TSM$^+$ and OTSM

Correct Application of Time Sharing Masking in Round-Based Designs

Hemin Rahimi[1] and Amir Moradi[2]

[1] Independent Researcher
rahimihemn@gmail.com
[2] Technische Universität Darmstadt, Darmstadt, Germany
{firstname.lastname}@tu-darmstadt.de

Abstract. Among the countermeasures against side-channel analysis attacks, masking offers formal security guarantees and composability, yet remains challenging to implement efficiently in hardware due to physical defaults like glitches and transitions. Low-latency masking techniques aim to mitigate the performance penalties but can inadvertently compromise security in certain architectural contexts. In particular, the recently proposed Time Sharing Masking (TSM) technique enables single-cycle masked implementations with composability under the SNI and PINI notions but fails to satisfy stronger composability guarantees required in iterative designs, i.e., OPINI. In this work, we show that TSM-based constructions can exhibit first-order leakage when used in single-register feedback architecture, such as round-based implementations of ciphers. To address this, we propose two new masking schemes: TSM$^+$, a more efficient variant of TSM satisfying only PINI (but not SNI), and OTSM, a construction satisfying OPINI, enabling secure round-based designs.
Our improved round-based masked implementations of PRINCE and AES ensure security in latency-critical applications under both glitch- and transition-extended probing model while demanding for slightly more area consumption.

Keywords: Side-Channel Analysis · Masking · Low-Latency · TSM

1 Introduction

Ensuring the security of cryptographic devices and the data they process from exposure to various Side-Channel Analysis (SCA) attacks, such as electromagnetic-based [GMO01], power-based [KJJ99], and timing-based [Koc96] attacks, remains a critical challenge in cryptographic implementations. These attacks exploit unintentional information leakage through design characteristics, bypassing traditional cryptographic protections and compromising sensitive data, such as secret keys. As cryptographic devices are increasingly deployed in resource-constrained environments, ranging from Internet of things (IoT) to embedded systems, these vulnerabilities necessitate robust countermeasures at both the design and implementation levels to ensure resilience against sophisticated adversaries.

Among the countermeasures proposed against SCA attacks, masking [CJRR99] has emerged as the most studied one due to its sound theoretical foundation and the ability to formally prove security under well-defined adversarial models, e.g., the d-probing model [ISW03]. Masking achieves security by splitting sensitive data into randomized shares, ensuring that no single share reveals meaningful information about the sensitive data. This approach provides mathematically verifiable security guarantees, making it particularly appealing for high-assurance use cases. Hiding-based countermeasures, in contrast, founded on covering the exploitable leakage under the noise level via physical means (e.g., noise injection, operation shuffling, or power equalization), offer empirical

reductions in attack success rates but not the same level of formal guarantees [MOP07]. Their effectiveness is typically accompanied by implementation-specific parameters, such as noise level, the degree of randomization, or process variation, which are difficult to analyze generically or ensure across diverse platforms.

A large body of prior work has focused on masking in hardware [MRB18, MMM21, SM21, CSV24], where some challenging issues arise due to the physical defaults of circuit operation. Hardware masking has to deal with low-level physical effects like glitches and transitions that can reveal sensitive information, in contrast to software masking, which can function in a more controlled computational environment. Before stabilizing, glitches may cause intermediate values to propagate through a circuit, which could recombine masked shares and reveal unmasked data [MPG05, MPO05]. Similarly, transitions in sequential circuits, driven by clocked operations, may introduce leakage depending on consecutive values stored by registers which supply combinational circuits. Such issues undermine the theoretical security assurances of masking since adversaries can exploit such physical artifacts to reconstruct secret data through SCA techniques.

Several structures, notions, and concepts have been proposed to address the problems of physical defaults in hardware masking. By taking into consideration more realistic attack scenarios, the robust probing model improves upon the d-probing model [ISW03] to better handle physical leakage, such as glitches and transitions [FGP$^+$17]. Design strategies such as Threshold Implementation (TI) [NRS11], Domain-Oriented Masking (DOM) [GMK16], and their corresponding cryptographic implementations [MPL$^+$11, CBG$^+$17, GSM17] ensure circuit's resistance (to some extent) by preventing any single component from processing all shares of sensitive data. Additionally, composable security notions, such as Probe-Isolating Non-Interference (PINI), allow for securely composing individually masked components without adding new vulnerabilities [CS20]. These advancements solve the issues with circuit-level implementation while enhancing masking ability to defend cryptographic hardware against SCA attacks.

With a clearer understanding of how masking in hardware can be utilized effectively to address the issues stated, attempts have shifted towards reducing the overheads involved and improving efficiency [SBHM20, MMM21, CSV24]. Techniques like TI, while robust, as powerful as they are, tend to consume a lot of resources, which leads to degradation in multiple directions. One of such directions is latency, or the number of clock cycles it takes to get the operation done. More specifically, masking circuits based on conventional approaches (e.g., TI or DOM) add extra register stages to stop glitches from spreading and to ensure that the timing of calculations does not expose sensitive information [GSM17]. These registers play a key role in security, but they slow operations down by adding more cycles. To diminish the latency in masking circuits, researchers are developing new design techniques and facilities to employ fewer registers or optimized designs to enhance the circuit performance while still keeping it secure against SCA attacks through a low-latency design approach. Examples include like but not restricted to Generic Low-latency Masking (GLM) [GIB18], Generic Hardware Private Circuits (GHPC) and its low-latency variant (GHPC$_{\text{LL}}$) [KSM22], HPC3 [KM22], HPC4 [CSV24], LMDPL [SBHM20], and Time-Sharing Masking (TSM) [VDB$^+$24] to make circuits faster while staying secure against SCA attacks. These methods aim to keep the number of clock cycles low, naturally at the cost of a higher area footprint.

GLM is a low-latency masking approach based on DOM, which was originally designed to develop low-area and low-randomness masked designs. In DOM, to avoid leakage originating from non-linear operations over cross-domain shares, they are first blinded (refreshed by a fresh mask) and stored in a register. In order to reduce the latency, this register layer has been removed in GLM, which naturally leads to increased randomness and area overhead. GHPC is a design framework for constructing secure and composable masked gadgets for arbitrary vectorial Boolean functions. The framework ensures that

the resulting circuits are composable by relying on the PINI security notion [CS20] over glitch-extended probing model. A secure implementation equipped with GHPC generally requires two register layers; one to refresh intermediate values and another to store the processed results. Its low-latency variant GHPC_{LL} reduces this to a single register layer at the cost of increased randomness. In GHPC which is limited to first-order security, the number of required registers and randomness depends on the number of inputs of the target function rather than its algebraic degree. Both HPC3 and HPC4 are known as PINI composable 2-input non-linear gadgets at arbitrary security order with only one register stage. It means that the latency of a circuit composed by such gadgets equals the number of non-linear operations cascaded. HPC4 additionally satisfies the Output Probe-Isolating Non-Interference (OPINI) notion, allowing it to be trivially composable even in iterative designs consisting of only one quadratic function. LMDPL pursues a different path, however, by constructing composable gadgets that can be composed without increasing latency. This is achieved by employing Dual-Rail Pre-charge (DRP) logic in the construction of masked gadgets. Similar to GHPC, it can only provide first-order security.

The most recent approach TSM markedly enhances the efficiency of low-latency masking for arbitrary functions, and achieves substantial area reduction in single-cycle masked composable circuits, e.g., 32% area savings for the PRINCE core compared to the state of the art. Unlike the other masking schemes that separate shares spatially, TSM processes shares in separate stages to achieve non-completeness. Complex cryptographic implementations can be developed more easily due to their conformity to the SNI [BBD+16] and PINI[CS20] security notions, which provide the trivial composition of masked components into larger, secure designs. TSM was initially limited for first-order security, but it has been expanded to higher orders [VDB+25].

The ability of TSM to construct first-order secure large non-linear functions, like Sboxes, with just one register stage inside the circuit is a major benefit. By eliminating the need for additional register layers, this simplified design makes it possible to implement masked cryptographic primitives efficiently following a round-based architecture. In such configurations, the low-latency masked circuit can complete one cipher round per clock cycle without pipelining. However, being SNI and/or PINI is not enough to guarantee security in such a construction. In particular, if the output of a single-register circuit is fed back to its input in the next immediate clock cycle, compliance with the SNI and PINI security notion may not suffice to maintain security. This has been shown in [CS21], where the composable security notion OPINI is introduced. Consequently, in the construction mentioned above, the security is guaranteed if the circuit realizing the cipher round function is in conformity with the OPINI security notion.

Our Contributions. In this work, we address the security limitations of TSM in round-based architectures by demonstrating its vulnerability and proposing an improved technique to overcome this shortcoming while maintaining the low-latency feature. A detailed list of our contributions is given as follows.

- We first show that the constructions presented by the authors in [VDB+24] are not in conformity with OPINI security notion, and demonstrate, through examples, that they may lead to first-order leakage if they are used in the aforementioned single-register round-based architecture. Along the same line, we show the first-order insecurity of both TSM round-based PRINCE and AES designs provided by the original authors.
- To cope with this issue, we propose a few solutions, including a circuit-level modification as well as two constructions inspired by TSM, given below.
  - Our TSM^+, which is an alternative to TSM, is in conformity only with PINI (not SNI) notion while requiring less fresh masks.

  - Our OTSM, which follows the same architecture as our TSM$^+$, is additionally in conformity with the OPINI notion at the cost of more fresh masks. This can, however, be trivially composed even in designs where the output is directly used as the next input, i.e., the aforementioned single-register round-based architecture.

- We additionally illustrate that it might be not necessary to instantiate only Sboxes in conformity with the OPINI notion in a round-based implementation of a block cipher. A careful combination of PINI and OPINI Sboxes may suffice to guarantee the security of the composition. This depends on the diffusion layer of the underlying cipher and the design architecture of the round-based implementation. As a solid result, our first-order round-based design of PRINCE and AES making use of TSM$^+$ and OTSM Sboxes guarantee first-order security under the glitch- and transition-extended probing model while requiring slightly more area compared to the original counterparts made solely by TSM Sboxes.

2 Background

This section introduces the fundamental concepts that support the methods and security proofs discussed in the rest of the paper.

2.1 Notations

In equations, brackets $[\,.\,]$ serve as a register. $\oplus$ is used to indicate addition in $\mathbb{F}_2^n = \mathrm{GF}(2^n)$. `AND` operations have been presented by "&" and in some cases, this notation has been omitted for brevity. By subscripts x_i we refer to the i-th element (bit) of a vector X, and by superscript x^j to the j-th share of x. Boolean functions are denoted by sans-serif fonts $\mathsf{f} : \mathbb{F}_2^n \to \mathbb{F}_2$, vectorial Boolean functions by capital sans-serif fonts $\mathsf{F} : \mathbb{F}_2^n \to \mathbb{F}_2^m$, and sets by calligraphy fonts $\mathcal{G}$.

2.2 Boolean Masking

Boolean masking is a countermeasure used to protect sensitive information from SCA, such as power analysis attacks. Its underlying concept is to randomize intermediate values within a circuit, thereby avoiding the correlation between power consumption and the processed data, such as the intermediates of a cipher. Under d-th order masking, this technique splits secret information $X \in \mathbb{F}_2^n$ into $d+1$ shares $\langle X^0, \ldots, X^d \rangle \in (\mathbb{F}_2^n)^{d+1}$. A common approach for generating these shares is to select d shares (e.g., $\langle X^0, \ldots, X^{d-1} \rangle$) uniformly at random over $(\mathbb{F}_2^n)^d$, and compute the remaining share (here X^d) as the XOR of the original value X with all the other shares. Formally, it can be written as $X = \bigoplus_{i=0}^{d} X^i$.

2.3 Security Models

Here, we present the robust probing model and the robust but relaxed probing model, highlighting their role in assessing the security of masked cryptographic implementations and completing the gaps in the standard probing model. Thus, let us begin with the standard probing model.

2.3.1 Standard Probing Model

The standard d-probing model, introduced in [ISW03], examines the security of masked cryptographic implementations by taking into account that the attacker can place at most

d probes on arbitrary internal wires within a circuit. A probe allows the attacker to observe the value of the target wire, i.e., noise-free and static signal. A masked circuit is considered secure in the d-probing model, if the joint distribution of any d observed signals does not contain any information about the secret information. The model provides a theoretical foundation for analyzing the security of masking schemes, but does not consider physical phenomena and dynamic hazards such as glitches and transitions that lead to additional information leakage and make d-probing secure circuits vulnerable in practice [MPO05].

2.3.2 Robust Probing Model

As an extension of d-probing model, the robust d-probing model has been introduced in [FGP+17], considering the impact of the physical defaults in hardware implementations. In addition to probed signals, it considers information leakage caused by unintended signal transitions (glitches) in combinational circuits [MPO05], unintended state transitions in memory cells [CGP+12], and signal interference and crosstalk in routing paths [CBG+17]. This model gives a more complete framework for assessing the security of cryptographic hardware implementations in non-ideal physical environments.

To formally capture the effect of such physical defaults, the robust d-probing model provides an extension mechanism that generalizes standard probes to model various forms of leakage. Based on the type of leakage to be modeled, the model provides three dedicated extensions: glitch-extension, transition-extension, and coupling-extension. Each of these extensions models the respective features of the underlying physical phenomenon and extends the adversary's ability to derive sensitive information from the hardware implementation.

Definition 1 (Glitch-Extension)**.** The glitch-extension considers information leakage through glitches, which are unintentional signal transitions in combinational circuits. Glitches are caused by unequal delay paths and switching delays making input signals arrive asynchronously at a gate and temporarily generate incorrect outputs before stabilizing. In the robust probing model, glitch-extended probes enable an attacker to observe all stable intermediates (e.g., register outputs or the circuit's primary inputs) that contribute to the probed wire.

Definition 2 (Transition-Extension)**.** The transition-extension captures leakages resulting from transitions between two consecutive values stored in registers. Hence, by observing the transition between the two successive states, the attacker can learn information about both states. In other words, when a probe reaches the output of a memory cell (register), it can be propagated to its input via transition-extension.

Definition 3 (Coupling-Extension)**.** The coupling-extension accounts for leakage due to signal interference and crosstalk between adjacent wires. By probing a signal, the attacker will be able to observe multiple neighboring wires at the same time.

Since coupling-extension relies on the physical proximity of wires and requires post-fabrication information about the circuit's layout, it is commonly ignored when dealing with the masking schemes at the design level. In this work, we also cover only glitch- and transition-extensions.

2.3.3 Robust but Relaxed Probing Model

While the robust d-probing model provides a complete framework for investigating the security of masked cryptographic circuits, it tends to be overly conservative, particularly for glitches. It assumes that the adversary is able to observe all possible values contributing to the combinational circuit generating a probed wire, even if some of the glitches are physically impossible to occur. This over-conservativeness leads to additional design

complexity, additional resource utilization, and potentially secure designs being ruled out under the strict requirements of the robust probing model.

To address this limitation, the Robust but Relaxed (RR) d-probing model has been introduced in [MM24] that improves the glitch-processing mechanism by considering only physically realistic glitches depending on the gate operations and data handled. Unlike the robust model, which assumes that a glitch can always propagate, the RR model employs a conditional probe-extension mechanism that examines whether a glitch can affect the output signal. It further does not allow to separate glitches from transitions since glitches are the result of transitions in registers or primary inputs. This relaxed approach reduces the number of probes extended as well as the computational overhead without compromising the degree of security.

2.4 Circuit Composition

Ensuring the security of a cryptographic implementation involves extensive evaluation using a variety of tools to verify its resilience to physical attacks. However, security assessment becomes computationally expensive and less scalable 1) as circuit complexity increases and 2) at higher security orders. Meanwhile, the concept of composable security, in which a large design is made up of smaller modules, each of which is a secure building block called a gadget, became a key factor. Composable gadgets allow for modular security verification, where the security of the entire circuit can be derived from the security of the components. This reduces the computational overhead of large-scale direct evaluation and allows for more efficient design and verification flows.

Composing stand-alone secure modules would not necessarily result in a secure circuit. Therefore, several security notions for composable security have been introduced to formalize the conditions under which modular security holds. These notions define how the security of individual components translates to the security of the entire circuit when composed. After having flawed or highly inefficient notions like Non-Interference (NI) [BBD$^+$15] and Strong Non-Interference (SNI)[BBD$^+$16], the focus has shifted to PINI [CS20] and its extension, OPINI [CS21], which provide guarantees for composable security in masked implementations. Before diving into the details of these notions, let us restate the simulatability framework.

Definition 4 (Simulatability [BBP$^+$16])**.** Let a masked gadget $\mathcal{G}$ take n binary input variables, each masked with $d+1$ shares according to a d-th order masking scheme. Let $x_*^* = \left\{x_i^j \mid 0 \le i \le n-1,\ 0 \le j \le d\right\}$ denote the complete set of input shares, where x_i^j is the j-th share of the i-th input variable. A set of probes $\mathcal{P} = \left\{p_1, \ldots, p_\ell\right\}$ observed during the execution of $\mathcal{G}$ is said to be *simulatable* from a subset of input shares indexed by

$$\mathcal{I} = \left\{(i^0, j^0), \ldots, (i^{k-1}, j^{k-1})\right\} \subseteq \left\{0, \ldots, n-1\right\} \times \left\{0, \ldots, d\right\},$$

if there exists a *randomized simulator* $\mathcal{S} : \mathbb{F}_2^k \to \mathbb{F}_2^\ell$ such that, for any input x_*^*, the joint distribution of the probed values $\mathcal{G}_\mathcal{P}(x_*^*)$ is identical to the distribution produced by $\mathcal{S}\left(x_{i^0}^{j^0}, \ldots, x_{i^{k-1}}^{j^{k-1}}\right)$. That is $\mathcal{G}_\mathcal{P}(x_*^*) \overset{d}{=} \mathcal{S}\left(x_{i^0}^{j^0}, \ldots, x_{i^{k-1}}^{j^{k-1}}\right)$, where $\overset{d}{=}$ denotes equality of distributions. Intuitively, this means that the adversary gains no additional information by observing the probes $\mathcal{P}$ beyond what is already available from the input shares indexed by $\mathcal{I}$.

Definition 5 (PINI [CS20])**.** Consider a gadget execution $\mathcal{G}$ with a set $\mathcal{I}$ containing at most t_1 probes on its internal wires and a set $\mathcal{O}$ comprising probes on its output shares. Let $\mathcal{A}$ denote the set of share indices associated with the shares in $\mathcal{O}$, where $t_2 = |\mathcal{A}|$, and let $\mathcal{I}$ and $\mathcal{O}$ be selected such that the total number of placed probes t satisfies $t_1 + t_2 \le t$. The gadget $\mathcal{G}$ is classified as t-PINI if, for every choice of $\mathcal{I}$ and $\mathcal{O}$, there exists a set $\mathcal{B}$

of at most t_1 share indices such that the combined observations from $\mathcal{I}$ and $\mathcal{O}$ can be perfectly simulated using only all input shares indexed by $\mathcal{A} \cup \mathcal{B}$.

While the PINI security notion facilitates trivial composition by ensuring security when the probes are propagated via glitches, it does not account for transitions between states if the sharing of two consecutive inputs of the gadget is not independent. To achieve trivial composability in the presence of both glitches and transitions, the notion of OPINI has been introduced, providing a more robust framework for secure composition in such scenarios.

Definition 6 (OPINI [CS21])**.** Consider a gadget execution $\mathcal{G}$ with a set $\mathcal{I}$ comprising at most t_1 probes on its internal wires and a set $\mathcal{O}$ consisting of probes on its output shares. Let $\mathcal{A}$ represent the set of share indices corresponding to the shares in $\mathcal{O}$, where $t_2 = |\mathcal{A}|$, and let $\mathcal{I}$ and $\mathcal{O}$ be selected such that the total number of placed probes t satisfies $t_1 + t_2 \leq t$. The gadget $\mathcal{G}$ is defined as t-OPINI if, for every choice of $\mathcal{I}$ and $\mathcal{O}$, there exists a set $\mathcal{B}$ of at most t_1 share indices such that the combined observations from $\mathcal{I}$, $\mathcal{O}$, and the output shares indexed by $\mathcal{B}$ can be perfectly simulated using only all input shares indexed by $\mathcal{A} \cup \mathcal{B}$.

By adopting the OPINI notion, a circuit achieves composability under both glitch- and transition-extensions, ensuring robust security in gadget executions [CS21]. This enhancement over the PINI notion, which primarily addresses glitches, is particularly critical in round-based architectures where transitions between consecutive clock cycles can introduce vulnerabilities, as observed, reported, and solved in [CSV24].

3 TSM and the Problem

While the well-known masking schemes effectively protect against SCA attacks, their overhead is not necessarily affordable. They typically incur high costs in terms of area, latency, and randomness, especially when applied on complex functions such as large Sboxes. Although solutions like GLM [GIB18] and GHPC [KSM22] mitigate latency, they are still considered inefficient in terms of area and randomness. In this section, we review the concept of TSM [VDB+24] and discuss the issues arising from its deployment. We elaborate on how TSM enhances the efficacy of masked cryptographic hardware implementations, precisely what it addresses, and what issues may arise in the presence of glitches and transitions. Therefore, let us begin with its underlying concept.

3.1 TSM

The core idea of TSM – as a novel first-order masking technique for low-latency hardware implementations – is to separate the processing of shares in time instead of in space, to achieve non-completeness. More specifically, after refreshing the input shares, it splits the processing of the shares into two stages (see Figure 1).

- In the first stage, cross-product combinations of the first input shares are computed ($\mathcal{G}_i$ function), blinded, and stored in registers.
- In the second stage, cross-product combinations of the second input shares are computed ($\mathcal{H}_i$ function) and combined with the blinded results from the first step.

With the single register stage, TSM can be applied to any (vectorial) Boolean function. Considering $X = X^0 \oplus X^1$ as the primary input, R and R' the given randomness, and

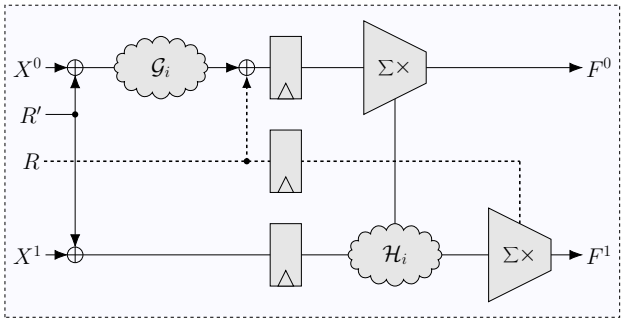


Figure 1: Architecture of the TSM scheme.

$F = F^0 \oplus F^1$ the primary output, the following equations are given.

$$\begin{aligned} F^0 &= \sum_{\mathcal{I}\in\mathcal{P}_k} \Big(\mathcal{G}_{\pi(\mathcal{I})}\big((x_i^0)_{i\in\mathcal{I}}\big) + r_{\pi(\mathcal{I})}\Big)\mathcal{H}_{\pi(\mathcal{I})}\big((x_i^1)_{i\in\Omega/\mathcal{I}}\big), \\ F^1 &= \sum_{\mathcal{I}\in\mathcal{P}_k} r_{\pi(\mathcal{I})}\mathcal{H}_{\pi(\mathcal{I})}\big((x_i^1)_{i\in\Omega/\mathcal{I}}\big), \end{aligned} \tag{1}$$

where the set of x_i^0 for all $i \in \mathcal{I}$ is denoted by $(x_i^0)_{i\in\mathcal{I}}$. Stated differently, this represents the bits of the first input share indexed by the elements of the set $\mathcal{I}$. Similarly, the set of bits from the second input share whose indices are not in $\mathcal{I}$ is denoted by $(x_i^1)_{i\in\Omega/\mathcal{I}}$. With k being the input size, the set $\mathcal{P}_k$ is the power set of the indices $\Omega = \{0, \ldots, k-1\}$, which includes all possible subsets of indices from Ω. The elements of $\mathcal{P}_k$ (which is of a size 2^k) are assigned unique indices using a function π that maps each subset in the power set to a distinct index.

The authors have proven that TSM is in conformity with 1-SNI and 1-PINI under glitch-extended probing model. This indicates that the masked implementations in TSM remain first-order secure even when combined with other PINI components. Exemplarily, consider a 2-input masked AND gate $z = x\,y$, where $x = x^0 \oplus x^1$, $y = y^0 \oplus y^1$, $z = z^0 \oplus z^1$, as illustrated in Figure 2. In the first step, the input shares are refreshed as follows.

$$x'^0 = x^0 \oplus r_3, \qquad x'^1 = x^1 \oplus r_3, \qquad y'^0 = y^0 \oplus r_4, \qquad y'^1 = y^1 \oplus r_4$$

The output is then computed based on Equation (1) as

$$\begin{aligned} z^0 &= [x'^0 y'^0 \oplus r_0] \oplus [x'^0 \oplus r_1][y'^1] \oplus [y'^0 \oplus r_2][x'^1], \\ z^1 &= [r_0] \oplus [r_1][y'^1] \oplus [r_2].[x'^1] \oplus [x'^1][y'^1]. \end{aligned} \tag{2}$$

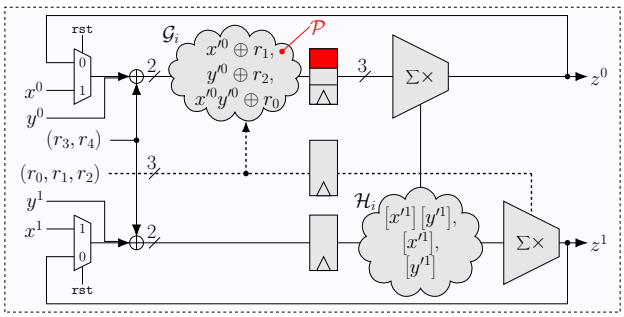


Figure 2: TSM, looped 2-input AND.

Table 1: Implementation cost of an $\mathbb{F}_2^k \to \mathbb{F}_2^k$ function with an algebraic degree of $k-1$.

Scheme	# of Registers	# of Fresh Masks	# of Register Layers
GHPC_{LL} [KSM22]	$2^k k + k$	$2^k k$	1
GLM [GIB18]	$2^k k$	$2^k k$	1
TSM [VDB+24]	$2^{k+1} + k - 4$	$2^k + k - 2$	1

In this design, shown in Figure 2, any probe placed on output or intermediate of $\mathcal{G}$ would reveal only the first input shares, and the probe placed on z^1 observes only the second input shares blinded by r_3 and r_4. The probe placed on z^0 would observe the same blinded second input shares as well as the registered outputs of $\mathcal{G}$ which are also blinded but by another set of randomness r_0 to r_2. In short, any single glitch-extended probe placed on this circuit would not reveal any information associated to different share indices.

One important aspect of evaluating the efficiency of TSM is the number of utilized registers and the amount of randomness required. The total number of registers corresponds to the storage needed for intermediate values between the two processing stages. According to Table 1, the equations for register utilization and randomness are derived based on the Algebraic degree of the masked function. Specifically, TSM reduces the number of required registers compared to GLM and GHPC_{LL} while minimizing the number of fresh randomness required to secure the intermediate values.

As an example for a TSM implementation, consider a function $\mathbb{F}_2^k \to \mathbb{F}_2^k$ with $k = 4$. It requires 32 registers (14 for the first share domain after the $\mathcal{G}$ function, 14 for intermediate fresh masks, and 4 for the second share domain) and employs 18 fresh masks (4 bits as the input refresher and 14 bits for blinding the intermediates). Despite these reductions in terms of the number of registers and the demand for fresh randomness compared to the former works, which result in a lower area overhead, an optimization technique, reusing lower-degree Algebraic Normal Form (ANF) term(s) in the higher degree computations, has been applied, further enhancing the efficiency with respect to the area overhead. As a side note, the ANF of the given function[1] is utilized when constructing its TSM.

3.2 The Shortcoming

Although TSM-based implementations can handle glitches by appropriately making use of fresh randomness and separating the processing stages, their security can be compromised when glitches and transitions are taken into account at the same time. The problem occurs because transitions lead to state holding between consecutive clock cycles, while glitches are handled at the combinational level [CS21]. The independence between consecutive intermediate values – which is an essential assumption in PINI circuits – may be compromised when the input of the circuit is a function of its output. This introduces potential vulnerabilities in real-world implementations and complicates the security proof, demanding consideration of both glitches and transitions.

3.2.1 2-Input AND

To provide further clarification, we analyze the security of the previously discussed TSM 2-input AND circuit when it is instantiated within a looped architecture (see Figure 2). This analysis highlights how temporal effects such as transitions, in conjunction with glitches, may influence the leakage behavior of such a masking scheme.

The functionality of the looped circuit shown in Figure 2 is as follows: the signal `rst` is initially enabled (set to 1), which allows the first primary input (the shares of x) to be loaded into the circuit. In the subsequent clock cycles, `rst` is disabled (set to 0). As

[1] It is a canonical representation.

a result, one of the inputs of the AND circuit (the first shared input in this design) is taken directly from the previously computed output. This value is then ANDed with the second input y to produce the next output. If we denote unmasked input y in the first three consecutive clock cycles by $y_{(1)}, y_{(2)}, y_{(3)}$, the circuit calculates $\big((x\, y_{(1)})\, y_{(2)}\big)\, y_{(3)}$ after three clock cycles. Hereafter, we denote the value of a signal in a specific clock cycle by a subscript in parentheses.

To evaluate the security of this design under the glitch- and transition-extended probing model, we focus on the inputs to the registers placed after $\mathcal{G}_i$, which represent 3-bit internal values. For the sake of simplicity in this analysis, we assume that the given primary inputs x and y do not change. At the first clock cycle, when `rst` is set to 1, x and y are given to the AND circuit, and in the second clock cycle, when `rst` is set to 0, z and y are given while $z = x\, y$. In this second clock cycle, let us place a probe $\mathcal{P}$ on the input of the first register (marked in red in Figure 2); the probe then observes $\mathcal{P} : \left\{x'^0_{(2)} \oplus r_{1(2)}\right\}$. Through glitch extension (via the XOR gates and the multiplexer), $\mathcal{P}$ propagates to $\left\{x^0,\ z^0_{(2)},\ r_{3(2)},\ r_{1(2)}\right\}$. Since x and y are supposed to be constant during the circuit evaluation, we omit their clock cycle notation for brevity, i.e., x^0 instead of $x^0_{(2)}$. If we now extend the probe on $z^0_{(2)}$, we can write (see Equation (2)):

$$\mathcal{P} : \left\{x^0,\ \left[x'^0_{(1)}\, y'^0_{(1)} \oplus r_{0(1)}\right],\ \left[x'^0_{(1)} \oplus r_{1(1)}\right],\ \left[y'^1_{(1)}\right],\ \left[y'^0_{(1)} \oplus r_{2(1)}\right],\ \left[x'^1_{(1)}\right],\ r_{3(2)},\ r_{1(2)}\right\}.$$

Among these probes, we concentrate on only a smaller set $\mathcal{P}'$ as $\left\{x^0,\ \left[x'^1_{(1)} = x^1 \oplus r_{3(1)}\right],\ r_{3(2)}\right\}$, whose elements are either a primary input or a register output. Therefor, by applying the transition extension (on only $r_{3(2)}$) we reach $\mathcal{P}' : \left\{x^0,\ \left[x^1 \oplus r_{3(1)}\right],\ r_{3(2)},\ r_{3(1)}\right\}$, leading to reveal both x^0 and x^1, i.e., x.

We have confirmed this vulnerability by employing `PROLEAD` [MM22], detecting first-order leakage in this looped circuit when probes are allowed to propagate via both glitches are transitions. The design files, synthesized netlist, and corresponding evaluation results are publicly via GitHub[2].

This leakage stems from the fact that TSM ensures conformity with the SNI and PINI security notion, but does not satisfy the OPINI security notion. It is important to stress that the observed leakage in this circuit arises from the combined effect of transitions and glitches. If the analysis is restricted solely to glitch propagation (i.e., excluding the transition-based extension of probes), the looped circuit remains first-order secure. This highlights the crucial role of transition-aware modeling in accurately assessing the composability of masking schemes in sequential and looped architectures.

Here, we show that the TSM 2-input AND circuit is not in conformity with OPINI notion. Concentrating on first-order security $t = 1$ and based on Definition 6, when a single probe $\mathcal{P}$ is placed on an internal node of an OPINI circuit (i.e., $t_1 = 1$ and $t_2 = 0$), the probe $\mathcal{P}$ and z^0 should be simulatable only by the input shares $x^0,\ y^0$, or analogously the probe $\mathcal{P}$ and z^1 should be simulatable only by the input shares $x^1,\ y^1$.

Focusing on the first case, we suppose that the probe $\mathcal{P}$ is the same as the one we placed on the looped circuit, i.e., the input of the first register marked in red in Figure 2. By extending the probes via glitches, we reach $\mathcal{P} : \left\{x^0,\ y^0,\ r_1,\ r_3,\ r_4\right\}$, while the probe on z^0 extends to $\mathcal{Z}^0 : \left\{\left[x'^0\, y'^0 \oplus r_0\right],\ \left[x'^0 \oplus r_1\right],\ \left[y'^1\right],\ \left[y'^0 \oplus r_2\right],\ \left[x'^1\right]\right\}$. Since $x'^1 = x^1 \oplus r_3$ (resp. $y'^1 = y^1 \oplus r_4$), it is trivial that both share domains $x^0,\ x^1$ (resp. $y^0,\ y^1$) are necessary to simulate $\mathcal{P}$ and $\mathcal{Z}^0$, i.e., not being simulatable only by x^0 and y^0.

In the second case, independent of how the probe on z^1 is extended, $\mathcal{P}$ contains x^0 (resp. y^0). Therefore, it is trivial that it cannot be simulated only by $x^1,\ y^1$. Hence, it

[2] https://github.com/ChairImpSec/TSMp-OTSM

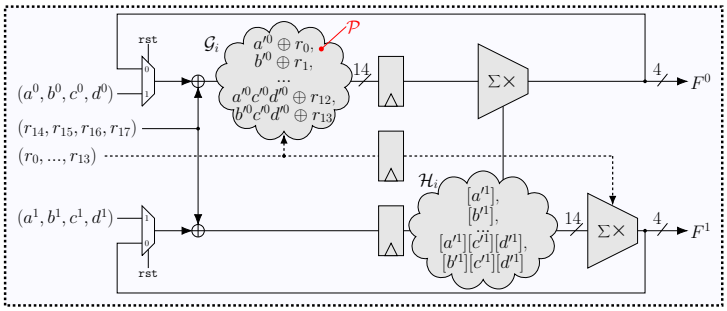


Figure 3: TSM, looped PRINCE Sbox.

can be concluded that the circuit is not 1-OPINI. We should highlight that this is not dedicated to the TSM 2-input AND circuit; any masked circuit formed following the TSM principle does not satisfy the requirements to be identified as 1-OPINI.

By this example, we have shown that a TSM circuit is not necessarily secure when it is employed in a loop, i.e., when the TSM internal registers are the sole registers of the circuit. It should be noted that such an iterated 2-input AND circuit may not be present in real cryptographic primitives; it has been utilized here only for conceptual clarification. Hence, we move forward by showing the same problem when the underlying function is the TSM PRINCE Sbox, presented by the original authors available in the corresponding GitHub repository[3] (and Artifact).

3.2.2 Sboxes

The PRINCE Sbox is defined as a function $\mathsf{F} : \mathbb{F}_2^4 \to \mathbb{F}_2^4$. Let the input bits be denoted by $\langle a, b, c, d\rangle \in \mathbb{F}_2^4$, where a is the Most Significant Bit (MSB). The output bits are denoted by $F : \langle f_3, f_2, f_1, f_0\rangle = \mathsf{F}(a, b, c, d)$, and the ANF of the outputs is given by the following equations.

$$\begin{aligned}
\mathsf{f}_0(a,b,c,d) &= 1 \oplus a \oplus b \oplus ab \oplus ad \oplus bc \oplus cd \oplus bcd \\
\mathsf{f}_1(a,b,c,d) &= 1 \oplus ac \oplus bc \oplus bd \oplus abc \oplus bcd \\
\mathsf{f}_2(a,b,c,d) &= a \oplus d \oplus ac \oplus ad \oplus cd \oplus abc \oplus acd \\
\mathsf{f}_3(a,b,c,d) &= 1 \oplus a \oplus c \oplus ab \oplus bc \oplus abd \oplus acd \oplus bcd
\end{aligned} \tag{3}$$

The TSM PRINCE Sbox design – integrated into a looped architecture – is shown in Figure 3. When the `rst` signal is enabled, the Sbox receives the primary inputs (i.e., the shares of a, b, c, and d). When `rst` is set to 0, the loop is closed, meaning that the Sbox output from the previous clock cycle becomes the next input of the Sbox.

As a first step in the theoretical security analysis, we assess the looped circuit under the glitch- and transition-extended probing model by focusing on a single probe $\mathcal{P}$ placed on the register input which stores the first output of $\mathcal{G}_i$, i.e., $a'^0 \oplus r_0$ (see Figure 3). Similar to the analysis illustrated on the 2-input AND circuit in Section 3.2.1, we assume that the given input $\langle a, b, c, d\rangle$ is constant and the `rst` signal is active only in the first clock cycle. Hence, the probe $\mathcal{P}$ in the second clock cycle observes $\mathcal{P} : \left\{a'^0_{(2)} \oplus r_{0(2)}\right\}$, which is extended via glitches to $\mathcal{P} : \left\{a^0,\ f^0_{0(2)},\ r_{14(2)},\ r_{0(2)}\right\}$, since sharing of a^0 and a^1 are refreshed by r_{14}, i.e., $a'^0 = a^0 \oplus r_{14}$ and $a'^1 = a^1 \oplus r_{14}$. Extending $f^0_{0(2)}$ via glitches would lead to several probes on the output of the top and bottom registers (see Figure 3). One of such probes is at the output of the bottom register storing a'^1. This term is required to construct the masked equation of f_0 (see Equation (3)).

[3] https://github.com/KULeuven-COSIC/TSM

Hence, we continue by focusing on a smaller probing set as $\mathcal{P}' : \left\{a^0,\ \left[a'^1_{(1)}\right],\ r_{14(2)}\right\}$. As given above, (a'^0, a'^1) is the mask refreshed version of (a^0, a^1) with r_{14}. Hence, $\mathcal{P}' : \left\{a^0,\ \left[a^1_{(1)} \oplus r_{14(1)}\right],\ r_{14(2)}\right\}$. Finally, via the transitions, we can extend $r_{14(2)}$ to additionally see $r_{14(1)}$, i.e., r_{14} in two consecutive clock cycles. Hence, we can write $\mathcal{P}' : \left\{a^0,\ \left[a^1_{(1)} \oplus r_{14(1)}\right],\ r_{14(2)},\ r_{14(1)}\right\}$. Similar to the previous analysis on the looped 2-input AND circuit, this reveals information about a, i.e., first-order vulnerability.

The same observation as the one given for the TSM 2-input AND circuit with respect to not being 1-OPINI can be given for the TSM PRINCE Sbox design. More precisely, placing the same above-explained internal probe $\mathcal{P}$ on the Sbox circuit would result in not being able to simulate $\mathcal{P}$ and $f^0_0, f^0_1, f^0_2, f^0_3$ only with a^0, b^0, c^0, d^0, and analogously not being able to simulate $\mathcal{P}$ and $f^1_0, f^1_1, f^1_2, f^1_3$ only with a^1, b^1, c^1, d^1, hence not being in conformity with the 1-OPINI notion.

The same observation holds for the TSM AES Sbox, which is also provided by the original authors in the corresponding GitHub repository[3]. Due to its higher structural complexity and the greater number of intermediate signals involved, we do not provide a detailed analysis of the leaking probes here. Instead, we refer to the supplementary material, where the evaluation results obtained using `PROLEAD` are provided.

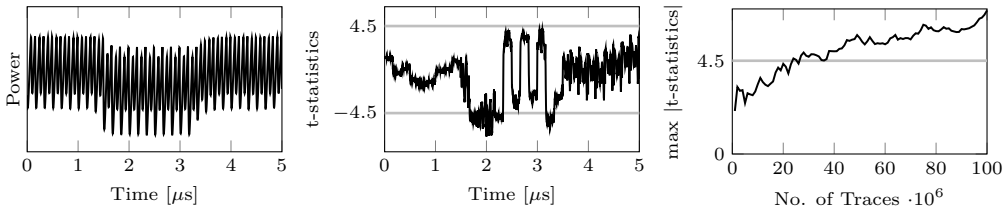
3.2.3 Experimental Validation

In addition to the above-presented theoretical evaluations (glitch- and transition-extended 1-probing model), we conducted experimental evaluations to confirm the aforementioned shortcoming. To this end, we took the TSM AES Sbox design available in the authors' GitHub repository[3], and constructed a looped circuit similar to the above-explained circuitry. In short, the circuit receives a first-order masked 8-bit input, gives this to the TSM AES Sbox, and then closes the loop, i.e., the Sbox output is given back as the next input. In other words, if the circuit runs 3 clock cycles, it generates the first-order masked output as $\mathsf{S}\Big(\mathsf{S}\big(\mathsf{S}(X)\big)\Big)$ for the given input X.

We integrated this circuit into a framework on an SCA-evaluation FPGA-based platform SAKURA-G [SAK16] while keeping the looped circuit running for 10 cycles, and conducted common and well-known fixed-vs-random t-test following the procedures given in [CDG$^+$13, SM15], also known as TVLA. Following the suggestions given in [CMM$^+$24], the 262-bit fresh mask bits (required by the TSM AES Sbox) is provided by an unrolled Trivium instance as a Pseudo-Random Number Generator (PRNG) which is updated every clock cycle. In order to follow the state of the art, we configured the PRNG to update its state at the negative-edge of the clock while the registers of the targeted design (TSM AES Sbox) are updated at the positive-edge of the clock. As suggested in [CRB$^+$16], this should reduce the noise and can deliver more confident results.

In order to collect power consumption traces of the SAKURA-G board, we employed a digital oscilloscope monitoring the voltage drop over a shunt resistor placed in the Vdd path of the target FPGA. In order to reduce the noise and obtain cleaner signals, we indeed monitored the output of the 10 db AC amplifier embedded on the SAKURA-G board. The signal has been sampled at a rate of 500 MS/s and the FPGA clock was provided by a stable jitter-free source at a frequency of 6 MHz. An overview of the measured traces can be seen in Figure 4(a).

For the first-order fixed-vs-random t-test, we provided the target circuit with either $X = \{0\}^8$ as the fixed value or a randomly selected input based on a flipped coin for each measurement. Note that in either case, the given input is freshly masked (presented by two shares) before being given to the target circuit. Similar to the common practice in state of the art, we collected 100 million traces following this scenario and estimated the t-statistics of the corresponding t-test, shown in Figure 4(b) while the maximum



(a) Mean over 1000 traces. (b) t-test, using 100 million traces. (c) t-test, over no. of traces.

Figure 4: Experimental leakage assessment results, original TSM AES Sbox in loop.

of the absolute value of t-statistics are illustrated in Figure 4(c). These results are in line with the issue explained above, i.e., first-order vulnerability of TSM circuits when employed in a loop. We should highlight that security under the robust probing model implies security in practice by analyzing power consumption traces [BDF+17], but not the other way around. Apart from the fact that the design under test does not meet the requirements for security under the glitch- and transition-extended probing model, no particular care has been devoted to addressing coupling, a common issue in FPGA-based masked designs [CEM18, CBG+17, MMM21]. Hence, the leakage observed in Figure 4 can originate from glitches-and-transitions and/or coupling.

3.2.4 More Realistic Scenario

Although we have shown that TSM circuits may fail to meet the OPINI security notion in certain compositions, it is important to clarify that such explicitly looped circuits are not present in the original PRINCE or AES encryption/decryption functions. However, similar structural patterns can be found in round-based implementations of these ciphers, where the internal registers of the TSM module are the only register layer in the encryption/decryption circuit used to store the cipher state. This implementation approach may introduce temporal dependencies between successive rounds that undermine PINI-secure composability, forcing to employ OPINI-secure modules. Therefore, the issues discussed above with respect to looped circuits also apply to round-based implementations of both PRINCE and AES, as considered in the original TSM work. Accordingly, our analysis instantiates the original TSM Sbox modules exactly as released by the authors and integrates them into round-based PRINCE and AES full cipher architectures for evaluation.

As a representative case, we focus on the round-based implementation of the PRINCE cipher. PRINCE is a lightweight block cipher that operates on 64-bit plaintext/ciphertext blocks and employs a 128-bit secret key. The full encryption/decryption process of a typical round-based implementation of the PRINCE cipher (as illustrated in Figure 5) consists of 12 rounds, each comprising the following stages:

- Key addition: each round involves an XOR with the round key and a round-dependent constant RC_i (for $i = 0, 1, \ldots, 11$).
- Substitution Layer `SB`: a nonlinear transformation is applied using the Sbox `S` in the forward rounds and its inverse $\texttt{S}^{-1}$ in the backward rounds.
- Linear Layer: This includes several components such as the diffusion layer `M` and $\texttt{M}^{-1}$ (in the forward and backward rounds, respectively), ShiftRows `SR` and its inverse $\texttt{SR}^{-1}$.

We aim to analyze the potential dependency between the input and output of each Sbox module within the round-based PRINCE cipher, which, as explained previously, can

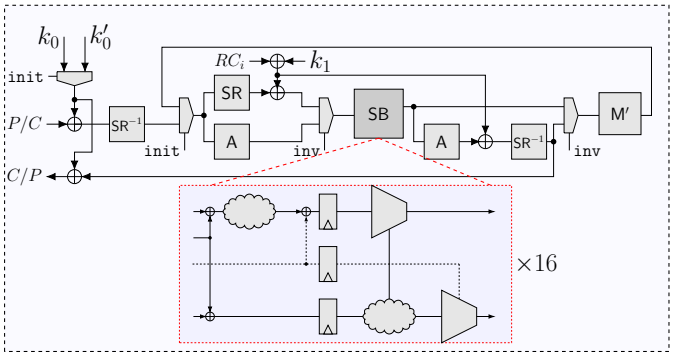


Figure 5: Architecture of the round-based TSM PRINCE encryption/decryption.

behave similarly to a looped structure in certain scenarios. As shown in Figure 5, the cipher employs several structural identities to facilitate forward and inverse operations. Notably, the inverse Sbox is defined as $\mathtt{S}^{-1} = \mathtt{A} \circ \mathtt{S} \circ \mathtt{A}$, where $\mathtt{A}(.)$ is an affine transformation. Similarly, the diffusion layers satisfy the relations $\mathtt{M} = \mathtt{SR} \circ \mathtt{M}'$ and $\mathtt{M}^{-1} = \mathtt{M}' \circ \mathtt{SR}^{-1}$, where $\mathtt{M}'$ denotes the base diffusion function, which is a multiplication of the cipher state with a constant matrix. In this round-based design, the cipher performs forward operations during the first 6 clock cycles, followed by reverse operations in the remaining 6 cycles, according to the above-mentioned methods. As a result, certain dependencies can emerge between the input and output of specific Sbox instances, forming a loop-like structure.

Since the Sbox is defined as a function $\mathtt{S} : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$, there are 16 parallel Sbox instances operating within each round. To better understand the propagation of signals across rounds, we analyze how the input to each Sbox instance is computed in different cipher rounds. Omitting the key and constant additions for brevity, in the round-based implementation, the input to each Sbox in the forward direction (i.e., rounds 1 to 5) is obtained by applying the linear transformation $\mathtt{M}$ to the outputs of the previous round's Sbox outputs. $S_{\mathrm{in}(i+1)} = \mathtt{M}(S_{\mathrm{out}(i)})$, where $S_{\mathrm{in}(i)}$ (resp. $S_{\mathrm{out}(i)}$) denotes the 64-bit input (resp. output) of the $\mathtt{SB}$ layer in the round i.

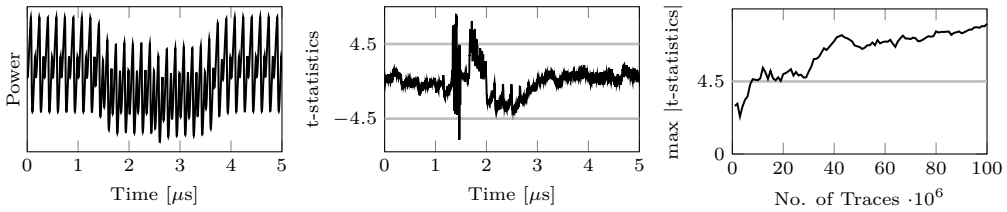
Splitting $S_{\mathrm{in}(i)}$ into 4-bit nibbles, we can write $S_{\mathrm{in}(i)} = \langle S_{\mathrm{in}_0(i)}, \ldots, S_{\mathrm{in}_{15}(i)} \rangle$, while each $S_{\mathrm{in}_j(i)}$ denotes the 4-bit input of the j-th Sbox instance in the round i. The same naturally holds for the Sbox output nibbles $S_{\mathrm{out}_j(i)}$. Due to the structure of the matrix associated with $\mathtt{M}'$, each bit of the Sbox input $S_{\mathrm{in}_j(i+1)}$ is derived as the XOR of three Sbox output bits as

$$S_{\mathrm{in}_j}[b]_{(i+1)} = S_{\mathrm{out}_{j_1}}[b_1]_{(i)} \oplus S_{\mathrm{out}_{j_2}}[b_2]_{(i)} \oplus S_{\mathrm{out}_{j_3}}[b_3]_{(i)},$$

where $S_{\mathrm{in}_j}[b]_{(i+1)}$ denotes the b-th input bit of the j-th Sbox instance in round $i+1$. Here, the indices j, j_1, j_2, j_3 as well as b, b_1, b_2, b_3 depend on the coefficients in the associated matrix of $\mathtt{M} = \mathtt{SR} \circ \mathtt{M}'$. We should note that $\mathtt{SR}$ which behaves like the AES ShiftRows and permutes the 16 nibbles in the following way

$$\mathbf{0}, 1, 2, 3, \mathbf{4}, 5, 6, 7, \mathbf{8}, 9, 10, 11, \mathbf{12}, 13, 14, 15 \;\rightarrow\; \mathbf{0}, 5, 10, 15, \mathbf{4}, 9, 14, 3, \mathbf{8}, 13, 2, 7, \mathbf{12}, 1, 6, 11.$$

This means that for $j = 0, 4, 8, 12$, one of j_1, j_2, j_3 equals j. This however does not happen for any other j. It in fact implies that for these four Sbox instances, the Sbox input depends on its own prior output. We should highlight that since the position of these four nibbles do not change via $\mathtt{SR}$, the same holds for $\mathtt{SR}^{-1}$. Therefore, in backward rounds, the input of the same four (inverse) Sbox instances (indexed by $j = 0, 4, 8, 12$) depend on their outputs. It is worth noting that we ignore analyzing the middle round, which consists of only one $\mathtt{M}'$ operation as the diffusion layer since exploiting SCA leakages of the cipher middle round would require guessing almost all secret bits.



(a) Mean over 1000 traces. (b) t-test, using 100 million traces. (c) t-test, over no. of traces.

Figure 6: Experimental leakage assessment results, original TSM round-based PRINCE encryption/decryption.

Therefore, dependencies in consecutive clock cycles may arise in the round-based TSM PRINCE circuit, in which a TSM Sbox input depends on its own output from the previous clock cycle. More precisely, considering the glitch- and transition-extension of the robust probing model, a probe placed on the input of an internal register within a TSM Sbox can propagate to the outputs of multiple Sboxes (including its own output) due to the linear diffusion layer. Similar to what we have shown for the looped circuits, this violates the inherent independence assumption (between the sharing of consecutive inputs) of PINI-only circuits.

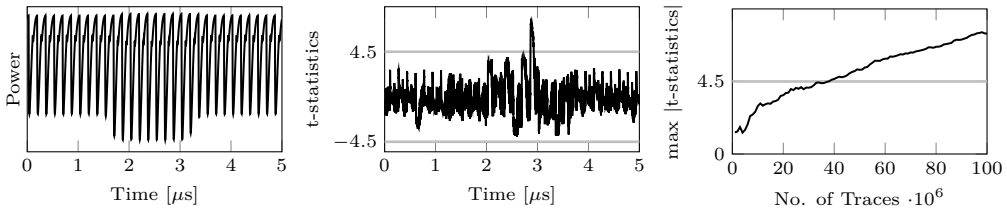
The ShiftRows operation of the AES is the same as the one employed in PRINCE, i.e., the four bytes of the first row (in the 4×4 cipher state matrix) are not shifted (the same indices given above $j = 0, 4, 8, 12$). Since the MixColumns operation of the AES would only operate on a column via XOR operations, the same observation holds. More precisely, in the round-based TSM AES encryption circuit, the input of the Sbox instances in the first row depends on their own output from the last round. Hence, the same vulnerability demonstrated earlier for the looped circuits remains valid in such round-based TSM implementations that exhibit similar structural feedback. These observations emphasize the need for composability-aware design practices even in seemingly modular, round-based architectures with only one register layer.

In short, both TSM PRINCE and AES round-based cipher implementations are insecure under the glitch- and transition-extended probing model. However, it should be noted that these implementations remain first-order secure if the transition-extension is ignored; that is, if only the glitch-extended model is taken into account. However, the leakages in practice are not a result of combinational glitches alone, but are critically dependent also on the presence of state transitions that link consecutive rounds. Therefore, both glitches and transitions should be considered when designing secure implementations.

Moreover, we repeated the same experimental analysis explained in Section 3.2.3 on the both aforementioned round-based TSM PRINCE and AES full cipher circuits. The corresponding results indicating detectable first-order leakage are shown in Figure 6 and Figure 7. As discussed in Section 3.2.3, coupling constitutes an additional source of leakage in FPGA-based masked implementations, originating from unintended interactions between physically adjacent wires and logic resources. Consequently, the detected leakage in round-based TSM implementations may stem from a combination of glitches, state transitions, and coupling.

4 Our Solutions

Having identified the limitations of the existing TSM-based implementations under the glitch- and transition-extended probing model, we provide some solutions in this section. We start with a circuit-level approach which is indeed applicable to almost any PINI gadget.



(a) Mean over 1000 traces. (b) t-test, using 100 million traces. (c) t-test, over no. of traces.

Figure 7: Experimental leakage assessment results, original TSM round-based AES-128 encryption.

We then present the construction of TSM$^+$, a lightweight variant of the original TSM scheme that reduces implementation overhead while remaining PINI-secure and preserving first-order security under the glitch-extended probing model. We describe its structure, underlying rationale, and security guarantees, followed by a formal discussion of our main contribution, OTSM, which is inspired by TSM$^+$ while additionally satisfying the OPINI composability criterion using only a single register layer.

4.1 Circuit-Level Approach

The problem of PINI circuits with one register stage when integrated in a loop architecture has been first identified in [CS21]. This indeed refers to the cases when the input of a PINI circuit depends on its former output, or in other words, the sharing of two consecutive inputs are not independent of each other. As also explained in [CS21], this can be trivially avoided by making sure that two register stages are between every input-output pair. Assuming that at least fresh masks are updated every clock cycle, the consecutive inputs will have independent sharings. This can be done on TSM circuits too, but the latency with respect to the number of clock cycles would be naturally doubled which contradicts with its central motivation.

Inspired by the pre-charging and duplicating technique of [SBHM20], every register of a TSM circuit can alternatively be implemented by the scheme shown in Figure 8, which contains two registers but not in series. The registers are equipped with synchronous reset via `rst` and $\overline{\texttt{rst}}$ signals respectively. As shown in the timing diagram of Figure 8, `rst` alternatively resets the top and bottom registers while the output y is also alternately connected to the output of the top and bottom registers $\hat{y}$ and $\check{y}$. We should highlight that this can solve the issue with a few drawbacks.

- This can only be secure under the robust-but-relaxed probing model [MM24], not under glitch- and transition-extended probing model [FGP$^+$17], which the original TSM approach is based on, since the probe placed in x would propagate to both $\hat{y}$ and $\check{y}$ via traditional glitch-extended probing model. However, AND gates controlled by `rst` and $\overline{\texttt{rst}}$ signals would stop the probe propagation in robust-but-relaxed probing model.

- This would not increase the latency with respect to the number of clock cycles, but the combinational circuit would receive null input during the half of each clock cycle. In other words, the critical path delay of the combinational circuit should be smaller than half of the clock cycle (compared to the full clock cycle in the original TSM circuit). Hence, considering the throughput at the maximum clock frequency, this does not have any advantages compared to the trivial solution by placing two registers in series.

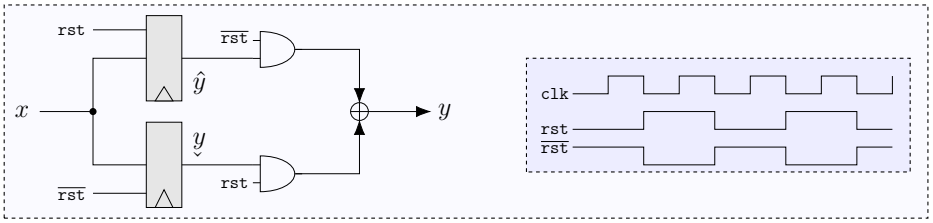


Figure 8: Doubling TSM internal registers.

Apart from these shortcomings, it should be noted that not all internal registers of a TSM circuit need to be doubled. By referring to a looped circuit, e.g., Figure 2, any probe placed on the input of the middle and the bottom registers (storing the randomness, and the blinded second input shares) would not propagate to any term consisting of the first input share. Alternatively, it can be said that only the probes placed on the input of the top register (storing $\mathcal{G}$ outputs) falsify the conditions for TSM to be 1-OPINI. Therefore, only these registers need to be doubled either in series or following the approach explained above and shown in Figure 8.

4.2 TSM$^+$

Here, we provide a detailed construction of TSM$^+$, highlighting its operation and how it modifies key aspects of the original TSM scheme. To this end, before presenting the formal definition, we first focus on a 2-input AND gate as an illustrative example to demonstrate the core principle of TSM$^+$.

4.2.1 TSM$^+$ 2-Input AND

The design of TSM$^+$ aims to simplify the original TSM structure by optimizing randomness usage and minimizing area overhead, making it more appropriate for lightweight implementations. Following the same notation as the TSM-based implementation of the 2-input AND gate, we begin by generating and refreshing the monomials associated with the second share domain of the primary inputs, as shown below.

$$x'^1 = x^1 \oplus r_0, \qquad y'^1 = y^1 \oplus r_1, \qquad (xy)'^1 = x^1 y^1 \oplus r_2$$

These refreshed values ensure that the second share of each monomial is independently blinded. Note that the first share domain of the primary inputs do not require any refreshing. The output shares, however, are computed as follows.

$$f^0 = \left[x^0\right]\left[y^1 \oplus r_1\right] \oplus \left[y^0\right]\left[x^1 \oplus r_0\right] \oplus \left[x^0 y^0 \oplus x^0 r_1 \oplus y^0 r_0 \oplus r_2\right]$$
$$f^1 = \left[x^1 y^1 \oplus r_2\right]$$

Naturally, $f^0 \oplus f^1 = x^0 y^0 \oplus x^0 y^1 \oplus x^1 y^0 \oplus x^1 y^1 = x\,y$, i.e., achieving correctness (as defined by TI [NRS11]). Further, we should highlight that the second output share f^1 includes only one quadratic term blinded by r_2 while the rest of the terms – including the correction terms $x^0 r_1 \oplus y^0 r_0$ – naturally also blinded by r_2 show up in the first output share f^0. This design is also shown in Figure 9, where $x^0 y^0 \oplus x^0 r_1 \oplus y^0 r_0 \oplus r_2$ is denoted by f_c as the correction function. We should also highlight that this construction needs 3 fresh masks while the original TSM 2-input AND requires 5 fresh mask bits (see Figure 2).

In order to show/prove that this construction is in conformity with the 1-PINI notion, we consider the following cases.

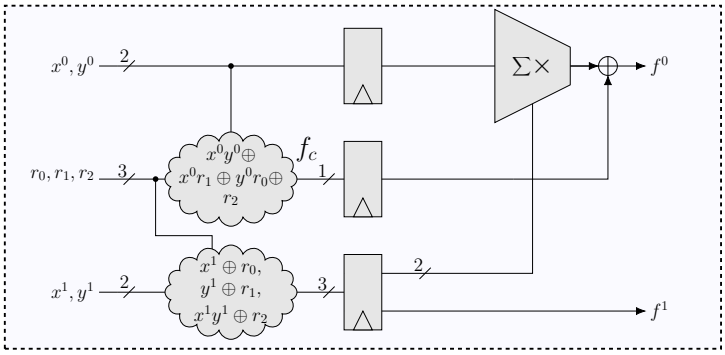


Figure 9: TSM$^+$, 2-input AND.

- A probe is placed on the first share domain output, f^0. This is propagated via glitches to $[x^0]$, $[y^0]$, $[y^1 \oplus r_1]$, $[x^1 \oplus r_0]$, $[x^0y^0 \oplus x^0r_1 \oplus y^0r_0 \oplus r_2]$, which can be simulated by x^0, y^0, i.e., first share domain inputs.
- A probe is placed on the second share domain output, f^1, which is the output of a register containing $[x^1y^1 \oplus r_2]$. This is simulatable by x^1, y^1, and drawing a random value from $\mathbb{F}_2$, i.e., second share domain inputs.
- A probe is placed on the input of any of the three bottom registers of Figure 9. This would propagate to some of x^1, y^1, r_0, r_1, and r_2. All of them are – even jointly – simulatable by the second share domain inputs x^1, y^1.
- A probe is placed on the input of the middle register or any of the two top registers of Figure 9. This would propagate to some of x^0, y^0, r_0, r_1, and r_2, which are – even jointly – simulatable by the first share domain inputs x^0, y^0.

This is fully inline with the requirements given in Definition 5, meaning that the circuit is 1-PINI. However, the requirements for being 1-SNI are not fulfilled.

4.2.2 TSM$^+$ 3-Input AND

To further illustrate the flexibility and scalability of the TSM$^+$ construction, we now extend the previous example to a 3-input AND gate. Let the primary inputs be

$$x = x^0 \oplus x^1, \quad y = y^0 \oplus y^1, \quad z = z^0 \oplus z^1,$$

and the output be $f = f^0 \oplus f^1 = xyz$. Following the same principle as in the 2-input case, we blind only the monomials corresponding to the second share domain inputs using fresh randomness r_0 to r_6:

$$\begin{aligned} x'^1 &= x^1 \oplus r_0, & y'^1 &= y^1 \oplus r_1, & z'^1 &= z^1 \oplus r_2, \\ (xy)'^1 &= x^1y^1 \oplus r_3, & (xz)'^1 &= x^1z^1 \oplus r_4, & (yz)'^1 &= y^1z^1 \oplus r_5, \\ & & (xyz)'^1 &= x^1y^1z^1 \oplus r_6. \end{aligned}$$

The output shares are then computed as:

$$\begin{aligned} f^0 =& [x^0]\,[(yz)'^1] \oplus [y^0]\,[(xz)'^1] \oplus [z^0]\,[(xy)'^1] \oplus \\ & [x^0]\,[y^0]\,[z'^1] \oplus [x^0]\,[z^0]\,[y'^1] \oplus [y^0]\,[z^0]\,[x'^1] \oplus [f_c], \\ f_c =& x^0y^0z^0 \oplus x^0r_5 \oplus y^0r_4 \oplus z^0r_3 \oplus x^0y^0r_2 \oplus x^0z^0r_1 \oplus y^0z^0r_0 \oplus r_6, \\ f^1 =& [(xyz)'^1], \end{aligned}$$

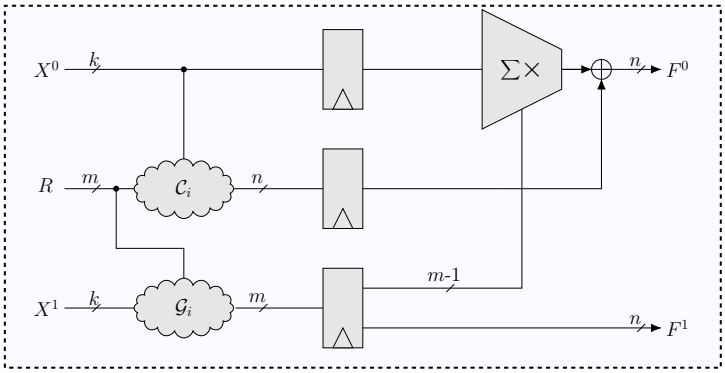


Figure 10: Architecture of our proposed TSM$^+$ scheme for an arbitrary function $\mathbb{F}_2^k \to \mathbb{F}_2^n$.

where f_c denotes the correction function. XORing the output shares yields the following Boolean function, confirming the correctness of the AND operation.

$$f^0 \oplus f^1 = x^0y^0z^0 \oplus x^0y^0z^1 \oplus x^0y^1z^0 \oplus x^0y^1z^1 \oplus x^1y^0z^0 \oplus x^1y^0z^1 \oplus x^1y^1z^0 \oplus x^1y^1z^1 = x\,y\,z$$

It is worth highlighting that this design requires only 7 bits of fresh randomness, compared to 10 bits required in the original TSM approach. In fact, the fresh masks required to refresh the primary inputs in TSM (whose denoted by R' in Figure 1) are not required in TSM$^+$, which is of the size of the number of (unmasked) primary inputs. Further, the same arguments we gave for the composable security of 2-input case (1-PINI) hold valid for this 3-input circuit too. For the sake of brevity, we avoid repeating the same justifications as both designs follow the same principle.

4.2.3 TSM$^+$ for Arbitrary Functions

In the previous examples, we demonstrated how the TSM$^+$ scheme applies to 2- and 3-input masked AND gates. We now generalize this methodology to support arbitrary Boolean functions of the form $\mathsf{F} : \mathbb{F}_2^k \to \mathbb{F}_2^n$. The general structure of a TSM$^+$-based realization for such functions is illustrated in Figure 10. To systematically construct a TSM$^+$-based masking scheme for an arbitrary function, the following steps should be followed:

- Derive the ANF of the target function.
- Determine the randomized function $\mathcal{G}_i$ by identifying the number of required fresh random bits and refreshing all second-share-domain monomials.
- Derive the final output shares from the share domains, and compute the correction term using the $\mathcal{C}_i$ block, which generates the correction function f_c to ensure correct reconstruction of the output.
- Consider logic optimization to simplify the monomial expressions.

To illustrate these steps more concretely, we apply the procedure to an end to end simple exemplary function $\mathsf{f}(x, y, z) = xy \oplus xyz$.

Step 1. The target function must first be expressed in its ANF, which represents the output as a sum (XOR) of monomials composed of input variables up to a certain degree $\mathcal{M}$. In our case, the function is already in ANF, consisting of two terms: xy and xyz, with a maximum degree of $\mathcal{M} = 3$.

Step 2. To construct $\mathcal{G}_i$, we begin by determining all product combinations of the second share domain, i.e., $x^1, y^1, z^1, x^1y^1, x^1z^1, y^1z^1, x^1y^1z^1$. The number of such terms depends on two parameters: the number of input variables k, and the maximum degree $\mathcal{M}$. Since each cross product term must be independently blinded, the number of required fresh randomness bits is equal to the number of product combinations of the second share domain and is given by $|R| = \sum_{i=1}^{\mathcal{M}} \binom{k}{i}$. In the given example, with $k = 3$ input variables and $\mathcal{M} = 3$, we have $|R| = \binom{3}{1} + \binom{3}{2} + \binom{3}{3} = 3 + 3 + 1 = 7$. Hence, seven bits of fresh randomness are required to implement the randomized function $\mathcal{G}_i$.

Step 3. To compute the output shares more formally, let us define the following notations. Below, we focus on only one of the n output bits of $\mathsf{F} : \mathbb{F}_2^k \to \mathbb{F}_2^n$ so-called $\mathsf{f} : \mathbb{F}_2^k \to \mathbb{F}_2$. The procedure is the same for every output bit.

- $\Omega = \{0, \ldots, k-1\}$ denotes the set of input variable indices.
- $\mathcal{P}_k^*$ denotes the set of all non-empty subsets of Ω.
- The ANF of the Boolean function $\mathsf{f}(.)$ is represented by a set of monomials $\mathcal{S} \subseteq \mathcal{P}_k^*$.

Computation of the First Share f^0: Each monomial in $\mathcal{S}$ is split across the shares by leveraging the distributive property. Terms involving mixed share domains (i.e., at least one variable from each share domain) are accumulated in the first output share:

$$f^0 = \bigoplus_{\mathcal{I} \in \mathcal{S},\ \mathcal{I}_0 \cup \mathcal{I}_1 = \mathcal{I}} \left(\prod_{i_0 \in \mathcal{I}_0} \left[x^0_{i_0} \right] \cdot \left[\left(\prod_{i_1 \in \mathcal{I}_1} x^1_{i_1} \right) \oplus r_{\mathcal{I}_1} \right] \right) \oplus \left[f_c \right],$$

where $\mathcal{I}_0, \mathcal{I}_1 \neq \emptyset$, $\mathcal{I}_1 = \mathcal{I} \setminus \mathcal{I}_0$, and f_c stands for the correction function explained below.

Computation of the Second Share f^1: Each monomial is evaluated solely using the second share of the inputs and blinded with a unique random bit $r_{\mathcal{I}}$:

$$f^1 = \bigoplus_{\mathcal{I} \in \mathcal{S}} \left[\left(\prod_{i \in \mathcal{I}} x^1_i \right) \oplus r_{\mathcal{I}} \right].$$

Computation of the Correction Function f_c:

$$\begin{aligned} f_c = & \bigoplus_{\mathcal{I} \in \mathcal{S},\ \mathcal{I}_0 \cup \mathcal{I}_1 = \mathcal{I}} \left(\prod_{i_0 \in \mathcal{I}_0} x^0_{i_0} \cdot r_{\mathcal{I}_1} \right) \oplus && \text{(removing the second-share blinding)} \\ & \bigoplus_{\mathcal{I} \in \mathcal{S}} \left(\prod_{i \in \mathcal{I}} x^0_i \right) \oplus && \text{(pure first share terms)} \\ & \bigoplus_{\mathcal{I} \in \mathcal{S}} r_{\mathcal{I}} && \text{(removing blinding in } f^1\text{)} \end{aligned}$$

Example: Let us now apply the generalized TSM$^+$ construction to the exemplary function $\mathsf{f}(x, y, z) = xy \oplus xyz$, which contains both quadratic and cubic terms. The monomial set corresponding to the ANF of $\mathsf{f}(.)$ is $\mathcal{S} = \big\{\{x, y\}, \{x, y, z\}\big\}$.

f^0: To compute f^0, we enumerate all valid bipartitions $(\mathcal{I}_0, \mathcal{I}_1)$ of each monomial $\mathcal{I} \in \mathcal{S}$, such that $\mathcal{I}_0 \cup \mathcal{I}_1 = \mathcal{I}$ with $\mathcal{I}_0, \mathcal{I}_1 \neq \emptyset$. Each such decomposition contributes a mixed-share term. For $\mathcal{I} = \{x, y\}$, the valid bipartitions yield to:

- $\{x\},\{y\} \Rightarrow [x^0][y^1 \oplus r_y]$
- $\{y\},\{x\} \Rightarrow [y^0][x^1 \oplus r_x]$

Thus, $f^0_{xy} = [x^0][y^1 \oplus r_y] \oplus [y^0][x^1 \oplus r_x] \oplus [f_{c,xy}]$.
For $\mathcal{I} = \{x, y, z\}$, we enumerate all six valid bipartitions as:

- $\{x\},\{y,z\} \Rightarrow [x^0][y^1 z^1 \oplus r_{yz}]$
- $\{y\},\{x,z\} \Rightarrow [y^0][x^1 z^1 \oplus r_{xz}]$
- $\{z\},\{x,y\} \Rightarrow [z^0][x^1 y^1 \oplus r_{xy}]$
- $\{x,y\},\{z\} \Rightarrow [x^0][y^0][z^1 \oplus r_z]$
- $\{x,z\},\{y\} \Rightarrow [x^0][z^0][y^1 \oplus r_y]$
- $\{y,z\},\{x\} \Rightarrow [y^0][z^0][x^1 \oplus r_x]$

Thus,

$$\begin{aligned} f^0_{xyz} = {} & [x^0][y^1 z^1 \oplus r_{yz}] \oplus [y^0][x^1 z^1 \oplus r_{xz}] \oplus [z^0][x^1 y^1 \oplus r_{xy}] \oplus \\ & [x^0][y^0][z^1 \oplus r_z] \oplus [x^0][z^0][y^1 \oplus r_y] \oplus [y^0][z^0][x^1 \oplus r_x] \oplus [f_{c,xyz}]. \end{aligned}$$

Combining both parts yields to:

$$\begin{aligned} f^0 = {} & [x^0][y^1 \oplus r_y] \oplus [y^0][x^1 \oplus r_x] \oplus \\ & [x^0][y^1 z^1 \oplus r_{yz}] \oplus [y^0][x^1 z^1 \oplus r_{xz}] \oplus [z^0][x^1 y^1 \oplus r_{xy}] \oplus \\ & [x^0][y^0][z^1 \oplus r_z] \oplus [x^0][z^0][y^1 \oplus r_y] \oplus [y^0][z^0][x^1 \oplus r_x] \oplus [f_{c,xy} \oplus f_{c,xyz}]. \end{aligned}$$

$\boldsymbol{f^1}$**:** Then, for $\mathcal{I} = \{x, y\}$ we have $f^1_{xy} = [x^1 y^1 \oplus r_{xy}]$, and for $\mathcal{I} = \{x, y, z\}$ we have $f^1_{xyz} = [x^1 y^1 z^1 \oplus r_{xyz}]$.

$\boldsymbol{f_c}$**:** Next, we compute the correction function f_c by applying the formal expression introduced earlier.
For $\mathcal{I} = \{x, y\}$: $f_{c,xy} = x^0 r_y \oplus y^0 r_x \oplus x^0 y^0 \oplus r_{xy}$.
For $\mathcal{I} = \{x, y, z\}$: $f_{c,xyz} = x^0 y^0 r_z \oplus x^0 z^0 r_y \oplus y^0 z^0 r_x \oplus x^0 r_{yz} \oplus y^0 r_{xz} \oplus z^0 r_{xy} \oplus x^0 y^0 z^0 \oplus r_{xyz}$.

Step 4. When the degree of the terms is larger than 2, logic sharing can be applied to optimize the circuit by reusing the lower-degree terms in higher-degree terms. This leads to a significant reduction in terms of gate count in practical realizations. Consider the following quadratic and cubic terms.

$$f^0_{xy} = \underbrace{[x^0][y^1 \oplus r_y] \oplus [y^0][x^1 \oplus r_x]}_{f'^0_{xy}} \oplus [\underbrace{x^0 r_y \oplus y^0 r_x \oplus x^0 y^0 \oplus r_{xy}}_{f_{c,xy}}]$$

$$\begin{aligned} f^0_{xyz} = {} & [x^0][y^1 z^1 \oplus r_{yz}] \oplus [y^0][x^1 z^1 \oplus r_{xz}] \oplus [z^0][x^1 y^1 \oplus r_{xy}] \oplus \\ & [x^0][y^0][z^1 \oplus r_z] \oplus [x^0][z^0][y^1 \oplus r_y] \oplus [y^0][z^0][x^1 \oplus r_x] \oplus \\ & [x^0 y^0 r_z \oplus x^0 z^0 r_y \oplus y^0 z^0 r_x \oplus x^0 r_{yz} \oplus y^0 r_{xz} \oplus z^0 r_{xy} \oplus x^0 y^0 z^0 \oplus r_{xyz}] \end{aligned}$$

It can be seen that the computation of xy appears as a subexpression within that of xyz. The term $f^1_{xy} = [x^1 y^1 \oplus r_{xy}]$ also appears in computation of f^0_{xyz}. Therefore, the intermediate result of xy can be reused to optimize the computation of xyz. The optimized version becomes

$$\begin{aligned} f^0_{xyz,\mathrm{opt}} = {} & [x^0][y^1 z^1 \oplus r_{yz}] \oplus [y^0][x^1 z^1 \oplus r_{xz}] \oplus [z^0] f^1_{xy} \oplus [x^0][y^0][z^1 \oplus r_z] \oplus \\ & [z^0] f'^0_{xy} \oplus [z^0 f_{c,xy} \oplus x^0 y^0 r_z \oplus x^0 r_{yz} \oplus y^0 r_{xz} \oplus r_{xyz}]. \end{aligned}$$

This form eliminates redundant computation by explicitly reusing the lower-degree term xy, allowing for reduced area. Finally, we can write

$$\begin{aligned} f^0 = &\big[x^0\big]\big[y^1z^1 \oplus r_{yz}\big] \oplus \big[y^0\big]\big[x^1z^1 \oplus r_{xz}\big] \oplus \big[z^0\big]f^1_{xy} \oplus \big[x^0\big]\big[y^0\big]\big[z^1 \oplus r_z\big] \oplus \\ &\big[z^0\big]f'^0_{xy} \oplus f'^0_{xy} \oplus \big[f_{c,xy} \oplus z^0 f_{c,xy} \oplus x^0y^0r_z \oplus x^0r_{yz} \oplus y^0r_{xz} \oplus r_{xyz}\big], \\ f^1 = &f^1_{xy} \oplus f^1_{xyz}. \end{aligned}$$

4.2.4 On Reusing the Logic

When constructing functions with multiple outputs, it is essential to take special care when reusing the logic for the optimization purposes. For example, if the given functions are $f(x, y) = xy$ and $g(x, y, z) = xyz$, one may write

$$\begin{aligned} f^0 = &\big[x^0\big]\big[y^1 \oplus r_y\big] \oplus \big[y^0\big]\big[x^1 \oplus r_x\big] \oplus \big[x^0r_y \oplus y^0r_x \oplus x^0y^0 \oplus r_{xy}\big], \\ f^1 = &\big[x^1y^1 \oplus r_{xy}\big], \end{aligned}$$

and

$$\begin{aligned} g^0 = &\big[x^0\big]\big[y^1z^1 \oplus r_{yz}\big] \oplus \big[y^0\big]\big[x^1z^1 \oplus r_{xz}\big] \oplus \big[z^0\big]\big[x^1y^1 \oplus r_{xy}\big] \oplus \\ &\big[x^0\big]\big[y^0\big]\big[z^1 \oplus r_z\big] \oplus \big[x^0\big]\big[z^0\big]\big[y^1 \oplus r_y\big] \oplus \big[y^0\big]\big[z^0\big]\big[x^1 \oplus r_x\big] \oplus \\ &\big[x^0y^0r_z \oplus x^0z^0r_y \oplus y^0z^0r_x \oplus x^0r_{yz} \oplus y^0r_{xz} \oplus z^0r_{xy} \oplus x^0y^0z^0 \oplus r_{xyz}\big], \\ g^1 = &\big[x^1y^1z^1 \oplus r_{xyz}\big]. \end{aligned}$$

Having two probes on f^0 and g^0, two register outputs $\big[x^0r_y \oplus y^0r_x \oplus x^0y^0 \oplus r_{xy}\big]$ and $\big[x^1y^1 \oplus r_{xy}\big]$ are among the probed values. These are however cannot be simulated only by first input shares x^0, y^0. To do so, x^1 and y^1 are also required, hence not in conformity with the PINI notion. As a result, f^1 cannot be reused in construction of g^0. In other words, each coordinate function should have its own mask r_{xy}. Generally, this issue may happen if it is possible to write one coordinate function $g(.)$ based on another coordinate function $f(.)$ as $g(.) = f(.)g'(.) \oplus g''(.)$, when $g'(.)$ contains only a single monomial.

4.3 OTSM

Following the analysis showing that the previous constructions, such as TSM and TSM$^+$ do not fulfill the requirements of the OPINI security notion, we now present our main contribution, referred to as the OTSM scheme. OTSM is specifically designed to close this composability gap by providing provable 1-OPINI security under the glitch- and transition-extended probing model, while still relying on a single register layer and preserving first-order security. In the following, we return to the masked AND gate and illustrate the OTSM approach. This example provides a minimal yet representative case for understanding the scheme's structure, its use of randomness, and its security properties.

4.3.1 OTSM 2-Input AND

Following the same notation, let (x^0, x^1) and (y^0, y^1) be the two shares of x and y, respectively. The goal is to securely compute $f = f^0 \oplus f^1 = x\,y$ while ensuring that the resulting circuit satisfies first-order security and is in conformity with the OPINI notion. In the following, we describe how input refreshing, randomness distribution, and output computation are done in the OTSM 2-input AND.

The concept is similar to that of TSM$^+$, but inspired by the HPC4 gadgets [CSV24], the blinding of cross-share-domain monomials are done by two independent fresh masks and the correction terms are also split into two parts avoiding every two fresh masks

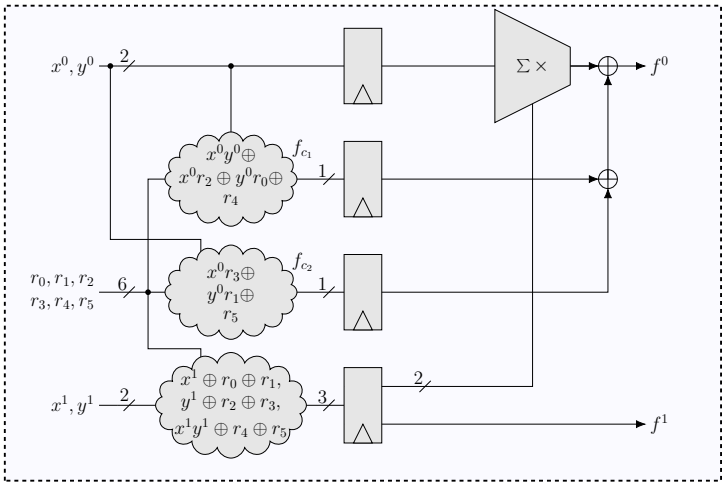


Figure 11: OTSM, 2-input AND.

involved in blinding a monomial do not appear in a correction function. This crucial property is analyzed and validated in the subsequent security evaluation part. Hence, monomials derived from the second input shares are blinded as follows.

$$x'^1 = x^1 \oplus r_0 \oplus r_1, \qquad y'^1 = y^1 \oplus r_2 \oplus r_3, \qquad (xy)'^1 = x^1y^1 \oplus r_4 \oplus r_5,$$

and the output shares are computed as outlined below.

$$\begin{aligned} f^0 &= \left[x^0\right]\left[y'^1\right] \oplus \left[y^0\right]\left[x'^1\right] \oplus \left[f_{c_1}\right] \oplus \left[f_{c_2}\right] \\ f_{c_1} &= x^0y^0 \oplus x^0r_2 \oplus y^0r_0 \oplus r_4 \\ f_{c_2} &= x^0r_3 \oplus y^0r_1 \oplus r_5 \\ f^1 &= \left[(xy)'^1\right] \end{aligned}$$

Correctness is trivially validated by XORing the output shares, i.e., $f^0 \oplus f^1$, which reconstructs the correct result $x\,y$.

It is important to note that this design requires 6 bits of fresh randomness, which is twice the amount needed in the corresponding TSM$^+$ 2-input AND. Additionally, this design includes two correction functions f_{c_1} and f_{c_2}, whereas the TSM$^+$ circuit requires only one. This increase in complexity stems from the fact explained above, i.e., blinding each second input share monomial with two fresh masks and splitting the correction function. The overall structure of the design is illustrated in Figure 11.

In the following, we aim to prove that this design satisfies 1-OPINI security notion by analyzing different representative cases considering Definition 6.

- A probe placed on the output share f^0 would trivially be simulated by only x^0, y^0, and analogously a probe placed on the other output share f^1 by only x^1, y^1.
- A probe placed on the input of the top register (in Figure 11) reveals either x^0 or y^0. Therefore, the output share f^0 must be simulatable by x^0 and y^0 too. The simulator requires to know the values of $[f_{c_1}]$, $[f_{c_2}]$, and that of the expressions $[x^1 \oplus r_0 \oplus r_1]$ and $[y^1 \oplus r_2 \oplus r_3]$. However, since no information about r_4 and r_5 can be revealed via these values, the simulation can be performed using only the first input shares, x^0 and y^0 and drawing a few random value from $\mathbb{F}_2$.
- A probe placed on the output of f_{c_1} is propagated to x^0, y^0, r_0, r_2, and r_4. Therefore, the output share f^0 must be simulatable. To do so, the simulator needs access to the

values $[x^0]$, $[y^0]$, $[f_{c_1}]$, $[f_{c_2}]$, as well as the value of the expressions $[x^1 \oplus r_0 \oplus r_1]$ and $[y^1 \oplus r_2 \oplus r_3]$. However, since the probed intermediate node does not have access to r_1 and r_3, it cannot extract any information about x^1 or y^1. Consequently, the simulation can be performed using only the first input shares, x^0 and y^0 and drawing a few random value from $\mathbb{F}_2$.

- A probe placed on the output of f_{c_2} propagates to x^0, y^0, r_1, r_3, and r_5. Therefore, the output share f^0 must be simulatable as well. Hence, the simulator needs access to the values $[x^0]$, $[y^0]$, $[f_{c_1}]$, $[f_{c_2}]$, as well as the value of the expressions $[x^1 \oplus r_0 \oplus r_1]$ and $[y^1 \oplus r_2 \oplus r_3]$. However, since the probed intermediate node does not have access to r_0 and r_2, it cannot extract any information about x^1 or y^1. So, the simulation can be performed using only the first input shares, x^0 and y^0 and drawing a few random value from $\mathbb{F}_2$.
- A probe placed on the input of one of the bottom registers (in Figure 11) propagates to x^1 and/or y^1 and additionally to r_0 and r_1 or alternatively to r_2 and r_3 or to r_4 and r_5. Therefore, the output share f^1 must be simulatable as well. This is however trivially possible using x^1, y^1 and in some cases by drawing random values from $\mathbb{F}_2$ for r_4 and r_5.

This completes all cases showing that our OTSM 2-input AND is in conformity with OPINI notion with $t = 1$.

4.3.2 OTSM for Arbitrary Functions

The procedure to apply OTSM on an arbitrary function is very similar to that of TSM$^+$. The number of fresh mask bits doubles the amount required in TSM$^+$, i.e., $2\sum_{i=1}^{\mathcal{M}} \binom{k}{i}$. Below we give the corresponding equations considering the same notations given for TSM$^+$.

$$f^0 = \bigoplus_{\mathcal{I}\in\mathcal{S},\ \mathcal{I}_0\cup\mathcal{I}_1=\mathcal{I}} \left(\prod_{i_0\in\mathcal{I}_0} \left[x^0_{i_0}\right] \cdot \left[\left(\prod_{i_1\in\mathcal{I}_1} x^1_{i_1} \right) \oplus r_{\mathcal{I}_1} \oplus r'_{\mathcal{I}_1} \right] \right) \oplus \left[f_{c_1}\right] \oplus \left[f_{c_2}\right]$$

$$f^1 = \bigoplus_{\mathcal{I}\in\mathcal{S}} \left[\left(\prod_{i\in\mathcal{I}} x^1_i \right) \oplus r_{\mathcal{I}} \oplus r'_{\mathcal{I}} \right]$$

$$f_{c_1} = \bigoplus_{\mathcal{I}\in\mathcal{S},\ \mathcal{I}_0\cup\mathcal{I}_1=\mathcal{I}} \left(\prod_{i_0\in\mathcal{I}_0} x^0_{i_0} \cdot r_{\mathcal{I}_1} \right) \oplus \bigoplus_{\mathcal{I}\in\mathcal{S}} \left(\prod_{i\in\mathcal{I}} x^0_i \right) \oplus \bigoplus_{\mathcal{I}\in\mathcal{S}} r_{\mathcal{I}}$$

$$f_{c_2} = \bigoplus_{\mathcal{I}\in\mathcal{S},\ \mathcal{I}_0\cup\mathcal{I}_1=\mathcal{I}} \left(\prod_{i_0\in\mathcal{I}_0} x^0_{i_0} \cdot r'_{\mathcal{I}_1} \right) \oplus \bigoplus_{\mathcal{I}\in\mathcal{S}} r'_{\mathcal{I}}$$

We should highlight that the justifications/proof given for the 2-input AND for being 1-OPINI are valid and identical when OTSM is applied on an arbitrary function. Further, the optimization given to TSM$^+$– with respect to using low-degree terms when constructing higher-degree terms – are valid and applicable on OTSM as well. Since they are very similar, we omit repeating the same procedure for brevity.

4.4 On the Number of Required Fresh Masks

As given in Table 1, a TSM circuit needs at most $2^k + k - 2$ fresh masks for an $\mathbb{F}_2^k \to \mathbb{F}_2^k$ function of an algebraic degree $k - 1$. TSM$^+$ reduces this by k bits, i.e., requires at most $2^k - 2$ fresh masks when realizing the same function. This is due to the absence of SNI security in TSM$^+$. OTSM, however, needs double the amount of randomness of TSM$^+$, i.e., $2(2^k - 2) = 2^{k+1} - 4$ fresh mask bits when realizing the same function in conformity with the OPINI security notion.

Table 2: Synthesis results of the PRINCE Sbox constructed by different masking schemes using Synopsys Design Compiler (Version M-2016.12) and NanGate 45 nm library.

Scheme	Area (GE)	Max Freq. (MHz)	Random (bits)	Latency (cycles)	Security Notion
TSM [VDB+24]	462	1190	18	1	PINI, SNI
TSM$^+$ [new]	414	854	14	1	PINI
OTSM [new]	573	787	28	1	OPINI

5 Case Studies

In addition to the exemplary small functions presented in Section 4, we applied TSM$^+$ and OTSM on both PRINCE and AES Sboxes to enable fair comparisons with the original TSM designs presented in [VDB+24]. We applied the optimizations – explained in Section 4.2.3 – on such larger functions, where many identical terms naturally arise in the ANF representation. These optimizations contribute significantly to minimizing the overall hardware cost of the masked implementations without compromising security.

5.1 PRINCE

We first consider the PRINCE Sbox, whose small logic footprint provides a clear view of the cost and efficiency of each construction. As summarized in Table 2, the TSM$^+$ design outperforms the original TSM in terms of area, while demanding for a slower clock frequency. On the other hand, the OTSM implementation – which naturally requires higher area and lower clock frequency – maintains a single-cycle latency and fulfills the OPINI security notion. In terms of randomness, TSM$^+$ reduces the number of required fresh masks from 18 to 14, while OTSM needs $2 \times 14 = 28$ due to its dual-blinding strategy to be in conformity with the OPINI notion. We should highlight that – to ensure a fair comparison – we took the original TSM RTL implementation from the authors' public repository, and did the synthesis using our toolchain with open-source NanGate 45 nm standard cell library. We further made sure that the synthesizer does not optimize the designs over the boundaries, i.e., keeping the hierarchy of the given designs.

Note that TSM$^+$ generally needs a lower maximum clock frequency than TSM. As detailed in Section 4.2.3, this stems from the asymmetry in computation of output shares; the second output share includes only a single monomial per ANF term, while all remaining monomials are accumulated in the first output share. This results in a deeper combinational logic path for the first output share, which naturally increases the critical path delay.

As discussed in Section 3.2.4, it might not be necessary to instantiate 16 OPINI Sboxes in a round-based implementation of the PRINCE cipher to maintain security. More precisely, it should be sufficient to instantiate OTSM only for the four nibbles at positions 0, 4, 8, and 12, while using the more efficient TSM$^+$ construction for the remaining twelve nibbles. However, we should review the underlying round-based design architecture shown in Figure 5. Due to the multiplexers controlled by `inv` – to enable forward and backward rounds – any probe placed on an Sbox input would propagate to its output via glitch-extended probing model. However, since `inv` is only supposed to change at round 6, such probes would not propagate that way under the robust-but-relaxed probing model. This hybrid approach enables a first-order secure round-based PRINCE implementation with one clock cycle latency per round, achieving security under only robust-but-relaxed probing model. In contrast, the original TSM-based design only ensures security under the glitch-extended model. Interestingly, our hybrid TSM$^+$–OTSM implementation – despite offering security – requires only $14 \times 12 + 28 \times 4 = 280$ fresh random bits per encryption round, compared to $18 \times 16 = 288$ bits in the original TSM design. This reduction is due to the fact that TSM$^+$ requires fewer fresh randomness per Sbox than TSM, and its

Table 3: Synthesis results of full PRINCE encryption/decryption.

Scheme	Area (GE)	Max Freq. (MHz)	Random (bits)	Latency (cycles)	Area w/ PRNG (GE)	Secure†
TSM [VDB$^+$24]	10 022	746	288	12	35 301	✗
TSM$^+$–OTSM [new]	10 243	714	280	12	33 897	✓*
OTSM [new]	12 326	709	448	12	38 575	✓

† Security guarantee considering both glitches and transitions.

* Under the robust-but-relaxed probing model [MM24].

integration dominates the design. Alternatively, if security under traditional glitch- and transition-extended probing model is desired, 16 OTSM Sbox instances should be integrated into the design. This naturally increase the area and the demand for randomness.

Since the authors of TSM [VDB$^+$24] only provided their Sbox design, to ensure a fair comparison, we plugged their TSM Sbox into our full PRINCE encryption/decryption framework, which follows the same architecture as shown in Figure 5, which is based on [MS16] also used in [VDB$^+$24]. The synthesis results are summarized in Table 3, showing that our hybrid design offers improved security while reducing the overall demand for randomness. To evaluate the overall implementation cost including randomness generation, we adopted a PRNG based on an unrolled Trivium as suggested in [CMM$^+$24], ensuring sufficient fresh masks per cycle. As also shown in Table 3, the PRNG introduces its area overhead. However, the overall area of our hybrid design becomes less than the TSM counterpart. We should also mention that this PRNG has a very short critical path delay; hence not affecting the overall maximum clock frequency of the designs.

5.2 AES

Following the same methodology as for the PRINCE Sbox, we applied both TSM$^+$ and OTSM to the AES Sbox, as well as to a round-based encryption function following a classical 11-cycle design architecture. The synthesis results for the standalone AES Sbox are given in Table 4. To ensure fair comparison with the original TSM design, all implementations were synthesized using an identical toolchain and constraints. We have set the synthesizer to achieve a maximum frequency of 333 MHz – closely matching the 349 MHz of the TSM reference implementation.[4] It can be seen that our TSM$^+$ design requires more area compared to the TSM design although reducing the demand for fresh randomness. However, considering the area required for the generation of randomness, TSM$^+$ outperforms the original TSM. We should highlight that a TSM$^+$ circuit generally requires less randomness compared to the corresponding TSM circuit. However, less area footprint (excluding the part required for the generation of randomness) is not guaranteed. How the RTL is generated and how the synthesis is performed can affect the area efficiency of the resulting circuit.

In the follow-up work [VDB$^+$25], a more efficient version of the first-order TSM AES Sbox is also presented. This design is made by dividing the circuit into two parts leading to two register stages in the Sbox, i.e., two clock cycles per Sbox evaluation. Hence, being in conformity with PINI is actually enough to construct a first-order round-based implementation of the AES encryption function secure under glitch- and transition-extended probing model while doubling the latency.

Similar to the PRINCE, the authors of the original TSM work [VDB$^+$24] did not provide their round-based AES design. To enable a fair comparison, we implemented the AES encryption function ourselves, and plugged the original TSM Sbox, the 2-stage TSM Sbox, and a hybrid design composed of OTSM and TSM$^+$ Sboxes. Following the same

[4] Without such a restriction our TSM$^+$ and OTSM designs require around 100 GE less area with maximum clock frequency of around 222 MHz.

Table 4: Synthesis results of the AES Sbox.

Scheme	Area (GE)	Max Freq. (MHz)	Random (bits)	Latency (cycles)	Security Notion	Area w/ PRNG (GE)
TSM [VDB+24]	14 333	349	262	1	PINI, SNI	34 268
TSM [VDB+25]	3 990	561	46	2	PINI, SNI	7 242
TSM$^+$ [new]	15 144	333	254	1	PINI	33 453
OTSM [new]	20 055	333	508	1	OPINI	56 658

rationale as in Section 3.2.4, the OTSM Sbox instances are assigned to state bytes indexed by $0, 4, 8, 12$, which are known to experience temporal transitions between consecutive rounds. The other twelve state bytes are associated with TSM$^+$ Sbox instances. Note that the problem associated with the multiplexers in the forward and backward rounds of the PRINCE round-based implementation does not hold for the AES round-based design. Therefore, our hybrid design can ensure security under the glitch- and transition-extended probing model (not only under robust-but-relaxed model). The corresponding synthesis results are shown in Table 5, indicating that our TSM$^+$–OTSM design is the only secure one with 11 clock cycle latency when both glitches and transitions are considered while it demands for higher area compared to the original TSM designs.

Table 5: Synthesis results of full AES-128 encryption.

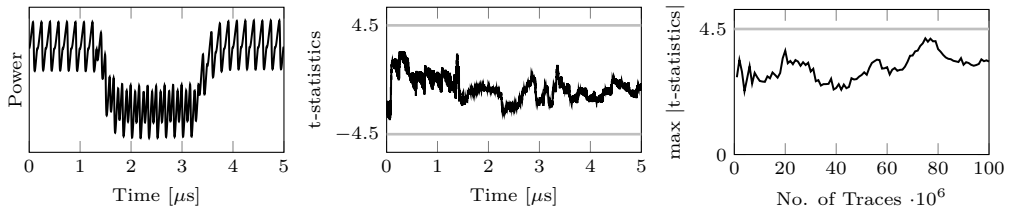
Scheme	Area (GE)	Max Freq. (MHz)	Random (bits)	Latency (cycles)	Secure†
TSM [VDB+24]	236 721	310	4 192	11	✗
TSM [VDB+25]	72 070	561	736	21	✓
TSM$^+$–OTSM [new]	269 878	303	5 080	11	✓

† Security guarantee considering both glitches and transitions.

5.3 Evaluations

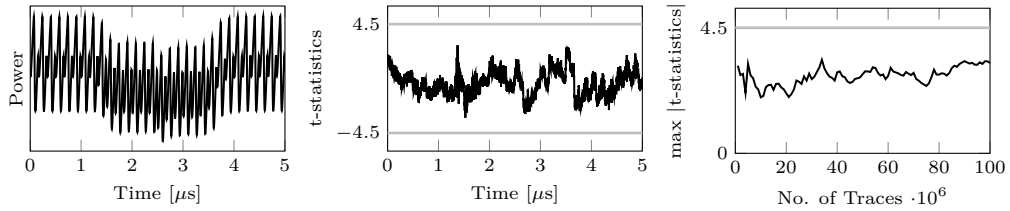
We evaluated all of our designs and claims using PROLEAD [MM22] under both glitch- and transition-extended as well as robust-and-relaxed probing models depending on the design being evaluated. The corresponding designs and evaluation results are publicly available via GitHub[2].

As a sanity check, we evaluated the leakage of our OTSM AES Sbox using the same experimental setup and under the same condition following the same procedure described in Section 3.2.3, where the masked Sbox is repeatedly evaluated in a loop. Figure 12 shows the mean trace, fixed-vs-random t-test (100M traces), and t-value evolution. No t-statistic exceeds the ± 4.5 threshold, confirming first-order security. We further conducted the same evaluation on our round-based full cipher PRINCE encryption/decryption and round-based AES-128 encryption designs composed of OTSM and TSM$^+$ Sboxes, as explained in Section 5.1 and Section 5.2, respectively. The corresponding results reporting no detectable first-order leakage, are given in Figure 13 and Figure 14.



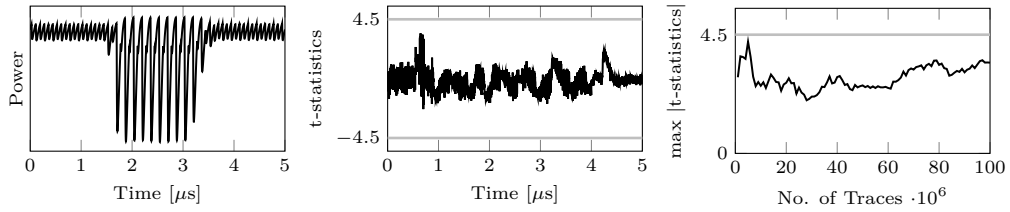
(a) Mean over 10^3 traces. (b) t-test, using 100 million traces. (c) t-test, over no. of traces.

Figure 12: Experimental leakage assessment results, OTSM AES Sbox in loop.



(a) Mean over 1000 traces. (b) t-test, using 100 million traces. (c) t-test, over no. of traces.

Figure 13: Experimental leakage assessment results, TSM$^+$–OTSM round-based PRINCE encryption/decryption.



(a) Mean over 1000 traces. (b) t-test, using 100 million traces. (c) t-test, over no. of traces.

Figure 14: Experimental leakage assessment results, TSM$^+$–OTSM round-based AES-128 encryption.

6 Conclusions

In this work, we revisited the TSM scheme and demonstrated its vulnerability when used in round-based architectures under the glitch- and transition-extended probing model. While TSM achieves low latency and satisfies the SNI and PINI composability notions, we showed both theoretically and experimentally that it may fail to provide first-order security in single-register feedback settings due to its incompatibility with the OPINI notion.

To address this issue, we proposed two improved constructions. First, TSM$^+$, a more efficient variant of TSM, preserves the low-latency design and reduces the number of fresh masks while remaining PINI-compliant. Second, OTSM extends TSM$^+$ by additionally satisfying the OPINI notion, thereby enabling secure composition in round-based masked implementations, even when feedback from output to input exists across clock cycles. We should highlight that none of our constructions is in conformity with 1-SNI notion. This, however, does not harm the composability since being PINI is sufficient (and usually more efficient than being SNI) when constructing large circuits by composing smaller modules.

We further showed that not necessarily all Sboxes of round-based implementations of PRINCE and AES need to be OPINI-compliant. Instead, a careful hybrid composition of PINI and OPINI Sboxes can ensure first-order security. Our secure masked implementations of PRINCE and AES exemplify this strategy, demonstrating improved security guarantees. Our findings highlight the importance of composability-aware design in low-latency masking and establish practical guidelines for secure cryptographic hardware under realistic leakage.

Acknowledgments

The work described in this paper has been supported in part by the German Research Foundation (DFG) through the projects 535533866 (SSIMA) and 549340884 (MatSec). We acknowledge Lixuan Wu from Shandong University, China who identified the issue given in Section 4.2.4.

References

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified Proofs of Higher-Order Masking. In *EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 457–485. Springer, 2015.

[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 *CCS 2016*, pages 116–129. ACM, 2016.

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

[BDF+17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel Implementations of Masking Schemes and the Bounded Moment Leakage Model. In *EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, 2017.

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

[CDG+13] Jeremy Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, Pankaj Rohatgi, et al. Test vector leakage assessment (TVLA) methodology in practice. In *International Cryptographic Module Conference*, volume 20, 2013.

[CEM18] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware Masking, Revisited. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):123–148, 2018.

[CGP+12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of Security Proofs from One Leakage Model to Another: A New Issue. In *COSADE 2012*, volume 7275 of *Lecture Notes in Computer Science*, pages 69–81. Springer, 2012.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CMM+24] Gaëtan Cassiers, Loïc Masure, Charles Momin, Thorben Moos, Amir Moradi, and François-Xavier Standaert. Randomness Generation for Secure Hardware Masking - Unrolled Trivium to the Rescue. *IACR Commun. Cryptol.*, 1(2):4, 2024.

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

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[CS21] Gaëtan Cassiers and François-Xavier Standaert. Provably Secure Hardware Masking in the Transition- and Glitch-Robust Probing Model: Better Safe than Sorry. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):136–158, 2021.

[CSV24] Gaëtan Cassiers, François-Xavier Standaert, and Corentin Verhamme. Low-Latency Masked Gadgets Robust against Physical Defaults with Application to Ascon. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(3):603–633, 2024.

[FGP$^+$17] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults and the robust probing model. *IACR Cryptol. ePrint Arch.*, page 711, 2017.

[GIB18] Hannes Groß, Rinat Iusupov, and Roderick Bloem. Generic Low-Latency Masking in Hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):1–21, 2018.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In *TIS@CCS 2016*, page 3. ACM, 2016.

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

[GSM17] Hannes Groß, David Schaffenrath, and Stefan Mangard. Higher-Order Side-Channel Protected Implementations of KECCAK. In *DSD 2017*, pages 205–212. IEEE Computer Society, 2017.

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

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KM22] David Knichel and Amir Moradi. Low-Latency Hardware Private Circuits. In *CCS 2022*, pages 1799–1812. ACM, 2022.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KSM22] David Knichel, Pascal Sasdrich, and Amir Moradi. Generic Hardware Private Circuits Towards Automated Generation of Composable Secure Gadgets. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):323–344, 2022.

[MM22] Nicolai Müller and Amir Moradi. PROLEAD A Probing-Based Hardware Leakage Detection Tool. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):311–348, 2022.

[MM24] Nicolai Müller and Amir Moradi. Robust but Relaxed Probing Model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(4):451–482, 2024.

[MMM21] Nicolai Müller, Thorben Moos, and Amir Moradi. Low-Latency Hardware Masking of PRINCE. In *COSADE 2021*, volume 12910 of *Lecture Notes in Computer Science*, pages 148–167. Springer, 2021.

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

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[MPL+11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.

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

[MRB18] Lauren De Meyer, Oscar Reparaz, and Begül Bilgin. Multiplicative Masking for AES in Hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):431–468, 2018.

[MS16] Amir Moradi and Tobias Schneider. Side-Channel Analysis Protection and Low-Latency in Action – Case Study of PRINCE and Midori –. In *ASIACRYPT 2016*, volume 10031 of *Lecture Notes in Computer Science*, pages 517–547, 2016.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptol.*, 24(2):292–321, 2011.

[SAK16] SAKURA. Side-channel Attack User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html, 2016.

[SBHM20] Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E. Marson. Low-Latency Hardware Masking with Application to AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):300–326, 2020.

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

[SM21] Aein Rezaei Shahmirzadi and Amir Moradi. Second-Order SCA Security with almost no Fresh Randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):708–755, 2021.

[VDB+24] Dilip Kumar S. V., Siemen Dhooghe, Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. Time Sharing - A Novel Approach to Low-Latency Masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(3):249–272, 2024.

[VDB+25] Dilip Kumar S. V., Siemen Dhooghe, Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. Higher-Order Time Sharing Masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(2):235–267, 2025.