proceedings of the 2016 ACM Workshop on Theory of Implementation Security. p. 3. Association for Computing Machinery (2016)
18. Guntur, H., Ishii, J., Satoh, A.: Side-channel attack user reference architecture board SAKURA-G. In: IEEE 3rd Global Conference on Consumer Electronics, GCCE 2014. pp. 271–274. IEEE (2014)

19. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003. pp. 463–481. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
20. Knichel, D., Moradi, A.: Low-latency hardware private circuits. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 1799–1812. CCS '22, Association for Computing Machinery (2022)
21. Knichel, D., Moradi, A., Müller, N., Sasdrich, P.: Automated generation of masked hardware. IACR Trans. Cryptogr. Hardw. Embed. Syst. (1), 589–629 (2022)
22. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 388–397. Springer Berlin Heidelberg (1999)
23. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology — CRYPTO '96. pp. 104–113. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
24. Kumar, S.V.D., Balasch, J., Gierlichs, B., Verbauwhede, I.: Low-cost first-order secure boolean masking in glitchy hardware. IEEE Trans. Inf. For. Sec. **20**, 2437–2449 (2025)
25. Leiserson, A.J., Marson, M.E., Wachs, M.A.: Gate-level masking under a path-based leakage metric. In: Batina, L., Robshaw, M. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2014. pp. 580–597. Springer Berlin Heidelberg (2014)
26. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
27. Mangard, S., Pramstaller, N., Oswald, E.: Successfully attacking masked aes hardware implementations. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2005. pp. 157–171. Springer Berlin Heidelberg (2005)
28. Monta, K., Nagata, M., Balasch, J., Verbauwhede, I.: On the unpredictability of SPICE simulations for side-channel leakage verification of masked cryptographic circuits. In: 60th ACM/IEEE Design Automation Conference, DAC 2023. pp. 1–6. IEEE (2023)
29. Moradi, A.: Postponing the glitches is not enough - A critical analysis of the DATE 2024 E-ISW masking scheme (2025), `https://eprint.iacr.org/2025/2084`
30. Müller, N., Moradi, A.: Robust but relaxed probing model. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(4), 451–482 (2024)
31. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security. pp. 529–545. Springer Berlin Heidelberg (2006)
32. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. pp. 142–159. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
33. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: Attali, I., Jensen, T. (eds.) Smart Card Programming and Security. pp. 200–210. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
34. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology – CRYPTO 2015. pp. 764–783. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
35. Trichina, E.: Combinational logic design for AES subbyte transformation on masked data. IACR Cryptol. ePrint Arch. p. 236 (2003), `http://eprint.iacr.org/2003/236`

36. V., D.K.S., Dhooghe, S., Balasch, J., Gierlichs, B., Verbauwhede, I.: Time sharing - A novel approach to low-latency masking. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(3), 249–272 (2024)
37. Zhang, Z., Petkova-Nikova, S., Nikov, V.: Glitch-stopping circuits: Hardware secure masking without registers. In: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security. p. 3406–3420. CCS '24, Association for Computing Machinery (2024)

A Shared Multiplier of Stage 5

An example of $\mathbb{GF}(2^4)$ multiplication in the case of $Bot_1Mid_1 = Bot_1 \odot Mid_1$:

$$(\mathbf{Bot_1Mid_1})^{\mathbf{3}} = Bot_1^3Mid_1^3 \oplus Bot_1^1Mid_1^3 \oplus Bot_1^0Mid_1^3 \oplus Bot_1^2Mid_1^2 \oplus Bot_1^1Mid_1^2 \oplus Bot_1^3Mid_1^1 \oplus Bot_1^2Mid_1^1 \oplus Bot_1^1Mid_1^1 \oplus Bot_1^0Mid_1^1 \oplus Bot_1^3Mid_1^0 \oplus Bot_1^1Mid_1^0;$$

$$(\mathbf{Bot_1Mid_1})^{\mathbf{2}} = Bot_1^2Mid_1^3 \oplus Bot_1^1Mid_1^3 \oplus Bot_1^3Mid_1^2 \oplus Bot_1^2Mid_1^2 \oplus Bot_1^0Mid_1^2 \oplus Bot_1^3Mid_1^1 \oplus Bot_1^1Mid_1^1 \oplus Bot_1^2Mid_1^0 \oplus Bot_1^0Mid_1^0;$$

$$(\mathbf{Bot_1Mid_1})^{\mathbf{1}} = Bot_1^3Mid_1^3 \oplus Bot_1^2Mid_1^3 \oplus Bot_1^1Mid_1^3 \oplus Bot_1^0Mid_1^3 \oplus Bot_1^3Mid_1^2 \oplus Bot_1^1Mid_1^2 \oplus Bot_1^3Mid_1^1 \oplus Bot_1^2Mid_1^1 \oplus Bot_1^1Mid_1^1 \oplus Bot_1^3Mid_1^0 \oplus Bot_1^0Mid_1^0;$$

$$(\mathbf{Bot_1Mid_1})^{\mathbf{0}} = Bot_1^3Mid_1^3 \oplus Bot_1^1Mid_1^3 \oplus Bot_1^2Mid_1^2 \oplus Bot_1^0Mid_1^2 \oplus Bot_1^3Mid_1^1 \oplus Bot_1^0Mid_1^1 \oplus Bot_1^2Mid_1^0 \oplus Bot_1^1Mid_1^0 \oplus Bot_1^0Mid_1^0;$$